Eric Bonjour · Daniel Krob
Luca Palladino · François Stephan
Editors

Complex Systems Design & Management

Proceedings
of the Ninth International Conference
on Complex Systems Design
& Management
CSD&M Paris 2018

Springer

Complex Systems Design & Management

Eric Bonjour · Daniel Krob
Luca Palladino · François Stephan
Editors

Complex Systems Design & Management

Proceedings of the Ninth International Conference on Complex Systems Design & Management, CSD&M Paris 2018

Springer

Editors
Eric Bonjour
Université de Lorraine
Laxou, France

Luca Palladino
Safran
Magny Les Hameaux, France

Daniel Krob
CESAMES
Paris, France

François Stephan
Be-Bound
Marnes La Coquette, France

ISBN 978-3-030-04208-0 ISBN 978-3-030-04209-7 (eBook)
https://doi.org/10.1007/978-3-030-04209-7

Library of Congress Control Number: 2018960915

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Introduction

This volume contains the Proceedings of the Ninth International Conference on "Complex Systems Design & Management" (CSD&M 2018; see the conference Web site: http://www.2018.csdm.fr/ for more details).

The CSD&M 2018 conference was jointly organized on 18–19 December 2018 at the Cité Internationale Universitaire de Paris (France) by the three following partners:

1. CESAM Community managed by the Center of Excellence on Systems Architecture, Management, Economy & Strategy (CESAMES);
2. AFIS, Association Française d'Ingénierie Système, the French Chapter of the International Council on Systems Engineering (INCOSE);
3. The Ecole Polytechnique; ENSTA ParisTech; Télécom ParisTech; Dassault Aviation; Naval Group; DGA; Thales "Engineering of Complex Systems" chair.

The conference also benefited from the technical and financial support of many organizations such as Airbus Apsys, Alstom Transport, ArianeGroup, INCOSE, MEGA International, Renault and Thales. Our sincere thanks therefore to all of them.

Then, many other organizations have been involved in the CSD&M 2018 Committee. We would like to thank all their members who helped a lot through their participation during the one-year preparation of the conference.

Why a CSD&M Conference?

Mastering complex systems require an integrated understanding of industrial practices as well as sophisticated theoretical techniques and tools. This explains the creation of an annual *go-between* forum at European level (which does not exist yet) both dedicated to academic researchers and industrial actors working on complex industrial systems architecture and engineering. Facilitating their *meeting* was actually for us a *sine qua non* condition in order to nurture and develop in Europe the science of systems which is currently emerging.

The purpose of the "Complex Systems Design & Management" (CSD&M) conference is exactly to be such a forum. Its aim, in time, is to become *the* European academic–industrial conference of reference in the field of complex industrial systems architecture and engineering, which is a quite ambitious objective. The last eight CSD&M Paris conferences—which were all held in the last quarter from 2010 to 2017 in Paris—were the first steps in this direction. In 2017, participants were again almost 250 to attend the two-day conference which proves that the interest for architecture and systems engineering does not fade.

Our Core Academic–Industrial Dimension

To make the CSD&M conference a convergence point of the academic and industrial communities in complex industrial systems, we based our organization on a principle of *parity* between academics and industrialists (see the conference organization sections in the next pages). This principle was first implemented as follows:

- Program Committee consisted of 50% academics and 50% industrialists,
- Invited Speakers came in a balanced way from numerous professional environments.

The set of activities of the conference followed the same principle. They indeed consist of a mixture of research seminars and experience sharing, academic articles and industrial presentations, software and training offer presentations, etc. The conference topics cover the most recent trends in the emerging field of complex systems sciences and practices from an industrial and academic perspective, including the main industrial domains (aeronautics and aerospace, transportation and systems, defense and security, electronics and robotics, energy and environment, health care and welfare services, media and communications, software and e-services), scientific and technical topics (systems fundamentals, systems architecture and engineering, systems metrics and quality, systemic tools), and system types (transportation systems, embedded systems, software and information systems, systems of systems, artificial ecosystems).

The 2018 Edition

The CSD&M Paris 2018 edition received 52 submitted papers, out of which the Program Committee selected 19 regular papers to be published in the conference proceedings. A 37% acceptance ratio was reached which guarantees the high quality of the presentations. The Program Committee also selected 16 papers for a collective presentation during the poster workshop of the conference.

Each submission was assigned to at least two Program Committee members, who carefully reviewed the papers, in many cases with the help of external referees. These reviews were discussed by the Program Committee Co-chairs during an online meeting by 26 June 2018 and managed via the EasyChair conference system.

We also chose several outstanding speakers with industrial and scientific expertise who gave a series of invited talks covering all the spectrum of the conference during the two days of CSD&M Paris 2018. The conference was organized around a common topic: "*Products and Services Development in a Digital World.*" Each day proposed various invited keynote speakers' presentations and a "à la carte" program consisting in accepted papers' presentations and in different sessions (thematic tracks on Day 1 and sectoral tracks on Day 2).

Furthermore, we had a "poster workshop", to encourage presentation and discussion on interesting but "not-yet-polished" ideas. CSD&M Paris 2018 also offered booths presenting the last engineering and technological news to participants.

August 2018

Eric Bonjour
Daniel Krob
Luca Palladino
François Stephan

Conference Organization

Conference Chairs

General Chair

Daniel Krob	CESAMES and Ecole Polytechnique, France

Organizing Committee Chair

François Stephan	Be-bound, France

Program Committee Co-chairs

Eric Bonjour (Academic Co-chair)	Université de Lorraine, France
Luca Palladino (Industrial Co-chair)	Safran, France

Program Committee

The Program Committee consists of 21 members (10 academics and 11 industrialists) of high international visibility. Their expertise spectrum covers all of the conference topics.

Academic Members

Co-chair

Eric Bonjour	Université de Lorraine, France

Members

Vincent Chapurlat	Mines Ales, France
David Flanigan	Chesapeake INCOSE Chapter, USA
Cecilia Haskins	NTNU, Norvegia
Neil Handen Ergin	Systems Engineering Penn State University, USA
Eric Levrat	Université de Lorraine, France
Anja Maier	Technical University of Denmark, Denmark
Eduarda Pinto Ferreira	ISEP-IPP, Portugal
Donna Rhodes	MIT, USA
Zoe Szajnfarber	George Washington University, USA

Industrial Members

Co-chair

Luca Palladino	Safran, France

Members

Ifede Joel Adounkpe	PSA, France
Raphael Faudou	Samares, France
Davide Fierro	INAF, Italy
Annabelle Meunier-Schermann	DGA (State Organization), France
Aurelijus Morkevicius	No Magic, Lithuania
Frederic Paci	Zodiac, France
Amaury Soubeyran	Airbus, France
Lawrence Toby	Jaguar Land Rover, UK
Lonnie Vanzandt	Sodius, USA
Christophe Waterplas	ResMed, Australia

Organizing Committee

The Organizing Committee consists of 18 members (academics and industrialists) of high international visibility. The Organizing Committee is in charge of defining the program of the conference, identifying keynote speakers, and has to ensure the functioning of the event (sponsoring, communication, etc.).

Organizing Committee

Chair

François Stephan	Be-bound, France

Members

Patrick Anglard	Assystem, France
Emmanuel Arbaretier	Airbus, France
Jean-François Bigey	MEGA International, France
Philippe Bourguignon	Engie, France
Guy-André Boy	Estia Institute of Technology, France
Eric Duceau	Airbus, France
Didier Dumur	CentraleSupelec, France
Gauthier Fanmuy	Dassault Systèmes, France
Pascal Foix	Thales, France
Alan Guegan	Sirehna, France
Omar Hammami	Ensta ParisTech, France
Fabien Mangeant	Renault, France
Luca Palladino	Safran, France
Pascal Poisson	Alstom Transport, France
Alain Roset	La Poste, France
Alain Roussel	AFIS, France
Richard Schomberg	EDF, France

Invited Speakers

Plenary sessions

Come Berbain	Chief Technical Officer of the French State, DINSIC
Manfred Broy	Professor, Technical University of Munchen
Yves Caseau	Group Chief Information Officer, Michelin
Pierre Chanal	VP Engineering Transformation, Alstom
Vincent Danos	Research Director, CNRS
Marc Fontaine	Head of Digital Transformation, Airbus
Hervé Gilibert	Chief Technical Officer and Quality, ArianeGroup

“Methods and Tools” Track

Martin Neff	Chief Architect Systems Engineering, Audi
Marie Capron	Engagement Manager and System Engineering, Sogeti High Tech

“Design, Manufacture and Operation of Complex Products and Services” Track

Yann Bouju	Project Manager, Virtual and Augmented Reality, Naval Group
Olivier Flous	VP Digital Transformation, Thales Digital Factory

“Aeronautics” Track

Thierry Chevalier	Chief Engineer Digital Design and Manufacturing, Airbus

“Energy” Track

Yannick Jacquemard	R&D Director, RTE
Isabelle Moretti	Chief Scientific Officer, Engie
Guillaume Brecq	Product Owner, Engie

“Healthcare Services” Track

Philippe Baron	Chief Executive Officer, AxDaNe
Fabrice Lejay	R&D Product Line Manager, Stago

“Transportation & Mobility” Track

Brigitte Courtehoux	Executive Vice President Head of PSA Group New Mobility, PSA

Acknowledgements

We would like to thank all members of the Program and Organizing Committees for their time, effort, and contributions to make CSD&M Paris 2018 a top-quality conference. Special thanks go to the CESAM Community team who permanently and efficiently managed all the administration, logistics, and communication of the CSD&M Paris 2018 conference (see http://cesam.community/en).

The organizers of the conference are also grateful to the following partners without whom the CSD&M Paris 2018 event would not exist:

- **Founding partners**
  - CESAM Community managed by the Center of Excellence on Systems Architecture, Management, Economy & Strategy (CESAMES),
  - Association Française d'Ingénierie Système (AFIS),
  - The Ecole Polytechnique – ENSTA ParisTech – Télécom ParisTech – Dassault Aviation – Naval Group – DGA – Thales "Engineering of Complex Systems" chair.
- **Industrial and institutional partners**
  - Airbus Group,
  - Alstom Transport,
  - ArianeGroup,
  - INCOSE,
  - MEGA International,
  - Renault,
  - Thales.
- **Participating engineering and software tools companies**
  - Airbus Apsys,
  - Aras,
  - Dassault Systèmes,
  - Easis Consulting,

- Esteco,
- Hexagon PPM,
- Intempora,
- No Magic Europe,
- Obeo,
- Persistent Systems,
- SE Training,
- Siemens,
- Sodius.

Contents

Regular Papers

Formal Methods in Systems Integration: Deployment of Formal Techniques in INSPEX

Richard Banach[1](✉), Joe Razavi[1], Suzanne Lesecq[2], Olivier Debicki[2], Nicolas Mareau[2], Julie Foucault[2], Marc Correvon[3], and Gabriela Dudnik[3]

[1] School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK
{richard.banach,joseph.razavi}@manchester.ac.uk

[2] CEA, LETI, Minatec Campus, 17 Rue des Martyrs, 38054 Grenoble Cedex, France
{suzanne.lesecq,olivier.debicki,nicolas.mareau,julie.foucault}@cea.fr

[3] CSEM SA, 2002 Neuchatel, Switzerland
{marc.correvon,gabriela.dudnik}@csem.ch

Abstract. Inspired by the abilities of contemporary autonomous vehicles to navigate with a high degree of effectiveness, the INSPEX Project aims to create a minaturised smart obstacle detection system, which could find use in a wide variety of leading edge smart applications. The primary use case focused on in the project is producing an advanced prototype for a device which can be attached to a visually impaired or blind (VIB) person's white cane, and which, through the integration of a variety of minaturised sensors, and of the processing of their data via sophisticated algorithms, can offer the VIB user greater precision of information about their environment. The increasing complexity of such systems creates increasing challenges to assure their correct operation, inviting the introduction of formal techniques to aid in maximising system dependability. However, the major challenge to building such systems resides at the hardware end of the development. This impedes the routine application of top-down formal methods approaches. Some ingenuity must be brought to bear, in order that normally mutually hostile formal and mainstream approaches can contribute positively towards system dependability, rather than conflicting unproductively. This aspect is illustrated using two strands of the INSPEX Project.

1 Introduction

The contemporary hardware scene is driven, to a large extent, by the desire to make devices smaller and of lower power consumption. Not only does this save materials and energy, but given the commercial pull to make mobile phones increasingly capable, when small low power devices are incorporated into mobile phones, it vastly increases the market for them. The smartphone of today is unrecognisable (in terms of the facilities it offers) from phones even as little as a decade old. This phenomenon results from ever greater advances in system

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 3–15, 2019.
https://doi.org/10.1007/978-3-030-04209-7_1

structure, and from the trend to incorporate minaturised sensing technologies that were well beyond the state of the art a short while ago. This trend continues unabated, and also massively propels advances in the Internet of Things.

The availability of such minaturised devices inspires the imagination to conceive novel applications, previously unrealised due to technological barriers. The INSPEX Project is the fruit of one such exercise in imagineering. Taking the autonomous vehicle [15] as inspiration, along with the data fusion that enables autonomous vehicles to elicit enough information about their environment from the data gathered by a multitude of sensors to navigate sufficiently safely that autonomous vehicles ‘in the wild’ are forseen within a few years [28,35], INSPEX aims to minaturise a similar family of sensors to create a device that offers comparable navigational support to a wide variety of smaller, more lightweight applications.

In the remainder of this paper we do the following. In Sect. 2 we cover the potential application areas for INSPEX, pointing to the key VIB use case that forms the focus of the project. In Sect. 3 we focus more narrowly on the technical elements of the VIB use case. In Sect. 4 we address ourselves to the deployment of formal modelling and verification technologies within the INSPEX development activity. We focus on two areas within which formal techniques were deployed in INSPEX, namely in the power management design and in the data acquisition pathway. Section 5 contains discussion and concludes.

2 INSPEX Application Use Cases

Figure 1 gives an indication of the range of applications that the INSPEX imagineering effort generated. The figure is divided into four broad aplication areas. Working left to right, we start with some examples of small autonomous vehicles. Autonomous navigation for these demands the small size, weight and power requirements that INSPEX seeks to provide. Small airborne drones have demands that are very similar, and as their number increases, their navigation and collision avoidance needs increase correspondingly. Considerations of size, weight and power also impinge on humanoid robots and specialised devices such a floor cleaning robots. INSPEX navigation capabilities will also increase autonomy and flexibility of use for factory based transport robots, which have to be prepared to avoid unexpected obstacles, unless their environment is sufficiently tightly constrained.

At the bottom of Fig. 1 we see some examples concerned with large enclosed environments, such as highly automated factories featuring assembly lines consisting of hundreds of robots. To increase the flexibility of reconfiguration of these, increased autonomy in the participating robots is one necessary ingredient. INSPEX, appropriately deployed, can significantly assist in meeting this requirement. The issue becomes the more forceful when the robots involved are

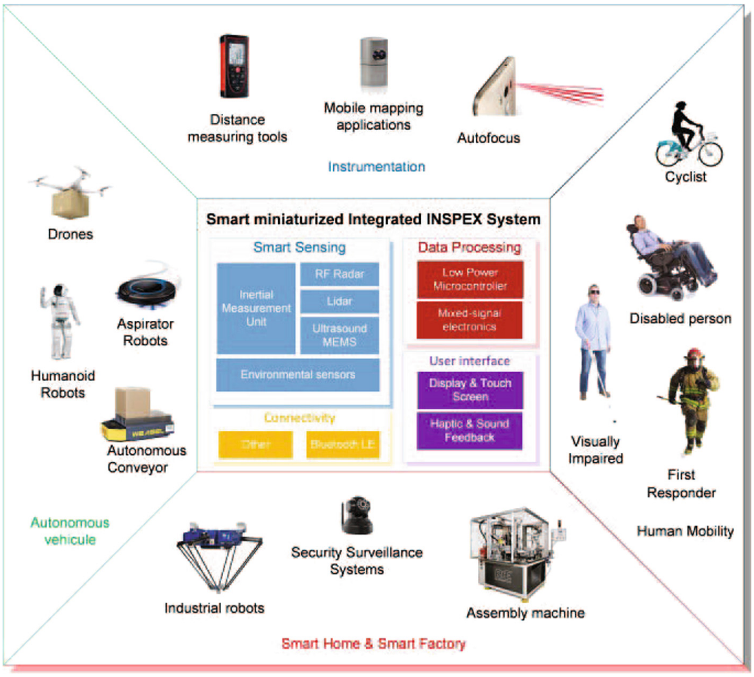


Fig. 1. Potential INSPEX use cases.

mobile, since along with the need to be more smart, they particularly need to avoid harm to any humans who may be working nearby. Security surveillance systems, traditionally relying on infra-red sensors, can also benefit from the extra precision of INSPEX.

At the top of Fig. 1 we see some examples concerned with distance estimation. Modern distance measuring tools typically make use of a single laser beam whose reflection is processed to derive the numerical result. For surfaces other than smooth hard ones, the measurement arrived at may be imprecise, for various reasons. INSPEX can perform better in such situations by combining readings from a number of sensors. A very familiar application area for such ideas is autofocus in cameras. These days, camera systems (typically in leading edge phones) employ increasingly sophisticated algorithms to distinguish foreground from background, to make up for varying lighting conditions, and generally to compensate for the user's lack of expertise in photography. INSPEX can add to the capabilities available to such systems.

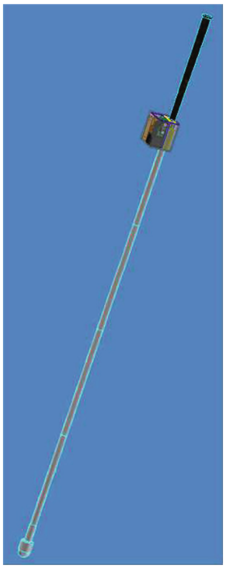


Fig. 2. The INSPEX white cane addon.

On the right of Fig. 1 we see the use cases for human centred applications. We see the VIB use case which forms the focus of the INSPEX project, and which will be discussed in detail later. There are also other prominent use cases. The first responder example includes cases like firefighters, who need to be able to enter hazardous environments such as smoke filled rooms, in which normal visibility is impossible. An aid like an INSPEX device can be of immeasurable help, in giving its users some orientation about the space in which they find themselves, without resorting to tentative feeling about, which is what firefighters are often reduced to. Other applications include the severely disabled who may have impediments to absorbing the visual information from their surroundings. And the able bodied too can benefit from INSPEX, when visibility is severely reduced. Although the cases of heavy fogs which reduced visibility to almost zero are thankfully history, today's mega-cities now feature smogs due to different sources of atmospheric pollution which can be just as bad.

3 The INSPEX VIB White Cane Use Case

Although a large number of use cases are envisaged for a system such as INSPEX, the primary use case addressed within the INSPEX Project is the smart white cane to assist visually impaired and blind persons. Figure 2 shows a schematic of one possible configuration for the attachment of a smart addon to a standard type of white cane. The white cane application needs other devices to support the white cane addon, in order that a system usable by the VIB community ensues. Figure 3 shows the overall system architecture.

As well as the Mobile Detection Device addon to the white cane, there is an Audio Headset containing extra-auricular binaural speakers and an inertial measurement unit (IMU)—the latter so that an audio image correctly oriented with respect to 3D space may be projected to the user, despite the user's head movements. Another vital component of the system is a smartphone. This correlates the information

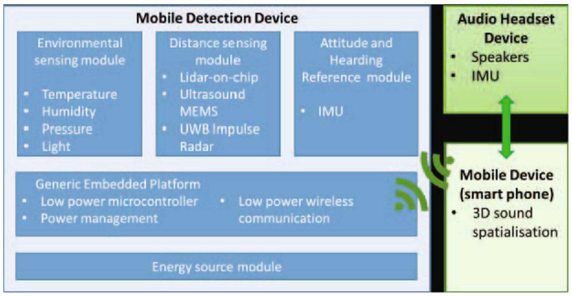


Fig. 3. The architecture of the INSPEX system.

obtained by the mobile detection device with what is required by the headset. It also is able, in *smart city* environments, to receive information from *wireless beacons* which appropriately equipped users can access. This enables the whole system to be even more informative for its users.

The white cane add-on contains the sensors that generate the data needed to create the information that is needed by the user. The chief among these comprise a short range LiDAR, a long range LiDAR, a wideband RADAR, and a MEMS ultrasound sensor. Besides these there are the support services that they need, namely an Energy Source Unit, environmental sensors for ambient light, temperature and humidity, another IMU and a Generic Embedded Platform (GEP).

The main sensors are subject to significant development and minaturisation by a number of partners in the INSPEX project. The short range LiDAR is developed by the Swiss Center for Electronics and Microtechnology (CSEM) and the French Alternative Energies and Atomic Energy Commission (CEA). The long range LiDAR is developed by the Tyndall National Institute Cork and SensL Technologies, while the wideband RADAR is also developed by CEA. The MEMS ultrasound sensor is from STMicroelectronics (STM). Cork Institute of Technology (CIT) design the containing enclosure and support services, while the audio headset is designed by French SME GoSense.

The GEP has a noteworthy challenge to confront. Data from the sensors comes in at various times, and with varying reliability. Distance measurements from the sensors are just that, merely distance data without any notion of direction, or orientation with respect to the user. The latter is elucidated by reference to data from the IMU in the mobile detection device. Data from both the IMU and directional sensors is timestamped, since freshness of data is crucial in providing information to the user that is not only accurate but timely. This enables distance sensor data to be aggregated by time and IMU data.

Once the data has been correctly aggregated, it is passed to the module in the GEP that computes the *occupation grid*. This is a partition of the 3D space in front of the user into cells, each of which is assigned a probability of its being occupied by some obstacle. The occupation grid idea is classical from the autonomous vehicle domain, but in its standard implementation, involves intensive floating point computation [28,35]. This is too onerous for the kind of lightweight applications envisaged by the concept of INSPEX. Fortunately INSPEX is able to benefit from a highly efficient implementation of the occupation grid, due to a careful analysis of the computations that are needed to derive a good occupation grid result [13]. The integration of all the hardware and software activities described, constitutes a non-trivial complex systems undertaking.

The wide range of sensors and their concomitant capabilities in the INSPEX white cane application is necessitated by the detailed needs of VIB persons navigating around the outdoors environment (in particular). Although a standard white cane can give good feedback to its user regarding the quality and characteristics of the ground in front of them, especially when the ground texture in the urban environment is deliberately engineered to exhibit a range of standard textures signifying specific structures [31], it gives no information about hazards to be found higher up. It is a fact of life for VIB persons, that, like it or not, unanticipated collisions with obstacles at chest or head height are an unavoidable occurrence [25]. Many VIB persons are prone to wearing some sort of headgear, more or less involuntarily, to try to mitigate the worst effects of such unanticipated high level collisions. The possibility of alleviating this situation, even in the absence of other use cases, makes for ample justification for the development of INSPEX.

4 Formal Modelling and Verification in INSPEX

By now, formal techniques of system development have had a substantial history. After the early years, and the widespread perception that such approaches were 'hard' and did not scale, there was a concerted effort to dispel this view in classic works such as [11,12,19]. It was increasingly recognised, especially in niche areas, that formal techniques, wisely deployed, can add a measure of dependability not achievable by other means.[1] It became recognised that tools, particularly ones that worked in a reasonably scalable way,[2] were key to this [33,34]. This spurred the idea of 'Grand Challenges' in verification, one purpose of which was to both test and further inspire the scalability of tools [23,38,39]. Later surveys include [3,8], and this trend is also evident in [5].

The classic way of applying formal approaches is top-down. One starts with an oversimplified, but completely precise, definition of the desired system. This is then enriched, via a process of formal refinement, to take into account more system detail in order to address more of the system's requirements. Eventually one gets close enough to the code level that writing code is almost a transcription, or the code can be generated automatically.

There are many variations, small and large, on this basic idea. An early account is in [30]. The Z approach is represented by [21,32]; the VDM apporach is in [17,22]; TLA+ is in [24]; Alloy in [1]. There are many others. The B-Method, of which more later, is represented by [2,4,29].

Accompanying these developments grew the subdiscipline of behaviour oriented, or process oriented descriptions of system behaviour. Early references are [7,20,26]. Not long afterwards, it was observed that many process oriented properties of interest for systems conformed to a so-called model checking pattern, and this led to an explosion of research and tool building, since model checking could then be completely automated, leading to tools that could work in a push-button manner, and that could be embedded in development environments, in which they worked 'behind the scenes', i.e. without explicit user control or invocation. Among the tools in this style that have proved to be of interest for the INSPEX project are FDR [16], NuSMV [27], Uppaal [36].

[1] In some niche areas, the recognition came as a direct result of painful and expensive failure, the Pentium Bug and Arianne Disaster being iconic examples.

[2] It became apparent at this time that scalable formal tools were not an impossible dream, even if the degree of scalability was not as great as typically found in conventional approaches.

Whereas all the preceding approaches relied on there being a model of the system that was presented in a relatively abstract language, the growing power and scalability of tools generated an interest in techniques that worked directly on implementation level code. By now there are many well established tools that input an implementation in a given language such as C or C++, and that take this implementation and then analyse it directly for correctness properties [37]. Very often these properties are predefined runtime correctness properties concerning commonly introduced programmer errors, such as (the absence of) division by zero or (the absence of) null pointer dereference. Some however, e.g. [6,9] allow more application specific properties to be checked.

While direct checking of implementations would appear to be a panacea for verification, it nevertheless risks overemphasising low level system properties at the expense of the higher level view. When we recognise that deciding what the system *should be* is always a human level responsibility, and that formal approaches can only police the consistency between different descriptions of the same thing, abandoning the obligation to independently consider the abstract high level view of the system risks abandoning a valuable source of corroboration of the requirements that the system is intended to address. It is this kind of 'stereoscopic vision' on what the system ought to do and to be that constitutes the most valuable contribution that a top-down formal approach makes to system development, quite aside from the formal consistency checking.

In normal software developments, one starts the process with a good idea of the capabilities of software in general, so in principle, it is feasible to use a relatively pure top-down approach. Likewise in most hardware developments that take place at the chip level, one starts the process with a good idea of the capabilities of the technology platform that will be used, and working top-down is perfectly feasible (and in fact is unavoidable given the scale of today's chips). In both of these cases deploying top-down formal techniques (if the choice is made to do so) is feasible.

In INSPEX however, the development of the devices at the physical level is a key element of ongoing project activity, and the low level properties of all the devices used in the INSPEX deliverable are contingent and emergent to a significant extent. This makes the naive use of top-down approaches problematic, since there is no guarantee that the low level model that emerges from a top-down development process will be drafted in terms of low level properties that are actually reflected in the devices available, since the constraints on the system's behaviour that are directly attributable to physics are simply incontestable. As a result of this, the approach to incorporating formal techniques in INSPEX was a hybrid one. Top-down and bottom-up approaches were pursued concurrently, with the aim of meeting in the middle.

The next sections cover how this hybrid approach was applied in two of the INSPEX Project's activities, namely the design of the power management strategy for the mobile detection device module, and in the verification of the data pathway from the sensors to the data fusion application.

4.1 Power Management Formal Modelling and Verification

In INSPEX, power management poses a number of challenges. As stated earlier, the concentration of effort in INSPEX is on engineering a suitable outcome at the hardware systems level. Each sensor and subsystem creates its own problems. However they all share a common goal, one common to all mobile systems, of making the smallest demand on the power system that is possible. However, a focus on individual submodules risks paying insufficient attention to issues of coordination. A higher level view offers a number of benefits.

The first benefit is an issue of correct functioning. A naive combination of low level modules, each of them correct in itself, is not guaranteed to generate in a straightforward manner (from a systems level perspective), a globally correct behaviour. For example a submodule might conceivably be left running when it ought not to be running as an unexpected consequence of some complex sequence of events. The second benefit is the issue of global optimality. Focusing on the low level prevents the global optimisation of performance (in this case power saving) by balancing criteria from competing interests originating in diverse submodules.

A formal approach rooted in a higher level view can assist in both of these aspects of the development. Formal techniques are suited *sans pareil* to targeting correctness aspects of a development. Moreover, they are capable of capturing the global consequences of a collection of submodels when they are combined into a single entity, since they do not suffer from the variability of focus that humans can exhibit when they concentrate on one or another aspect of an activity.

Power management design in INSPEX proceeded top-down. From a human perspective this might mean considering broad properties of the power regime first, descending to low level detail at the end — this would fly in the face what has been stated above since what is most incontestable about the design is the low level properties of individual sensors etc. We reconcile these views by observing that formally, 'top level' properties are those that will not be contradicted in subsequent steps of development. This implies that they will be the most primitive rather than the most far reaching among the properties that the system satisfies.

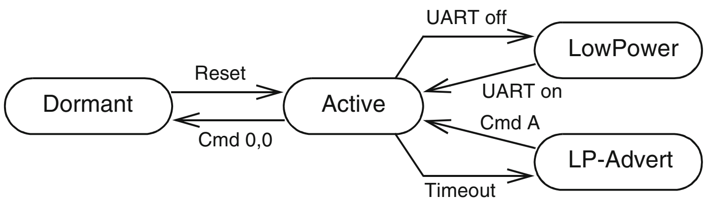


Fig. 4. A simplified Bluetooth transition diagram.

The most primitive properties include the state transition diagrams of the various sensors and other components. Figure 4 gives an example of a transition diagram for the Bluetooth submodule, rather drastically simplified from the description in [10]. To incorporate this into a wideranging formal model we used the Event-B formalism [4]. This enables many levels of abstraction to be formally related to each other via refinement, and is supported by the Rodin tool which features many provers and plugins [29]. A state transition diagram such as Fig. 4 can be captured in Event-B in a fragment like:[3]

```
EVENTS
  Dor2Act
    WHEN  state = Dormant ∧ Reset
    THEN
      state := Active
    END
  LP2Act
    WHEN  state = LowPower ∧ UART_on
    THEN
      state := Active
    END
  LPA2Act
    WHEN  state = LP_Advert ∧ Cmd_A
    THEN
      state := Active
    END
  ...  ...
  Act2Dor
    WHEN  state = Active ∧ Cmd_0, 0
    THEN
      state := Dormant
    END
  Act2LP
    WHEN  state = Active ∧ UART_off
    THEN
      state := LowPower
    END
  Act2LPA
    WHEN  state = Active ∧ Timeout
    THEN
      state := LP_Advert
    END
```

A formal model such as the fragment above relates to the low level real time software and firmware as follows. Each event in the model corresponds to a software or firmware command, or an interrupt routine. The guard portion, expressed in the WHEN clause of the event, corresponds to the entry condition code in the command, or scheduler code that checks the cause of the interrupt. The event's THEN clause corresponds to the software command body, or the interrupt handler routine. As stated earlier, capturing all the commands and sources of interrupt enables questions of overall consistency to be examined.

Once the low level integrity has been established, other considerations can be brought to bear. A major element is the quantitative aspect. Event descriptions as above are embellished with numerical data regarding the energetic consequences of executing the event, enabling overall conclusions about energy consumptions to be drawn. Finally, considerations of overall power management policy can be layered onto the formal model and made to correspond with the implementation code.

4.2 The Data Acquisition Pathway

Another major area in which formal techniques were deployed in INSPEX to add robustness to the software design was the data acquisition pathway. As

[3] For reasons of the confidentiality of the future commercial exploitation of the INSPEX platform, what is shown here is not actual code.

outlined earlier, in INSPEX, there are several sensors, each working to different characteristics, but all contributing to the resolution of the spatial orientation challenge that is the *raison d'être* of INSPEX.

The various INSPEX sensors work at frequencies that individually can vary by many orders of magnitude. For example, the LiDARs can produce data frames with extreme rapidity, whereas the ultrasound sensor is limited by the propagation speed of pressure waves through the air, which is massively slower than the propagation characteristics of electromagnetic radiation. The ultrasound sensor, in turn, can produce data frames much faster than typical human users are able to re-orient their white canes, let alone move themselves to a significant degree, either of which requires a fresh occupation grid to be computed. This means that the data integration performed by INSPEX has to be harmonised to the pace of the human user.

The main vehicle for achieving this is the IMU. The IMU is configured to supply readings about the orientation of the INSPEX mobile detection device addon at a rate commensurate with the needs of human movement. This 'pacemaker rate' is then used to solicit measurements from the other sensors in a way that not only respects their response times but is also staggered sufficiently within an individual IMU 'window' that the energy demands of the individual measurements are not suboptimal with respect to the power management policy currently in force.

The above indicates a complex set of information receipt tasks, made the more challenging by the fact that all the sensors speak to the same receiving hardware. The goal of the information receipt tasks is to harvest a collection of data frames from the individual sensors, each timestamped by its time of measurement, and each related to a before- IMU data frame, and an after- IMU data frame, each itself timestamped. The two successive IMU data frames, and way their data might differ due to user movement, enable the interpolation of orientation of the distances delivered at various times by the other sensors.

Timing is evidently of critical importance in the management of the incoming data. This notwithstanding, all the tasks that handle these information management duties are executed at the behest of the generic embedded device's low level scheduler. The scheduler used belongs to the real time operating system employed in the GEP, which is a version of FreeRTOS [18].

Turning to the formal modelling of what has just been described, it may well seem that the complexity of the situation might defeat efforts to add useful verification to the design. The situation is helped considerably by the existence of a formal model of the FreeRTOS scheduler [14]. This is in the kind of state oriented model based form that can be made use of in the modelling and verification of the data acquisition pathway in INSPEX. Accordingly, the properties of the FreeRTOS scheduler derived in [14] can be translated into the Event-B framework used for INSPEX and then fed in as axioms in the Event-B models that contribute to the INSPEX data acquisition pathway verification.

Within this context, the rather delicate modelling of timing issues indicated above can be based on a sensible foundation. The complexities of the behaviour of the INSPEX data acquisition pathway imply that relatively straightforward models of time, as typically included in timed tools, are not sufficiently incisive to capture the potentially problematic aspects of the system, so a bespoke approach to the modelling of time within Event-B is needed, and this consequently drives the structure of the verification activity.

5 Discussion and Conclusions

In the preceding sections, we introduced the INSPEX Project and its intended use cases, before homing in on the VIB white cane add-on use case which forms the focus of the project itself. The main objective of this paper was to describe the use of formal techniques within INSPEX, to which we addressed ourselves in Sect. 4. This contained a summary of the deployment of formal techniques in the data acquisition pathway and the power management design.

Given the practical constraints of the project, it was impossible to follow a pristine top-down formal development approach in combining formal and more mainstream techniques. Given that the two approaches were being pursued concurrently, one of the greatest challenges that arises is to keep both activities in step. Little purpose is served by verifying the correctness of a design that has been superseded and contradicted in some significant aspect. The formal activity therefore paid significant attention to checking whether the growing implementation continued to remain in line with what had previously been formally modelled and verified. This way of working contributed the greatest element of novelty to the combined use of formal and conventional techniques in the project, and constitutes a stimulus for finding further novel way of reaping the benefits of fusing the two approaches.

Acknowledgement. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 730953. The work was also supported in part by the Swiss Secretariat for Education, Research and Innovation (SERI) under Grant 16.0136 730953. We thank them for their support.

References

1. Alloy. http://alloy.mit.edu/
2. Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (1996)
3. Abrial, J.R.: Formal Methods in Industry: Achievements. Problems Future. In: Proceedings of ACM/IEEE ICSE 2006, pp. 761–768 (2006)
4. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. CUP (2010)
5. Andronick, J., Jeffery, R., Klein, G., Kolanski, R., Staples, M., Zhang, H., Zhu, L.: Large-scale formal verification in practice: a process perspective. In: Proceedings of ACM/IEEE ICSE 2012, pp. 374–393 (2012)
6. Astrée Tool. http://www.astree.ens.fr/

7. Baeten, J.: Process Algebra. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (1990)
8. Banach, R. (ed.): Special Issue on the State of the Art in Formal Methods. Journal of Universal Computer Science, vol. 13(5) (2007)
9. BLAST Tool. https://forge.ispras.ru/projects/blast/
10. Bluetooth Guide. http://ww1.microchip.com/downloads/en/DeviceDoc/50002466B.pdf
11. Bowen, J., Hinchey, M.: Seven more myths of formal methods. IEEE Software **12**, 34–41 (1995)
12. Clarke, E., Wing, J.: Formal methods: state of the art and future directions. ACM Comput. Surv. **28**, 626–643 (1996)
13. Dia, R., Mottin, J., Rakotavao, T., Puschini, D., Lesecq, S.: Evaluation of occupancy grid resolution through a novel approach for inverse sensor modeling. In: Proceedings of IFAC World Congress, FAC-PapersOnLine, vol. 50, pp. 13,841–13,847 (2017)
14. Divakaran, S., D'Souza, D., Kushwah, A., Sampath, P., Sridhar, N., Woodcock, J.: Refinement-based verification of the FreeRTOS scheduler in VCC. In: Butler, M., Conchon, S., Zaïdi, F. (eds.) Proceedings of ICFEM 2015. LNCS, vol. 9407, pp. 170–186. Springer (2015)
15. Fausten, M.: Evolution or revolution: architecture of AD cars. In: Proceedings of IEEE ESWEEK (2015)
16. FDR Tool. https://www.cs.ox.ac.uk/projects/fdr/
17. Fitzgerald, J., Gorm Larsen, P.: Modelling Systems: Practical Tools and Techniques for Software Development. Cambridge University Press (1998)
18. FreeRTOS. https://www.freertos.org/
19. Hall, A.: Seven myths of formal methods. IEEE Software **7**, 11–19 (1990)
20. Hoare, C.: Communicating Sequential Processes. Prentice-Hall (1985)
21. ISO/IEC 13568: Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics: International Standard (2002). http://www.iso.org/iso/en/ittf/PubliclyAvailableStandards/c021573_ISO_IEC_13568_2002(E).zip
22. Jones, C.: Systematic Software Development Using VDM, 2nd edn. Prentice-Hall (1990)
23. Jones, C., O'Hearne, P., Woodcock, J.: Verified software: a grand challenge. IEEE Comput. **39**(4), 93–95 (2006)
24. Lamport, L.: Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley (2002)
25. Mandruchi, R., Kurniavan, S.: Mobility-Related Accidents Experienced by People with Visual Impairment. Insight: Research and Practice in Visual Impairment and Blindness (2011)
26. Milner, R.: Communication and Concurrency. Prentice-Hall (1989)
27. NuSMV Tool. http://nusmv.fbk.eu/
28. Qu, Z.: Cooperative Control of Dynamical Systems: Applications to Autonomous Vehicles. Springer (2009)
29. RODIN Tool. http://www.event-b.org/, http://sourceforge.net/projects/rodin-b-sharp/
30. de Roever, W.P., Engelhardt, K.: Data Refinement: Model-Oriented Proof Methods and their Comparison. Cambridge University Press (1998)
31. Rosburg, T.: Tactile ground surface indicators in public places. In: Grunwald, M. (ed.) Human Haptic Perception: Basics and Applications. Springer, Birkhauser (2008)

32. Spivey, J.: The Z Notation: A Reference Manual, 2nd edn. Prentice-Hall International (1992)
33. Stepney, S.: New Horizons in Formal Methods. The Computer Bulletin, pp. 24–26 (2001)
34. Stepney, S., Cooper, D.: Formal Methods for Industrial Products. In: Proceedings of 1st Conference of B and Z Users. LNCS, vol. 1878, pp. 374–393. Springer (2000)
35. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
36. UPPAAL Tool. http://www.uppaal.org/
37. Wikipedia: List of tools for static code analysis. https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
38. Woodcock, J.: First steps in the the verified software grand challenge. IEEE Computer **39**(10), 57–64 (2006)
39. Woodcock, J., Banach, R.: The verification grand challenge. JUCS **13**, 661–668 (2007)

Ontology-Based Optimization for Systems Engineering

Dominique Ernadote(✉)

Airbus Defence and Space, Elancourt, France
dominique.ernadote@airbus.com

Abstract. Model-Based Systems Engineering techniques used with decriptive metamodel such as NAF, SysML or UML often fails to provide quick analyses of huge problem spaces. This is generally compensated by Operations Research technique supporting the resolution of constraint-based problems. This paper shows how both perspectives can be combined in a smooth continuous bridge filling the gap between the two universes whilst hiding the operations researchs complexity for the modelers and automating the exploration of a very huge problem space for the finding of optimized solutions.

Keywords: MBSE · Model-Based Systems Engineering
Systems engineering · Operations research

1 Introduction

System engineering is a multi-domain process that encompasses the design, realisation, delivery, and management of complex systems or system of systems. Best-practices that ensure the quality of such processes has been documented in standards, such as ISO 15288 [15], and system engineering resources, such as the INCOSE handbook [2,12]. These standards assess and describe the different activities of the system engineering process, detail the involved stakeholders and their responsibilities with respect to these activities, and specify the required and the produced deliverables. These descriptions are highly useful, but they mainly remain document centric. Meanwhile, a Model-Based System Engineering (MBSE) approach is commonly accepted by the system engineers community [17] that depends up on the creation of centralized models to produce the expected deliverables. Standard metamodels such as UML [21], SysML [20] or NMM/NAF [18] are typically used to describe the relevant concepts for these descriptive models.

A concrete implementation of an MBSE approach implies that system engineers know the corresponding metamodels. Such metamodels are typically well understood within technical domains, for example software development based on UML, or database development using conceptual data models and schemas. However, if one considers the stakeholders involved in the entire system engineering process, he has to include the V&V stakeholders, the stakeholders involved in the system rollout, the end-users... each in a specific domain with its own vocabulary. Even if there are some commonalities, each specialist integrates specific terms due to the type of systems designed. With those sharing constraints, the usage of a predefined metamodel creates barriers between the modeling experts and the stakeholders responsible for other activities. Thus, an efficient collaboration requires the adoption of the model presentation to this later audience via domain specific languages (DSLs) also named ontologies (see [13] for a discussion of the differences between DSL, ontology, and metamodel).

The author proposed an approach in [6] to reconcile the usage of complex but necessary metamodels with dedicated ontologies which are more friendly for the end users. Briefly, the method is based on the mapping of the user-defined ontology against a modeling tool metamodel. This added layer eases the creation of models by different communities of users whilst ensuring the storage mode is valid against a selected standard metamodel which is of interest when the modeling tools interoperability becomes a concern.

However, all these mechanims rely on common desciptive languages which means the models are first depicted in the selected language by creating one by one the modeling elements, setting their property values and linking them each other. Then the models are assessed against engineering criteria, stakehoders needs, or system requirements. Then, the system engineer applies such or such analysis technique on 2, 3, or more model alternatives. The more models are studied, the more confident the engineer is regarding her/his final choice.

The main drawback of this way of doing resides in the time consuming effort to produce the models. According to [9], the current complexity increase in system design is posing dramatic increasing challenges for a system engineering under time and resources constraints. System architecture designs have been proved to be NP-Hard [10] and the direct consequence is that the system architecture design by itself should be considered as an optimization problem. This non-linear complexity increase accompanied with few changes regarding the resource availability enforce to seek for new means and specifically to be machinely helped during the optimal system discovery process.

Still focusing on the engineering productivity, this raises the following questions: Is there a way to handle the variability of a system such a way that the modeling activities are effortless, or at least not linear against the number of system solutions to be compared? Extending again the expectations, is it possible to declare the constraints from the stakeholder perspectives and then to automatically produce a model that fulfills all the constraints and is proved to be one of the best solutions? These are the questions addressed in this paper.

2 Prerequesites for an Automated System Design Optimization

This paper presents the optimization of descriptive models in two parts. First, there is a step of description of the model variability which defines the solutions space of the expected models. The notion of variability includes information on both the individual modeling elements and the sets of those elements. The Sect. 3 addresses this topic.

Secondly, the constraining expression of the model variability are used to find out an optimal solution. An appropriate algorithm must be found to efficiently find the solution, and the impact of an existing model have to be considered so that the solution is stored in the model repository. This topic is addressed in the Sect. 4.

2.1 The Modeling Planning Process

The Modeling Planning Process (MPP) is a modeling method detailed in [6] which aims to objectively tailor the system engineering modeling activities. This is not the subject of this paper to explain the method but the presented optimization of descriptive models presented here relies on it so its main phases are briefly summarized as follows:

1 - Identify and prioritize the modeling objectives. By identifying the modeling objectives we except to find out the rationale for the modeling activities. The question to answer is “what are the expected outcomes of the models?”

2 - Determine the relevant concepts to be modeled which help achieving the modeling objectives. Doing so, a conceptual data model emerges which must be expressed in the end-users’ terms including the relations between the concepts and their properties. At this level, there is no technical standards involved.

3 - Map the identified concepts to a selected modeling standard. Even though a dedicated ontology could be used to implement the modeling tool aligned with the designed data model, we recommend to rely on existing standards such as NAF, SysML or UML. The rationale for this recommendations is to ensure the lifespan of the data and the ability to read and write the information in different modeling tools (data interoperability).

4 - Define the contents of the expected deliverables. The modeling activities achievement is concretized thanks to human-readable documents. This is still a mandatory step for the final validation of the design. The contents of such documents can be expressed by referring to the end-user concepts, the relation, and the properties.

Since all these phases require traceability from the objectives to the final deliverables, the method is itself implemented by modeling the involved notions which allows a quick automation of the deliverable production (step MMP-4). In the following sections, we name MB^{2}SE this modeling framework because the model-based systems engineering is itself model-based.

3 Model Variability

Variability modeling approaches define different ways to express the space of freedom for a model. It is in this space that an optimal solution is searched for.

According to [3], there are two kinds of variability modeling approaches: feature modeling (FM), and decision modeling (DM). Feature modeling captures features defined as capabilities of systems having a specific interest for the end-users. On the other hand, a decision model focuses on decisions trading-off the different branches of a variation model. [3] compares different approaches including both the two modeling types and concludes there are mainly similar. In our context, the models based on SysML or NAF do not integrate by themselves the notion of decision. This should be added but for standard compatibility reason which is often a customer requirements in the defense sector, we have to reduce the metamodel extension. This is why we will consider the feature modeling in more detail.

Regardless the modeling mode, there are one of the compared characteristics studied in [3]. *Orthogonality*: the degree to which the variability is separated from the system model. This is for us an important concern in order to ensure the constraints elicitation can be listed by family and then applied to an independent specific system model. *Data Types*: types refer to the primitive and composite values which opens the solution space. Still according to [3], DM and FM are similar. *Modularity*: Ability to reuse variability expression in order to handle the variability model complexity.

One conclusion of this comparison paper, is that the Common Variability Language (CVL) [11], which is a variability modeling language, presents most of the expected advantages including orthogonality, and a large expressiveness. Another advantage is the standardization of this language since the Object Management Group organization (OMG) is under the process of integrating the language proposal. This should be made easier by the fact it is a sublanguage of OCL (Object Constraint Language) [19,22,24], an already standardized constraint language for UML. By the way, the *compositional layer* of CVL provides ways to encapsulate and reuse variability specifications through *Configurable Units* and *Variability Interfaces* [4, Sect. 2.5].

3.1 Link with the MB^{2}SE Conceptual Data Model

OCL-like constraint language have been developed to specify contraints on UML model. Thus, such language integrates a grammar which takes advantages from the classifier characteristics: attributes, and association ends. These languages usually encompass both characteristics under the term *property.* Figure 1 shows an example of UML class diagram illustrated in [22]. Through this example, constraints can be written in the OCL language which uses the dot notation to access the different properties either directly or via a path. As an example, the constraint "`Person  self.age > 0`" indicates that the attribute `age` of a class `Person` shall be greater than 0. The constraint:

```
   RentalStation self.employee->forAll(p |
p.income(‘‘98/03/01’’) > 2000)
```

includes a reference to the association end `employee` and the function `income()`. The dot notation can be used to concatenate association ends via the notion of Set [19, Sect. 7.5.3, p. 18].

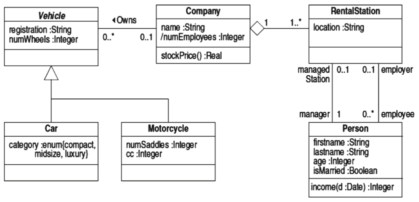


Fig. 1. UML class diagram.

This kind of notation suits perfectly the MB^{2}SE framework; since the modeling objectives are mapped to a conceptual data model (step MMP-2) of the MPP method, the notions of classes, attributes, and association ends are also present. The MB^{2}SE foundation uses a very small subset of the class diagram capabilities (for example, no functions are described) but at least the important property notions are there and should be integrated into the constraint language finally selected to support the design process. By using a model to describe the conceptual data model allows linking the corresponding modeling elements (class, attribute, and association ends) to the constraints. Thus syntax checking are possible by analyzing the data model for example to check a path is valid.

4 Fulfilling the Modeling Constraints

Once the variability modeling has been expressed, it is time to exploit it in order to find out 1 or several optimal solutions to the design problem. As exposed in the introduction of this chapter, the search for an optimal solution is proven to be NP-Hard. Thus, operations research techniques are mandatory to solve the listed contraints in a reasonable time.

Operations Research (OR) is a discipline involving analytical methods to help finding optimal or near-optimal solutions to complex decision-making problems. This is a practical discipline used in combinatory problem such as critical path analysis, project planning, network optimization, allocation problems... In the context of the system engineering OR is used to solve thermal, or mechanical problems, or any other domains relying on strong mathematical foundations. The current lack for a theoretical formalism of descriptive models based on SySML or NAF makes the usage of such techniques very rare. This is a gap we aim to fill with the research perspectives exposed in this chapter.

[5, Sect. 1] exposes a state of the art of the Operations Research. This first chapter demonstrates the complexity of the discipline; the applied algorithms shall be selected with care depending on the nature of the constraints to be solved. [8] also confirms how the fine-tuning of the algorithm is a key aspect of the high-performance of the research. Some algorithms are based on analytical approaches to find out exact solutions to linear problems. Other algorithms use heuristics mimicing behaviors observed from the nature such as the simulated annealing or genetic algorithms. In this case, these algorithms return approximate solutions in a reasonable time compared to the size of the problem space. Even though the algorithms have considerably progressed during the last decade, they remain a matter of experts and the selection of the appropriate algorithm depends of the nature of the constraints. For example, the ratio between the number of variables against the number of constraints influences the choice of such or such algorithm and subsequently the resolution performance [5, Sect. 1.3.1].

Another concern encountered with a large amount of the existing algorithms, is the numeric approximation of the real numbers. Some algorithms must be cautiously adapted so that the approximation errors have less impacts on the problem resolution [5, Sects. 1.2.4, 1.3.2].

Finally, the tuning of OR tools such as IBM CPlex also requires a specific expertise; an important number of options have to be set which impact the resolution problem capabilities.

4.1 Operations Research and Systems Engineering

In the context of the systems engineering process, there are two distinct phases inducing different concerns: the *bid phase* where the provider answers for a quotation proposal, and the *complete design phase* where the provider has won a contract and has to complete the design of the foreseen system. In both the two steps, the left part of the V lifecycle corresponding to the design phases – Concept, and Development in [12, Sect. 3.3] – have to be completed but the conditions are not the same; the bid step is extremely constrained regarding human resources and time. In this step, the research for an optimal solution contributes to the cost evaluation and so it is a strong support for the gain of the contract. Hardly constraint resources and time, and huge impact on the capability to win a contract militate for a very accessible technique of research operations. By the way, this accessibility quality generates additional benefits; the more the technique is made easy, the more the system engineers will use it, even in the case of won contract, to optimize every aspects of their design. This is then a major feature to support an holistic optimization of the system.

4.2 Local Search Algorithm Tuning Delegation

[1] proposes a black-box local search software component, named `LocalSolver` in order to combine both these accessibility and performance expectations. This black-box implements an heuristic based algorithm starting from a point in the solution space and moving in a reasonable time towards a non optimal but of high quality solution. The authors demonstrate that the tuning of the moves and the incremental evaluation algorithms consume respectively 30% and 60% of the development time. So, the idea consists on a programmatic transfer of the local search practionners' knowledge into the software black-box. LocalSolver

performs structured moves tending to maintain the feasibility of solutions at each iteration. The tool main specifities are: a simple mathematical formalism to model the problem in an appropriate way for local search resolution, and an effective black box local search based solver focused on the feasibility and the efficiency of moves.

The model formalism remains simple; the constraints and the space problem are declared through a simple language named LSP (which stands for Local Search Programming [14]), and there is no specific or very few code to write in order to tune the problem resolution (see an example in [1, Sect. 3]). This kind of black-box perfectly suits the need for supporting the systems engineering the way exposed in the previous paragraphs.

4.3 A Pragmatic Implementation of the Constraint Language

In 3.1, we showed that languages like OCL or CVL are convenient to express the constraints linked to the conceptual data model used in the MB^{2}SE approach. Thus, one topic to be studied is the link between such constraint language and the LSP language. A first approach consists of analyzing the OCL grammar, and trying to convert all combinations into the equivalent LSP syntax. The main blocking point is the language equivalence; the grammars are different and designed for specific purpose: a generic constraint declarations based on a conceptual data model on one hand, and the declaration of constraints for research operation treatments on the other hand. Thus, the OCL usage shall be limited to what the LSP language can address.

After, a first attempt in that direction we decided to implement the opposite: what about writing the constraints in the LSP language but adding it the dot notation? That way, there is no complex translater to implement but the one converting the dot notation based paths into something LSP understands: we chose to convert the path into LSP lists where items refers to the different data model elements (class, attribute, association end). The Sect. 3.1 demonstrates that the relevant benefit of the constraint language is the alignment with the data model which is still fulfilled that way. Regarding the constraint capabilities both implementations are finally restricted by the LSP language functionalities so there is no technical loss by using this final approach.

4.4 The Ontology-Based Optimization Process

Having resolved this language link, the engineering steps integrating the delegation of the optimization resolution to a local-search blackbox would be (see Fig. 2):

1. A system engineer describes her/his contraints an easy way from a modeling tool,
2. the constraints are automatically converted into a LSP-like language accompanied with the current model instances,

3. the black-box local search tool searches for a solution to the constraints considering the current model instances,
4. the solution is pushed back into the modeling tool with some facilities to help the system engineer understanding the changes.

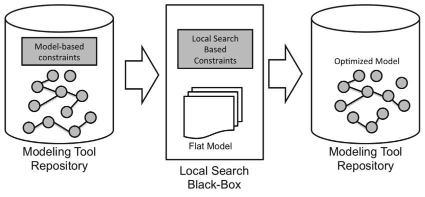


Fig. 2. Model optimization chain.

5 Example of an Optimization-Based Systems Engineering

The two following examples illustrate the usage of the optimization chain in an incremental approach.

The first example is split into two parts: First, we start from customer requests for application-based functions. The provider has a portfolio of applications supporting the expected functions by sets. We first try to answer the customer request through an optimized subset of applications covering all the requested functions and optimizing the cost.

Once selected, the applications have to be deployed into servers. The servers are themselves hosted into racks. Each items has a cost which must be minimized. Other constraints will be detailed later which are related to the disk space, the weight of the servers in the racks... The answer shall proposed an optimized infrastructure technology (IT) to deploy the selected applications.

A second example takes exactly the same customer requests but the idea is then to consider the global picture: optimizing the application coverage AND the IT deployment in the same row. The intent is to demonstrate that a global consideration which is possible through a well-managed MBSE implementation yields to a better optimized solution compared with the two chained but separate optimization problems.

5.1 Selecting the Applications

The first part of the problem is to select optimized subsets of the applications. Figure 3 shows the information model related to this question.

In order to test the proposed chain, 5 `CustomerNeeds` have been created. Each of them represents 5 different customer requests. A `CustomerNeed` is connected to a subset of the 100 `Functions`. The provider proposes 50 `Applications`, each of them supporting some of the `Functions`. Resolving the problem is then to create

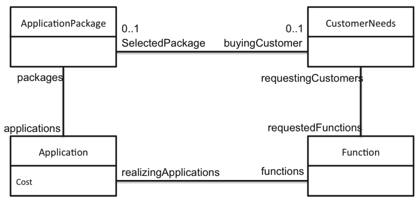


Fig. 3. Example of information model to be optimized.

1 `ApplicationPackage` per `CustomerNeed` which links some of the applications so that the following constraints are fulfilled. These constraints are expressed with an LSP-like language where sets of modeling elements are aggregated by declaring dot-notated pathes. Native keywords of the LSP language are displayed bold, while keywords refering to the data model are underlined.

R01 Minimize the application package costs.

```
for [cn in CustomerNeeds]
  minimize mlSum(cn.selectedPackage.applications.cost);
```

R02 Ensure all requested functions are provided in the corresponding package.

```
for [cn in CustomerNeeds]
  mlIsSubSetOf (cn.selectedPackage.applications.functions,
                cn.requestedFunctions
```

R03 1 package is provided for each customer needs set and vice-versa.

```
for [cn in CustomerNeeds]
  mlCard(cn.selectedPackage) <= 1;
for [ap in ApplicationPackage]
  mlCard(ap.deployedPackage) <= 1;
```

The function mlIsSubSetOf, mlSum and mlCard respectively indicates whether a set is included in another one, returns the sum of the set elements attribute values, and counts the number of elements in a set. Since we start with 5 independent `CustomerNeeds`, 5 `ApplicationPackages` are proposed in a 1 to 1 mode (requirement R03).

5.2 Designing the Optimized IT

Once the applications are selected, an appropriate IT has to be selected to deployed the applications. Figure 4 shows the related concepts to be modeled.

An IT solution (`TechnicalInfrastructure`) is to be provided for each `ApplicationPackage`. An IT solution is composed of `Racks` hosting `Servers`. We want to optimize the cost which is the sum of the servers costs + the sum of

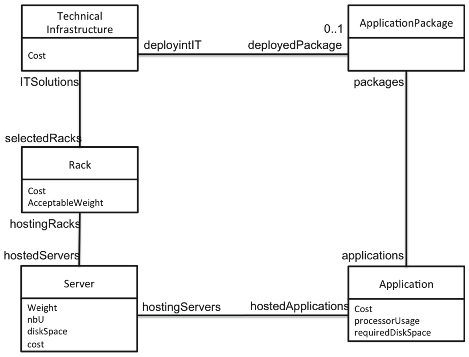


Fig. 4. A complementary example of information model to be optimized.

the racks costs, and ensure all the applications are deployed. On the other hand, the solution must fulfill the following technical constraints:

- The weight of servers is compatible with the Rack specification,
- The size in U (Unit) of the server is compatible with the Rack specifications,
- The disk space of the server is compatible with the application specifications,
- Assuming the applications are runned by the hosting server (for simplification purpose), the global processor usage consumption shall be < 0.9 (90%).

This is more formally translated into the LSP-like language:

R04 Minimize the IT Solution Cost.

```
for [ti in TechnicalInfrastructure]
  minimize (mlSum(ti.selectedRacks.cost) +
           mlSum(ti.selectedRacks.hostedServers.cost);
```

R05 Find a solution for each Package.

```
for [ap in ApplicationPackage]
  mlCard(ap.deployingIT) <= 1;
```

R06 Weight constraint.

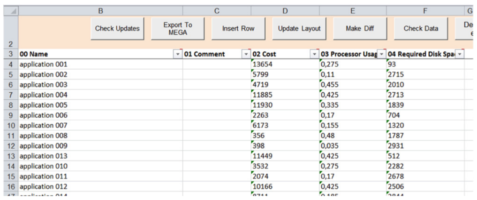
```
for [r in ApplicationPackage.deployingIT.selectedRacks]
  mlSum(r.hostedServers <= r.acceptableWeight);
```

R07 Processing Usage Constraint.

```
for [s in ApplicationPackage.deployingIT.selectedRacks.hostedServers]
  mlSum(s.hostedApplications.processorUsage <= 0.9);
```

5.3 Optimizing with the HOPEX-LocalSolver Bridge

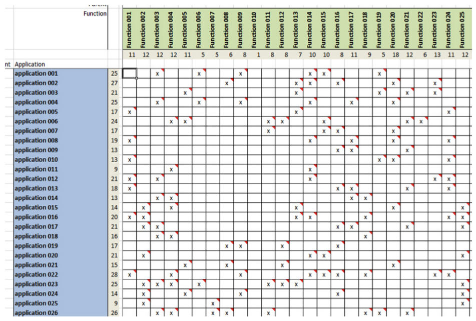
A prototype has been developed which is based on the HOPEX for NAF modeling tool (MEGA company) and the LocalSover local-search black-box component (Innovation24 company). The implementation follows the principle described in the Fig. 2. The HOPEX for NAF contains both the initial model and the LSP-Like constraint declarations. Fake data have been created for the purpose of the test of this paper but actual optimization has also been performed for a French MoD project on a more representative set of data. Each modeling concept (`Application`, `ApplicationPackage`...) involved in a constraint is converted from the HOPEX tool to a corresponding flat file (see Fig. 5).



	00 Name	01 Comment	02 Cost	03 Processor Usag	04 Required Disk Spa
4	application 001		13654	0,275	93
5	application 002		5799	0,11	2715
6	application 003		4719	0,455	2010
7	application 004		11885	0,425	2713
8	application 005		11930	0,335	1839
9	application 006		2263	0,17	704
10	application 007		6173	0,155	1320
11	application 008		356	0,48	1787
12	application 009		398	0,035	2931
13	application 013		11449	0,425	512
14	application 010		3532	0,275	2282
15	application 011		2074	0,17	2678
16	application 012		10166	0,425	2506

Fig. 5. A CSV flat file corresponding to the application concept.

Other CSV flat files are also generated for each relation between two concepts. Refer to Fig. 6 for an example of the relation between the applications and the functions displayed as a 2-D table.



Function		Function 001	Function 002	Function 003	Function 004	Function 005	Function 006	Function 007	Function 008	Function 009	Function 010	Function 011	Function 012	Function 013	Function 014	Function 015	Function 016	Function 017	Function 018	Function 019	Function 020	Function 021	Function 022	Function 023	Function 024	Function 025
		11	12	12	12	11	5	5	6	8	1	8	8	7	10	10	8	11	9	5	18	12	6	13	11	12
Application																										
application 001	25			x			x			x					x	x				x						
application 002	27								x					x	x		x				x			x		
application 003	21					x								x						x		x		x		
application 004	25			x			x			x					x			x			x					
application 005	17	x												x											x	
application 006	24				x	x						x	x			x						x	x			
application 007	17											x				x	x				x					
application 008	19	x													x			x			x				x	
application 009	13																x	x				x				
application 010	13	x																		x	x				x	
application 011	9				x										x											
application 012	21	x		x											x									x	x	
application 013	18	x															x	x				x			x	
application 014	13			x	x													x	x							
application 015	14		x		x									x							x					x
application 016	20	x	x											x	x				x						x	x
application 017	21		x	x													x	x				x				x
application 018	16			x	x														x							
application 019	17								x	x			x				x									
application 020	21		x												x	x										x
application 021	15					x			x				x								x					
application 022	28	x			x					x					x	x	x		x					x	x	
application 023	25		x	x	x		x					x	x	x												x
application 024	14		x			x				x							x									x
application 025	9		x					x																		x
application 026	26			x	x			x	x			x							x	x		x				

Fig. 6. A 2-D table corresponding to the relation between applications and functions.

The usage of this modeling tool allows creating transparent links between the textual constraint and the referenced data model artefacts. These links are used to automate the export of the appropriate flat files according to the constraints contents. Once the computation is done, the results are against transferred into the flat files for difference visualization and re-import into the modeling tool.

From the local search toolbox perspective, the CSV files are the input data. What is missing to complete the solution research are the constraints. All the constraints expressed in the LSP-like language for friendliness purpose have to be converted into the actual LSP language. In this language, the constraints are declared and contribute to the building of a constraint tree which is solved by the local search tool. Such conversion leads to the rewriting of the requirements as follows:

RA Defines the solution space for problem A.

```
mlDecideRelation(selectedPackage);
mlDecideRelation(applications);
mlDecideRelation(functions);
```

R01 Minimize the application package costs.

```
local sumCosts = {};
for [cn in md.[CustomerNeeds]]
  minimize mlSum (cn, { selectedPackage, applications, cost });
```

R02 Ensure all requested functions are provided in the corresponding package.

```
for [cn in md.[CustomerNeeds]]
  mlIsSubSetOf (cn, { selectedPackage, applications, functions },
                cn, { requestedFunctions });
```

R03 1 package is provided for each customer needs set.

```
for [cn in md.[CustomerNeeds]]
  mlCard(cn, { selectedPackage }) <= 1;
for [ap in md.[ApplicationPackage]]
  mlCard(ap, { deployedPackage }) <= 1;
```

RB Define the solution space for problem B.

```
mlDecideRelation(deployingIT);
mlDecideRelation(selectedRacks);
mlDecideRelation(hostedServers);
mlDecideRelation(hostedApplications);
```

R04 Minimize the IT Solution Cost. This is transformed the same way...

As explained in the Sect. 4.3, the LSP-like language is simply converted by replacing any concept C into a statement $md[C]$ where md is a map of modeled data and the index matches the concept identifier in the modeling tool. For this index, the map returns of the concept instances with their property values. A dot-based path $o.r1.r2....a$ is converted into a sequence $o, \{r1, r2, \dots, a\}$, so the first element of the path followed by a list of identifiers matching the path. Depending on the context the last attribute can be optional. The result of such a path must be the set of the elements explored from o (e.g $mlCard()$) or the result of a function applied on the a property of the set elements (e.g. $mlSum()$);

The implementation of the constraints into the actual LSP language raised several writing and transformation rules. First of all, *each rule must be independent to the others*. This means the variables declared in the context of a constraint shall not conflict with any other constraint. This is obvious if the LSP program is written in the context of a program editor but the initial context is the modeling tool where the constraints are individual modeling elements refering to the MB^{2}SE information model. In this context, all the variables are declared `local` (see `sumCosts` in requirement R01 as an example). The To-LSP converter also adds curling braces around each constraint so that to ensure the variable declarations are only specified in the context of the constraint.

Secondly, *the space of the solution must be explicitely declared.* For the first optimization problem, only the relations `selectedPackage`, `applications`, and `functions` are allowed to be updated. This must be explicitely said so the added requirements RA and RB. For simplification reason, the function mlDecideRelation() hides the LSP code which transforms the relation between a pair of modeling elements into an open boolean which can take one of the `false` or `true` values, the last one indicating the relation exists. For all the variables not allowed to be changed, their values are initiated from the modeling repository data with fixed values.

This problem space definition mechanism is replicated for the concept of the data model; each concept corresponds to a list of arrays matching the concepts' attributes. They are initiated according to the modeling repository data and only those used in the left part of a constraint are included in the problem space. In this case, the user has nothing to do but declaring the constraints.

Note also that requirement R03 is in fact already formalized by the association multiplicities set on the relation between `CustomerNeeds` and `ApplicationPackage` (Fig. 2). So, we can remove the explicit requirement and introduce an automated generation based on the multiplicities this for the sake of simplicity.

5.4 Performance of the Optimal Solution Search

Once converted into LSP program, the constraints are analyzed by the tool and a simulated annealing algorithm [23] is launched to find out an optimal solution. A solution which is not guaranteed to be the best – but at least a "good" solution – is computed in a timeframe defined by the user. The longer the timeframe is, the greater is the probability to be closed to an optimal solution. Then, the opened relations, attributes, and set of individuals are updated according to the found solution. All the changes are transferred again into the flat files (see Figs. 5 and 6) which allow understanding which are the changes, and automating the reimport into the modeling tool.

Even though the local search toolbox is considered a black-box fed with declared constraints, we have to consider the impact of the different ways the constraint can be written. Let's consider the requirement R01 stating the minimization of the application package costs.

```
for [cn in CustomerNeeds]
  minimize mlSum(cn.selectedPackage.applications.cost);
```

This requirement can be written using only the LSP native language as follows:

```
local sumCosts = {};
for [cn in CustomerNeeds] {
  sumCosts[cn] <- sum[ap in ApplicationPackage](
    selectedPackages[cn][ap] ? (
      sum[app in Application](
        Applications[ap][a] ?
    Application[a].Cost : 0)) : 0);
  minimize sumCosts[cn];
}
```

The main idea of the first formulation is: to sum on $Applications[a].Cost$, we need to check if there is a path from cn to ap. This path is decomposed in two relations: `CustomerNeeds` – `selectedPackage` → `ApplicationPackages`, and `ApplicationPackages` – `applications` → `Application`. The sum checks whether there is path between cn and ap, if so it looks for a path between ap and a.

Another way of writing the same objective is the following:

```
local sumCosts = {};
for[cn in CustomerNeeds] {
  sumCosts[cn] <- sum[ap in ApplicationPackage]
    [app in Application](
      selectedPackage[cn][ap] * applications[ap][a] * Application[a].Cost);
  minimize sumCosts[cn];
}
```

The main idea here is that we sum of every 3-tuple on cn, ap and a, and if no path exists between the three, the following product : $selectedPackage[cn][ap] \times applications[ap][a] \times Application[a].Cost$ results to 0 (taking into account that boolean values are 0 and 1 in LSP language).

As far as the LocalSolver tool is concerned, both formulations are not equivalent in terms of performance to obtain the results. We run LocalSolver twice on a set of 5 customer needs (CN1 . . . CN5), changing the formulation of the above constraint, leading to the different results expressed in the Table 1. Best results are displayed in bold format.

Table 1. Local solver execution with different constraint writings.

Formulation	CN 1	CN 2	CN 3	CN 4	CN 5	Iteration	Moves	Time
First	**16846**	**18886**	**29591**	**22653**	**35088**	401404	802870	10 s
Second	19562	23763	33514	37210	37735	240000	464145	10 s
Second'	**16846**	23763	33514	27983	35382	27713443	55425069	10 mn

In the same given time (10 s), the first formulation gives better results in every customer needs. Even after 10 min, the solution of the second formulation doesn't reach the quality of the first formulation. In this example, time isn't enough to compensate for a computationally expensive formulation. For an identical time (10 s), we can observe that the numbers of iterations and moves – which reflects the heuristics based algorithm used to solve the problem – are higher for the first formulation, meaning LocalSolver could better explore the space of feasible solutions (i.e. could explore more solutions in the same time). So, the formulation of the constraint not only affects the complexity of the evaluation, but also the quality of the results given by LocalSolver. As LocalSolver's time spent is controllable, the issue to properly face is not the time that LocalSolver spends on solving the problem but the quality of the operations that the local search tool can do during this given time.

One way to reduce the risk of bad solutions due to the writing of constraint expressions by non LSP experts is by transforming the constraint graphs into an equivalent one but proven to be optimal regarding the search performance. There are currently works done in that direction by the developers of the LocalSolver tool so for us the issue of finding an optimal solution will continue to be deletaged to the optimization tool as a black box regardless the writing of the expressions. Another more affordable way, is by wrapping the constraint implementation into predefined functions with the best known approach regarding the implementation. This is what has been done by proposing the functions $mlIsSubSetOf()$, $mlSum()$, or $mlCard()$. This function hides the complex implementation and give the end-user a more friendly understanding of the constraint meanings. Regarding the $mlSubSetOf()$ function, a naive implementation leads to the resolution of an actual project in 37 mn. By considering the function as a pure function (which from a programmactic perspective always returns the same result for a given input and has no side-effect), it is possible to optimize the computation time of this recursive function by using a memoization mechanism [16]. This dramatically reduced the general complexity and the same computation is now done in 2 s while still keeping the same function declaration for the end-users.

6 Future Work

This paper demonstrates how the declaration of constraints against a conceptual data model can be automatically converted into an LSP program including the modeling data which enables the computed search for optimal solutions satysifying the constraints. This mechanism allows the update of concept attributes, and corresponding relations through the simple declaration of contraints based on the LSP language augmented with the dot notation for pathes.

We also explained how the writing of the constraints can alterate or improve the obtention of a solution within a reasonable time. This is why a library of predefined functions hides this writing complexity. By applying this approach in an actual French MoD project, we learnt that additional functions are again necessary to be developed to foster the usage of the constraint declarations, especially all the set functions diving into the model (union, intersection, different modes of properties agregation...).

The modeling principles established by the MB^{2}SE framework [6] create new issues regarding the research for optimality. This framework relies on the declaration of a modeling data model mapped to a storage metamodel such as NAF. This mapping separates the business perspectives from the technical requirements relate to modeling tool interoperability. As illustrated in [7], two system engineers can create distinct data models pointing to the same modeling elements. So the following questions to be studied and solved: How to solve system engineering constraints declared in two different perspectives? How to detect whether the system engineering constraints declared for one perspective impact the ones declared in another perspective? Is there similar issues to be solved in the context of a unique conceptual data model? For example, how to handle the inheritance between two concepts dealing with a common subset of modeling elements?

Acknowledgement. I warmly thank Thierry Benoist from the LocalSolver company for his precious help regarding the constraints implementation. I also thank Erwan Beurier from IMT Atlantiques (Institut Mines-Telecom Bretagne, France) for his first implementation of the MEGA to LocalSolver converter.

References

1. Benoist, T., Estellon, B., Gardi, F., Megel, R., Nouioua, K.: Localsolver 1. x: a black-box local-search solver for 0-1 programming. 4OR Q. J. Oper. Res. **9**(3), 299–316 (2011)
2. BKCASE. Sebok, guide to the systems engineering body of knowledge. http://sebokwiki.org
3. Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., Wąsowski, A.: Cool features and toughdecisions: a comparison of variability modeling approaches. In: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, pp. 173–182. ACM (2012)
4. Dumitrescu, C.: CO-OVM: a practical approach to systems engineering variability modeling. Université Panthéon-Sorbonne - Paris I, Theses (2014)
5. Dupin, N.: Modélisation et résolution de grands problèmes stochastiques combinatoires. Ph.D. thesis, Université Lille 1, Laboratoire Cristal (2015)
6. Ernadote, D.: An ontology mindset for system engineering. In: 2015 1st IEEE International Symposium on Systems Engineering (ISSE), pp. 454–460. IEEE (2015)
7. Ernadote, D.: Ontology reconciliation for system engineering. In: 2016 IEEE International Symposium on Systems Engineering (ISSE), pp. 1–8. IEEE (2016)
8. Estellon, B., Gardi, F., Nouioua, K.: Two local search approaches for solving real-life car sequencing problems. Eur. J. Oper. Res. **191**(3), 928–944 (2008)
9. Hammami, O.: Multiobjective optimization of collaborative process for modeling and simulation-$<$ q, r, t. In: 2015 IEEE International Symposium on Systems Engineering (ISSE), pp. 446–453. IEEE (2015)
10. Hammami, O., Houllier, M.: Rationalizing approaches to multi-objective optimization in systems architecture design. In: 2014 8th Annual IEEE Systems Conference (SysCon), pp. 407–410. IEEE (2014)
11. Haugen, Ø., Wasowski, A., Czarnecki, K.: CVL: common variability language. SPLC **2**, 266–267 (2012)

12. INCOSE: INCOSE System Engineering Handbook, 4 edn. (2015)
13. InfoGrid. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? http://infogrid.org/trac/wiki/Reference/PidcockArticle
14. Innovation24. Lsp reference manual. https://www.localsolver.com/documentation/lspreference/index.html
15. ISO: ISO/IEC 15288:2008, Systems and software engineering–System life cycle processes (2008)
16. Milewski, B.: Category theory for programmers (2014)
17. Morkevicius, A.: Integrated modeling: adopting architecture frameworks for model-based systems engineering. http://163.117.147.32/joomlaaeis/sese/slides/SESE_2014-Integrated_Modeling_Aurelijus.pdf
18. NATO. Naf v4 meta-model (model) (2013). http://nafdocs.org/modem/
19. OMG: Object Constraint Language, version 2.4, February 2014. http://www.omg.org/spec/OCL/2.4/PDF
20. OMG: OMG Systems Modeling Language (OMG SysML ™), version 1.4. June 2015. http://www.omg.org/spec/SysML/1.4/PDF
21. OMG: OMG Unified Modeling Language ™(OMG UML). Structured Classifiers, p. 181. OMG, March 2015. http://www.omg.org/spec/UML/2.5/PDF
22. Richters, M., Gogolla, M.: On formalizing the UML object constraint language OCL. In: International Conference on Conceptual Modeling, pp. 449–464. Springer (1998)
23. Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. In: Simulated Annealing: Theory and Applications, pp. 7–15. Springer (1987)
24. Warmer, J.B., Kleppe, A.G.: The Object Constraint Language: Precise Modeling with UML (Addison-Wesley object technology series) (1998)

On-Time-Launch Capability for Ariane 6 Launch System

Stéphanie Bouffet-Bellaud(✉), Vincent Coipeau-Maia, Ronald Cheve, and Thierry Garnier

ArianeGroup, Route de Verneuil, 78133 Les Mureaux, France
{stephanie.bouffet, vincent.coipeau, ronald.cheve, thierry.garnier}@ariane.group

Abstract. For space transportation systems used to place satellites in space, the launch rate fulfillment (turnover) and the launch delay reduction (avoid additional cost) are key parameters driving the launch cost. In addition, launching on-time is beneficial to the payload operator business model.

In Europe, Ariane 5 has demonstrated its unmatched reliability with more than 80 successful consecutive launches. Due to increasing competition worldwide for space transportation systems, Ariane 6 will have to achieve the same reliability but with twice the launch cadence.

This is the reason why, on Ariane 6, the On-Time-Launch capability has been taken into account since upstream development phases. Firstly, the main drivers for this performance have been identified (lessons learnt) on the complete life cycle. Based on cost approach, an allocation methodology has been defined, including risk severity and occurrence management. Then, mitigation actions and robustness to degraded cases have been deduced.

In the frame of CSDM 2018, the On-Time-Launch Capability for Ariane 6 Launch System and associated methodology is proposed to be presented during a 30 min talk.

1 Context and Study Logic

Ariane 6 is the new European launcher with two versions A62 and A64. Its performances to geostationary transfer orbit will be 5 tons (A62 with 530 t weight at lift-off) and more than 10,5 tons (A64 with 860 t weight at lift-off) (Fig. 1).

Ariane 6 development is in progress. First flight is planned in 2020.

Ariane 6 launcher is composed of:

- a Central Core providing thrust with a Lower Liquid Propulsion Module (LLPM) equipped with a Vulcain engine and an Upper Liquid Propulsion Module (ULPM) equipped with a Vinci engine;
- two (A62) or four (A64) Equipped Solid Rockets (ESR) boosters depending on the Ariane 6 Launcher configuration;
- an Upper Part including the fairing, the Launch Vehicle Adaptor, the Payload Adaptor Fitting (to fit with payload interface diameter) and any system allowing to perform dual launch or multiple launch.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 33–43, 2019.
https://doi.org/10.1007/978-3-030-04209-7_3

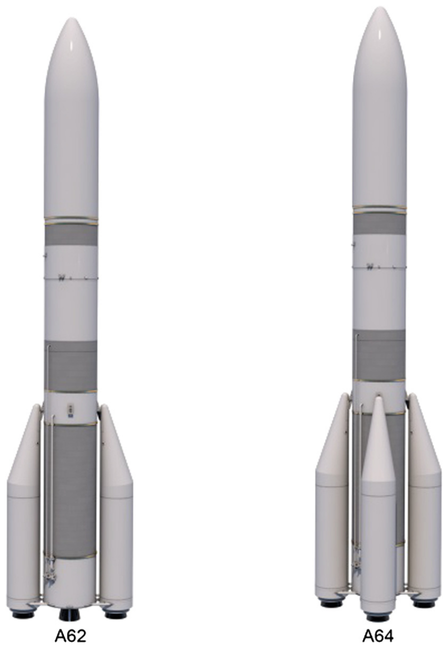


Fig. 1. Ariane 6 versions

Ariane 6 fully integrated launcher is built with the following process (Fig. 2).

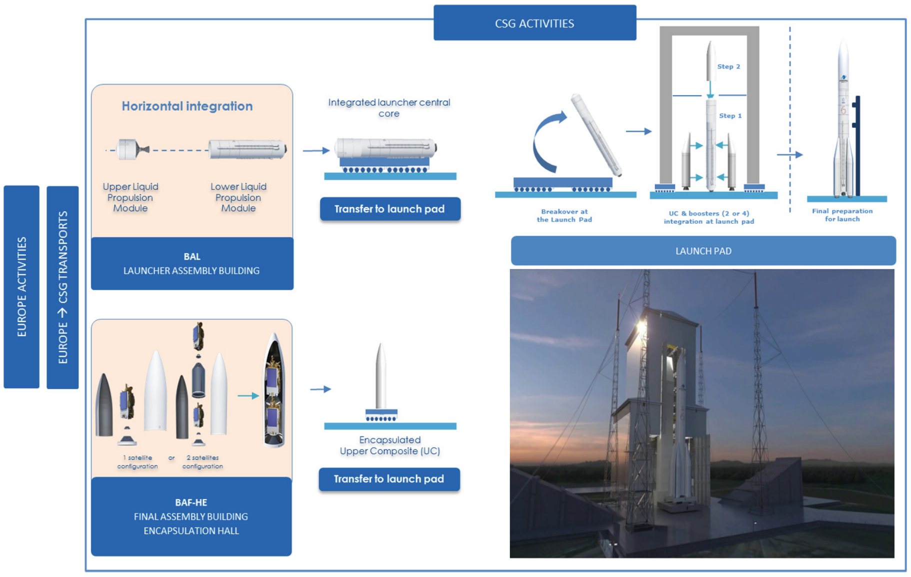

CSG ACTIVITIES
EUROPE ACTIVITIES
EUROPE → CSG TRANSPORTS
Horizontal integration
Upper Liquid Propulsion Module
Lower Liquid Propulsion Module
BAL
LAUNCHER ASSEMBLY BUILDING
Integrated launcher central core
Transfer to launch pad
Breakover at the Launch Pad
Step 2
Step 1
UC & boosters (2 or 4) integration at launch pad
Final preparation for launch
LAUNCH PAD
1 satellite configuration
or
2 satellites configuration
BAF-HE
FINAL ASSEMBLY BUILDING
ENCAPSULATION HALL
Encapsulated Upper Composite (UC)
Transfer to launch pad

Fig. 2. Ariane 6 integration logic

European Space Agency (ESA) is in charge of Ariane 6 Launch System (Launcher System and Launch Base). ArianeGroup is prime contractor and design authority for Ariane 6 Launcher System. As design authority, ArianeGroup is in particular responsible for requirement derivation, design rules and requirement verification.

Due to increasing competition worldwide for space transportation systems, Ariane 6 will have to achieve the same reliability as Ariane5 with twice the launch cadence.

The Ariane 6 On-Time-Launch approach aims at considering this topic in upstream development phase in order to anticipate as much as possible launch delay risk. The objective is the improvement of launch availability by limiting launch delay and reducing associated cost.

The launch planning delay and disruption during operation have a cost impact. The fact to master these operation disruptions or planning delays can be a way to reduce Recurring Cost (risk management ref.1).

The global logic (Fig. 3) is based in three main steps:

- identification of delay risks (lessons learnt and current project specificities)
- classification and selection of major delay risks
- reduction of major delay risks (occurrence and severity aspects)

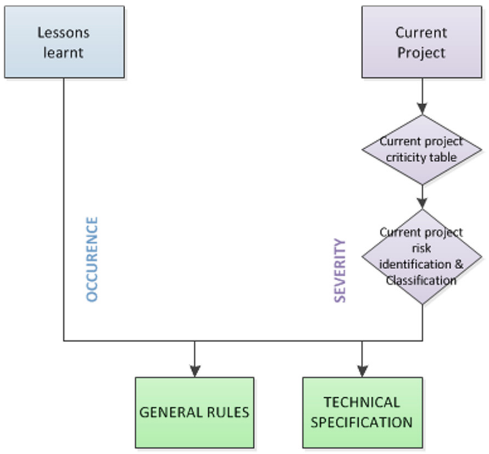
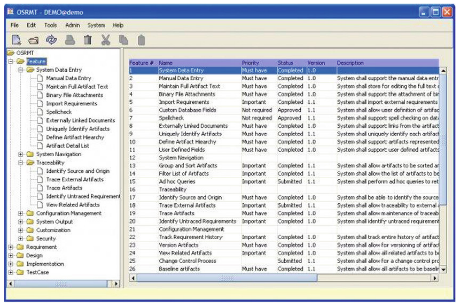

Lessons learnt
Current Project
Current project criticity table
Current project risk identification & Classification
OCCURENCE
SEVERITY
GENERAL RULES
TECHNICAL SPECIFICATION

Fig. 3. Ariane 6 On Time Launch study logic

The DOORS system engineering tool is used for the requirements allocation and flow down in the technical specification (Fig. 4).

Requirements are based on either Probabilistic approach (like MTTR (Mean Time To Repair)/MTBF (Mean Time Before Failure) or determinist approach (see ref.2 for vocabulary definition).



Feature #	Name	Priority	Status	Version	Description
1	System Data Entry	Must have	Completed	1.0	
2	Manual Data Entry	Must have	Completed	1.0	System shall support the manual data entr
3	Maintain Full Artifact Text	Must have	Completed	1.0	System shall store for editing the full text
4	Binary File Attachments	Must have	Completed	1.0	System shall support the attachment of bi
5	Import Requirements	Important	Completed	1.1	System shall import external requirements
6	Custom Database Fields	Not required	Approved	1.1	System shall allow user definition of artifac
7	Spellcheck	Not required	Approved	1.1	System shall support spell checking on dat
8	Externally Linked Documents	Must have	Completed	1.0	System shall support links from the artifact
9	Uniquely Identify Artifacts	Must have	Completed	1.1	System shall uniquely identify each artifact
10	Define Artifact Hiearchy	Must have	Completed	1.0	System shall support artifacts represented
11	User Defined Fields	Must have	Completed	1.0	System shall support user defined artifacts
12	System Navigation				
13	Group and Sort Artifacts	Important	Completed	1.1	System shall allow artifacts to be sorted ar
14	Filter List of Artifacts	Important	Completed	1.1	System shall allow the list of artifacts to be
15	Ad hoc Queries	Important	Submitted	1.1	System shall perform ad hoc queries to ret
16	Traceability				
17	Identify Source and Origin	Must have	Completed	1.0	System shall be able to identify the source
18	Trace External Artifacts	Important	Submitted	1.1	System shall allow traceability to external
19	Trace Artifacts	Must have	Completed	1.0	System shall allow maintenance of traceab
20	Identify Untraced Requirements	Important	Completed	1.0	System shall identify untraced requirement
21	Configuration Management				
22	Track Requirement History	Important	Completed	1.0	System shall track entire history of artifact
23	Version Artifacts	Must have	Completed	1.0	System shall allow for versioning of artifac
24	View Related Artifacts	Important	Completed	1.0	System shall allow all related artifacts to be
25	Change Control Process		Submitted	1.1	System shall allow for a change control pro
26	Baseline artifacts	Must have	Completed	1.1	System shall allow all artifacts to be baseli

Fig. 4. DOORS tool

2 Delay Risks Identification

The On-Time-Launch activity has firstly analyzed the customer need on this topic and performed a lessons-learnt on similar complex system.

2.1 Lessons-Learnt

The purpose of the present chapter is to have a status about risk of launch schedule slipping by performing a "lessons learnt" activity on similar projects (Ariane 5 and other Space-Systems projects).

The lessons-learnt activity was based on twenty years of experience in space domain, on three major space program and many databases of incidents.

This activity aims to identify causes and consequences of disruptions in operation and the stakeholders.

A classification and summary has been performed to identify the main unavailability contributors (Launcher, Ground, Weather, Check, etc.…).

A list of recommendations has been extracted from these lessons learnt.

Due to Ariane 6 launch rate specificities, some new unavailability contributors could appear. Indeed, the Ariane 6 launch rate objective (customer requirement) is twice higher than Ariane 5 one. The fact to be in a tight planning will highlight some new contributors. Complementary unavailability will appear and lessons-learnt activity is not sufficient to address the global Ariane 6 topic. A global Ariane 6 approach is necessary and proposed in paragraphs below.

2.2 Functional Risk Analysis

In order to take into account Ariane 6 specificities, a functional risk analysis for On-Time-Launch needs has been performed to identify the possible delay risk. Co-engineering working sessions (experts review) were organized (Fig. 5) for following life phases:

- Europe activities
- Europe → Centre Spatial Guyanais Transports

- Centre Spatial Guyanais activities.

Fig. 5. Co-engineering working sessions with high tech tools

This functional risk analysis consists on going through process of Assembly Integration and Tests (AIT) in each factory (exhaustive approach) and identifying risk of delay (examples: means failure, means not available, integration difficulties, etc.…).

All information gathered during the co-engineering working sessions were summarized in an excel file afterwards based on an "Ariane 6 APQP+ (Advanced Product Quality Planning) template.

A verification process was implemented in parallel to confirm that the process was adapted to the need.

An exhaustive list of around 5000 delay risks was obtained for all the phases analyzed. The need to classify and select the major ones was highlighted and it is the point of the following chapter.

3 Delay Risk Classification and Selection

Once the risk identification performed, the next step is the classification and selection/prioritization of possible delay risk in each life phase before launch.

3.1 Cost Analysis Wrt "Delay Cost and Cost Acceptability"

The global logic is based on three main parameters: delay duration/delay cost/cost acceptability. The objective is to elaborate a severity table on launch delay. This table shall cover all the life phases of the Launcher System.

The cost topic has been managed through qualitative criteria and lessons-learnt figure.

The ranking of the delay duration severity is based on the cost (money lost acceptability). The scale of this occurrence acceptability uses the Likelihood class table (Fig. 6) below:

Likelihood class	**Level**	**Description**
Extremely rare	P4	They are those failures so unlikely that they are not anticipated to occur during the entire operational life of Ariane, but nevertheless, have to be regarded as being possible.
Very rare	P3	They are those failure conditions unlikely to occur during Ariane life but that may occur once when considering the total operational life of the Ariane fleet
Rare	P2	They are those failures not anticipated to occur to each Ariane launcher but that may occur a few times when considering the total operational life of the fleet.
Occasional	P1	They are those failures anticipated to occur once during the operations of a limited set of Ariane launchers
Probable or frequent	P0	They are those failures anticipated to occur during the operations of each Ariane launcher.

Fig. 6. Likelihood class table

The Launch Pad destruction is treated at P4 which will be our limitation level for the On-Time-Launch severity table: The On-Time-Launch severity table will then used the likelihood level from P1 to P3.

The severity table allows linking cost and occurrence acceptability, this table concerns cost topic and is managed through qualitative criteria. Finally, the severity table makes a connection between the delay duration and the likelihood objective.

In order to derive the On-Time-Launch severity table, the objective is shared between the different life phases of the launcher preparation.

In this sharing allocation objective, it has been chosen to put more constraints on operations which are far from the expected launch slot because it is easier (more time) to treat contingencies. Indeed, very close to the “expected launch slot”, it remains very short time to treat contingencies without impact on “expected launch slot”.

That means that:

- constraints are higher on phase “Before integration process in Kourou”.
- constraints are minimized for Launch-Pad activities.

The allocation is shared between Europe, Kourou CSG before Launch Pad arrival, Launch Pad. The severity table makes a link, for each previous life phase, between the delay duration and the likelihood objective.

It is proposed to transform allocations in number of barriers. A barrier is a mitigation measure (storage, logistic logic, failure treatment, etc.…) put in place to avoid delays. The objective is to design an AIT concept robust to feared events which could lead to launch delay.

The assumption is taken that:

- 1 barrier allows avoiding an occurrence of P2.
- 2 barriers allow avoiding an occurrence of P4.
- Lessons learnt recommendations based on similar program, allows reducing risk (occurrence of P1: risk reduction by recommendation application).

This assumption (link between barrier and occurrence) is a convention which is usually used but it has to be shared and agreed with the authority who is the recipient of the evaluation.

Finally, the severity table makes a connection, for each previous life phase, between the delay duration and the "barrier/mitigation action" number to be implemented.

3.2 Major Delay Risk Selection

In order to classify the delay risks identified during the functional analysis, some criticality matrix have been established in consistency with On Time Launch Severity table.

Three criticality matrix have been established: Europe criticality matrix, Kourou CSG before Launch Pad arrival criticality matrix, Launch Pad criticality matrix.

These criticality matrix allocate for each delay risk, the risk level with regards to its severity and its occurence.

Once the matrix elaborated, the selection has been performed and lead to around 100 delay risks to be mastered. At this step, the classification in term of Ariane 5 lesson learnt occurrence level has not been used yet because the objective was first to catch and then master the delay risk in term of severity (duration) in consistency with On Time Launch Severity table. This Ariane 5 lesson learnt occurrence level will be used afterwards during the internal value analysis on delay cost evaluation when needed.

Based on these criticality matrix, the exhaustive list of around 5000 delay risks (see Sect. 2.2) has been reworked and a selection of major risks is created.

This selection is consistent with On Time Launch Severity table of Sect. 3.1:

- In Europe: selection of risks leading to delay wrt Launcher constitution in Kourou. For Europe factory: selection of risks leading to delay wrt factory exit
- In Kourou: selection of risks leading to delay wrt expected launch slot.

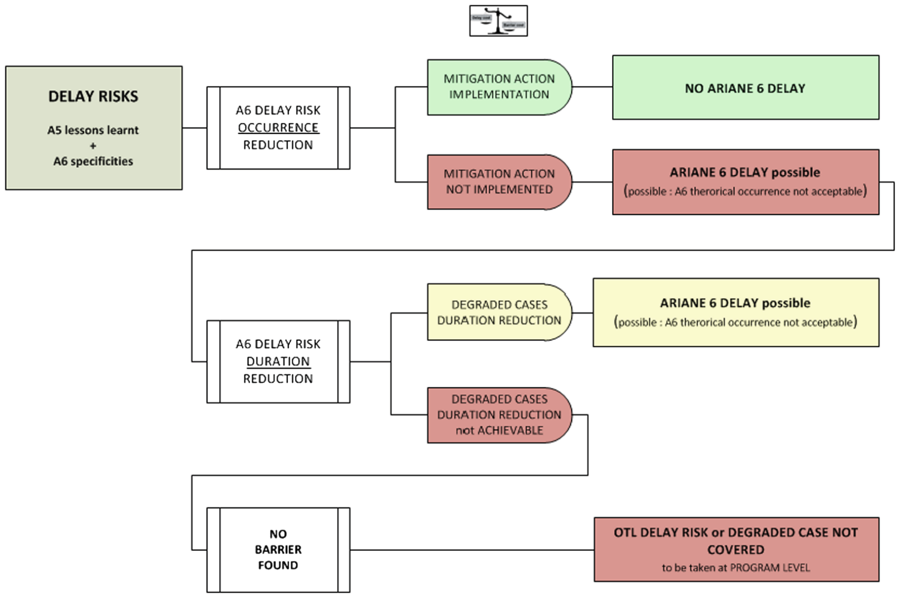
Additionally to this selection, Ariane 6 specific risks, leading to shorter delay wrt expected launch slot, have been also retained in order to focus on Ariane 6 specificities.

At this step, a list of Ariane 6 major delay risks is identified (100 risks).

4 Delay Risk Reduction

The On Time Launch approach is based on the fact that mitigation actions are implemented to avoid delay risk (delay occurrence reduction). In a perfect allocation, all the major delay risks identified won't occur because mitigation actions (occurrence) are implemented. In the real allocation, in front of some delay risks, no adequate mitigation actions (occurrence) are found. In this case, the delay risk can appear and it corresponds to an On-Time-Launch degraded case. This On-Time-Launch Degraded Case shall be managed as quickly as possible (mitigation actions (severity)) in order to limit as much as possible launch delay duration.

This mitigation action (barrier) logic is described in the following picture (Fig. 7):



DELAY RISKS
A5 lessons learnt
+
A6 specificities
A6 DELAY RISK OCCURRENCE REDUCTION
MITIGATION ACTION IMPLEMENTATION
NO ARIANE 6 DELAY
MITIGATION ACTION NOT IMPLEMENTED
ARIANE 6 DELAY possible
(possible : A6 therorical occurrence not acceptable)
A6 DELAY RISK DURATION REDUCTION
DEGRADED CASES DURATION REDUCTION
ARIANE 6 DELAY possible
(possible : A6 therorical occurrence not acceptable)
DEGRADED CASES DURATION REDUCTION not ACHIEVABLE
NO BARRIER FOUND
OTL DELAY RISK or DEGRADED CASE NOT COVERED
to be taken at PROGRAM LEVEL

Fig. 7. Ariane 6 On-Time-Launch mitigation actions (barriers) logic

4.1 Mitigation Action to Reduce Delay Risk Occurrence

The occurence mitigation actions (barrier) are answers to reduce On-Time-Launch Delay risk. The different types of mitigation actions to face risk occurrence are:

- SIZING ex: Means/infras sizing (MTBF/MTTR)
- HARDWARE ex: Means mechanical device
- ANALYSIS (development phase) ex: FMEA, Zonal Analysis
- PROCEDURES (exploitation phase)
- BUFFER
- etc.…

Few examples of delay risks and associated mitigations are detailed below:

1- Ground means and/or infrastructures failure leading to launch delay
In front of this delay risk, the following mitigations are proposed:

- Total Corrective Maintenance (TCM) management through failure occurrence reduction (MTBF - Mean Time Before Failure) and corrective maintenance (MTTR - Mean Time To Repair)
- Recovery procedure (including hardware implementation) to mitigate the delay

2- Operation (assembly/integration/test/maintenance) scheduling leading to launch delay
In front of this delay risk, the following mitigations are proposed:

- Operation duration specified for AIT and maintenance, learning curve follow and associated actions in case of drift
- Availability objectives specified in Maintenance and Exploitation Contracts.

3- Weather alerts leading to launch delay
In front of this delay risk, the following mitigations are proposed:

- Launcher system shall be operated with a value of ground wind in Kourou compatible with availability need, including sizing of launcher vehicule wrt to level of ground wind. The ground wind value is based on a database (ground wind measurements in CSG during 20 years) associated with an annual probabilistic approach.

4- Equipments/Products alerts leading to launch delay
In front of this delay risk, the following mitigations are proposed (Fig. 8):

- APQP+ (Advanced Product Quality Planning) method

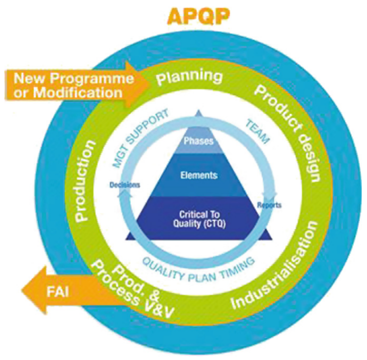


Fig. 8. APQP Method

5- Logistic and transport difficulties leading to delay

In front of this delay risk, the following mitigations are proposed:

- Analysis: Management of Transport Plans and Authorizations, Transport means, Handling means, Rupture of supply, etc.… to avoid delay

4.2 Mitigation Action to Reduce Delay Risk Severity

Once the task on delay risk occurrence reduction performed, some delay risk remain not mastered (no adequate mitigation actions (occurrence) found) and they can occur during Ariane 6 life.

In this case, we speak in this article about On-Time-Launch major degraded case. These On-Time-Launch major Degraded Case shall be managed as quickly as possible in order to limit as much as possible launch delay duration.

The following types of On-Time-Launch degraded cases are identified and shall be managed in limited duration consistent with On-Time-Launch severity table:

- Launcher element or Payload damage
- Equipments/Products alerts
- Problem during tests leading to change equipments
- Transatlantic boat failure
- Strike in Port
- Etc…

5 Conclusion and Way Forwards

Due to world competitiveness increasing, launch rate fulfillment and launch delay reduction are key parameters for launcher program.

For Ariane 6, On-Time-Launch capability has been taken into account during upstream phases.

Based on cost approach, an allocation methodology has been defined, including risk severity and occurrence management implementation. Then, mitigation actions and degraded cases have been deduced.

The way forwards will consist on requirement verification logic definition and requirement compliance verification through DOORS tool (Requirement Verification Plan, Verification Compliance Document). The On-Time-Launch performance will be monitor with indicators implementation and with continuous improvement in exploitation.

The On Time Launch objectives achievement is foreseen at Full Operational Capability after 2023, benefiting from learning curve during the transition period since the first flight in 2020.

References

1. Walden, D., Roedler, G., Forsberg, K., Hamelin, D., Shortell, T.: Systems Engineering Handbook
2. ECSS-Q-ST-30-09C - Availability analysis

Towards a Standards-Based Domain Specific Language for Industry 4.0 Architectures

Christoph Binder[1](✉), Christian Neureiter[1](✉), Goran Lastro[1], Mathias Uslar[2], and Peter Lieber[3]

[1] Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Urstein Süd 1, 5412 Puch/Salzburg, Austria
{christoph.binder,christian.neureiter,goran.lastro}@fh-salzburg.ac.at

[2] OFFIS – Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany
mathias.uslar@offis.de

[3] LieberLieber Software GmbH, Handelskai 340, Top 5, 1020 Wien, Austria
peter.lieber@lieberlieber.com

Abstract. Advances in research and development paved the way for a new revolution concerning industrial manufacturing, called Industry 4.0. Cyber-Physical Systems (CPS) contain methods for ubiquitous monitoring of information and synchronizing it with any other component on each hierarchical level participating in the value chain. Developing Industry 4.0 architectures for pointing out the structural cooperation of these Systems of systems (SoS) is a challenging task including a lot of different stakeholders. To bring together knowledge and experience, a common methodology is necessary. Regarding this, several German industrial associations created a suitable reference architecture, called Reference Architecture Model Industry 4.0 (RAMI 4.0). In this paper, a Domain Specific Systems Engineering approach using a Domain Specific Language (DSL) based on the results of this reference architecture is proposed and evaluated by a suitable case study.

1 Introduction

Optimized management of available resources in order to maximize profit and at the same time reducing costs and expenses is the main goal of most manufacturing companies. Results from research and development in the area of information technology (IT) offer new possibilities to support this goal, which drive change in the present industrial area and lead the path to a new form of an automation driven industry, the so-called Industry 4.0. An example of a technology resulting from this change is cyber-physical system (CPS). CPS are mainly intelligent components of a manufacturing process, where they take over a specific task. The main advantage regarding productivity is, that CPS are able to find the economically most valuable decision on their own, based on information provided by other CPS taking part in the system [6]. As discussed before, the aim

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 44–55, 2019.
https://doi.org/10.1007/978-3-030-04209-7_4

of Industry 4.0 is to advance automation in manufacturing companies. In order to achieve this, data between machines, processes and lots need to be exchanged over direct communication structures. In [8], several documents concerning the definition of Industry 4.0 and its components have been analyzed and some criteria describing it have been formulated. According to [23], a Digital Twin of the physical production system needs to be realized. As the name assumes, this concept refers to the digital representation of a physical asset, containing its information throughout the whole life cycle. Since some lots like screws do not have any possibility to communicate, its Digital Twin helps collecting useful data and provides them to other CPS in the industrial system.

To describe the structure and behavior of a system, a system architecture needs to be defined. Therefore, a tailored architecture to describe Industry 4.0 based systems must be available, managing the complexity that comes with it. Furthermore, this type of architecture should (1) be able to deal with dynamic changes that occur in this area (2) simplify complexity in order to improve user experience and (3) increase productivity instead of generating overhead, according to [25]. To deal with this issue, several approaches like RAMI 4.0 [7] or IIRA [16] have been launched. Although both reference architectures provide a certain tool-set for developing concrete industrial system architectures, the goal and used methods differ from each other. A comparison between RAMI 4.0 and IIRA can be found in [9].

The definition of RAMI 4.0 as well as its use and methods are clearly defined in the German norm DIN SPEC 91345 [5]. However, the application of RAMI 4.0 is not properly formalized and there are no solutions yet existing dealing with this. This is a big issue to solve, because ensuring the applicability of RAMI 4.0 could take this approach a big step forward towards establishing it in the industry. A suitable technology needs to be determined in order to deal with this open gap. Since the aspects concerning Industry 4.0 are usually too complicated to be treated with a generic technology, a domain specific approach tailored to the application domain must be specified. Furthermore, another required action is to provide comprehensive tool support for developing architectures of systems based on RAMI 4.0.

To address these aspects, this contribution is structured as follows: In Sect. 2 an overview of RAMI 4.0 and domain specific systems engineering is given. Hereafter, the creation of the domain specific language (DSL) is stated in Sect. 3. Based on a suitable demonstration example, the applicability is demonstrated in Sect. 4. Finally, in Sect. 5 the paper is summarized and then the conclusion is given.

2 Related Work

2.1 Domain Specific Architecture Framework

The Reference Architecture Model Industrie 4.0 (RAMI 4.0), depicted in Fig. 1, has been developed by the Plattform Industrie 4.0, a union of leading German associations in the industrial area. The three-dimensional model, derived from the already established Smart Grid Architecture Model (SGAM) [5], has been developed to create a common understanding. To do so, it contains standards and use cases related to Industry 4.0. Due to the big influence on the German industry of its creators, the reference architecture encloses multiple industry sectors and its span ranges over the complete value chain. This allows the system to be seen as a whole to find connections and display tasks or sequences of events over the whole process as well as create the possibility of providing a detailed consideration of parts from the system [1].

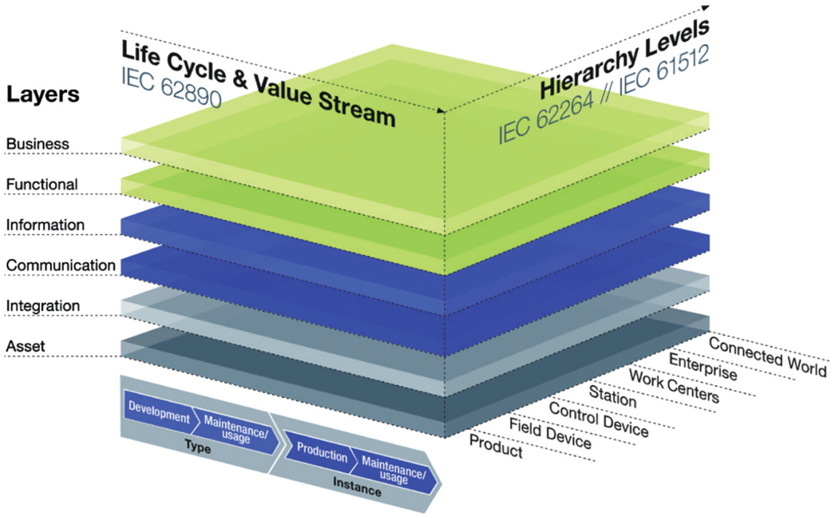


Fig. 1. Reference Architecture Model Industrie 4.0 (RAMI 4.0) [1]

To represent an asset over its whole life cycle, the horizontal axis has been introduced. It defines a product according to IEC 62890 [12] as type and instance, whereas the type represents the asset during development and prototype creation. However, the instance states a product as individuality and all its administration. Furthermore, the second axis deals with the classification of an item within the factory. Based on IEC 62264 [11] and IEC 61512 [10] a single product can be located regarding its spreading, from connected world over enterprise up to a single device used in production [1]. The top-down arrangement of the layers enables the classification of subjects according to their task areas. This also enables the mapping of the single system development processes to their respective area. The system analysis takes place at the top layers, more detailed the Business Layer and the Function Layer. It describes the conditions and business processes the system has to follow and in further consequence the run-time environment of the system with all functionalities of its services and applications. The Information Layer provides all kind of data to its adjacent layers and the Communication Layer takes care of the connections within the system. To deal with all characteristics of CPS the Integration Layer has been defined only to display the physical objects on the Asset Layer.

2.2 Domain Specific Systems Engineering

Since Industry 4.0 is a widespread and challenging domain, engineering of systems is a complex task and needs to be confronted with suitable methods. Usually, in a complex field not a single system is constructed but an interaction of multiple homogeneous systems called System-of-Systems (SoS). According to [15], two disciplines need to be fulfilled. On the one hand, decent knowledge about the domain to operate with should be appropriated, on the other hand, it is mentioned that systems engineering management contributes significantly to the overall success. While a system is determined to fulfill a certain purpose, a SoS offers a solution for a more complex and extensive problem area [2]. Dynamic structures and changing conditions hinder the modeling of such a system with a generic approach.

To keep the overview of every single aspect included during the engineering of a SoS the concept of Model Based Systems Engineering (MBSE) is usually used. It enables stakeholders to gain different viewpoints by abstracting the architecture into different levels. Furthermore, it provides technologies to ensure the availability of an iterative development process. The application of the concepts of MBSE must be assured by a suitable modeling language. Due to its freedom, a so-called General Purpose Language (GPL) can be used in a wide variety of application domains. These language with low constraints is tailored to develop systems working in multiple areas. On the other hand, for describing detailed processes within a certain area, this kind of language is missing specifications. Therefore, the designing of a DSL usually is unavoidable in order to consider all domain-specific features [24]. To increase the effectiveness of the application of MBSE, a well-known approach called Model Driven Architecture (MDA) can be used, which has been introduced by the Object Management Group (OMG). The views specified in MDA are (1) Computation Independent Model (CIM) to provide an understandable description of the system for end users, (2) Platform Independent Model (PIM) to define functionalities and display components of the system, (3) Platform Specific Model (PSM) to formulate interfaces and other technical specifications and (4) Platform Specific Implementation (PSI) to maintain a detailed presentation of code used for describing components within the system [14].

An example of how to generate interoperability throughout the whole engineering process has already been successfully overcome in the Smart Grid domain. The SGAM has been introduced in order to provide an environment that helps building Smart Grid systems [21]. In [19] the design and implementation of a tool called SGAM Toolbox is described, which ensures the applicability of the theoretical approach. By doing this, the SGAM Toolbox consists of three major parts:

- MDG Technology, which contains the specifications stated in the DSL and provides them for usage
- Model Templates, which support system engineers by providing a fully modelled example and giving information about specific problems
- Reference Data, that contain information about the matrix used in SGAM and make sure to integrate those information into the model

With the help of the SGAM Toolbox, several international projects have already been realized. Through years of use it has established itself as main technology driver to create Smart Grid systems. Adopting these successful concepts to the industrial area can bring the approach of RAMI 4.0 a major step forward.

3 Approach Taken for Transfer

As already mentioned, the goal behind this approach is to adopt the already established concepts of the SGAM Toolbox for the scope of RAMI 4.0. Although, in the Smart Grid domain, the process of generating energy and providing it to the customers is hierarchically structured. The energy flow passes through multiple zones including several elements, where information is exchanged only with adjacent elements over defined interfaces. This keeps a specific abstraction level, and therefore, allows the modeling of Smart Grid systems to remain structured and understandable, as required from the design principles "divide and conquer" as well as "separation of concerns". However, cross-linking in the Smart Factory is considerable more difficult due to versatile connections and dynamic changes of elements communicating within the network. Adopting the Smart Grid approach to Industry 4.0 including the integration of new domain-specific features and the problem of outcome validation is a big challenge. To deal with this complexity, a new methodology needs to be developed, where all industrial particularities are considered. A similar approach dealing with this issues using the IIRA is introduced in [18]. However, to create a DSL tailored to RAMI 4.0, a dynamic approach needs to be used, where adaptations of the concept may take place anywhere during the engineering process. The concepts introduced by the Agile Design Science Research Methodology (ADSRM) [3] are tailored to this problem. In Fig. 2, a visual representation of this methodology adapted to the development of the DSL for RAMI 4.0 is given.

In three phases the designing of a dynamic technology, like the creation of the DSL for RAMI 4.0, can take place. In the analysis phase, the domain is elaborated and requirements derived from the Case Study are specified. Afterwards, the development of the Process Model, the DSL and the Toolbox itself takes place during the implementation phase, resulting in an applicable yields model. The last step is the evaluation of the developed technology towards the problem domain. The big advantage of this method is the loose coupling between the single phases, which allow flexible interactions and therefore changes may take place in every phase without influencing the functionality of the whole process.

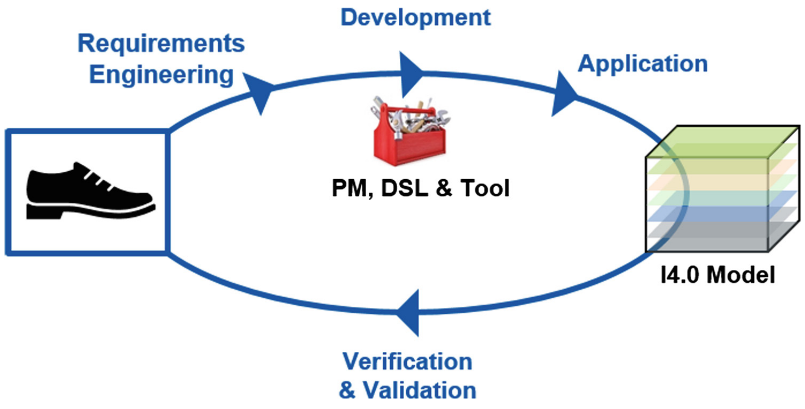


Fig. 2. Agile design science research methodology for RAMI 4.0

3.1 Case Study Design and Requirements

According to ADSRM, the first step is to draw up a suitable Case Study. In this case a typical use case concerning Industry 4.0 is presented. More precisely, this example makes use of a shoe manufacturing company that offers the creation of individual shoes to its customers. The manufacturer provides all tools used for customer interaction as well as the factories where the shoes are produced. The goal is to optimize production processes, therefore raw materials and supplier goods need to be available at the time they are used in production. On the other hand, the customer wants to create his individual pair of shoes out of his mind. The ideology of Industry 4.0 is the fully automated processing of the order and the consequent production of the shoes. Therefore, all machines should communicate with each other in order to find the optimal solution concerning resources. Firstly, to keep track of administrative and change efforts, the requirements that the system underlies are elaborated. Concerning the classification of non-functional requirements, the following five requirements have been specified:

1. Functionality: The system to be developed needs to contain all important aspects of Industry 4.0 to allow a detailed and complete description. To achieve this, the framework should support the creator of the system by using well known methods and without raising complexity or administration expenses.
2. Usability: Users may come in contact with Industry 4.0 based systems for the first time. Therefore, usage should be clear and supported by demonstration examples as well as automation tools.
3. Efficiency: After all, the framework is used to increase productivity, this means that resources should be kept low and time-consuming tasks should be avoided.

4. Reliability: The proper creation of a system could be a problem for first-time users. The consideration and prevention of incorrect statements needs to be part of the solution too.
5. Changeability: RAMI 4.0 and Industry 4.0 is in consistent change, therefore the framework should be adaptable to these changes. In addition, it should be possible to integrate user-specific solutions in order to react to proprietary implementations.

3.2 Process Model

To manage the creation of industrial models, a specific development process is needed. Technically this process is comparable with the model transformations used by MDA. Furthermore, the single steps of the process are described by the technical processes introduced by the ISO 15288, as depicted in Fig. 3.

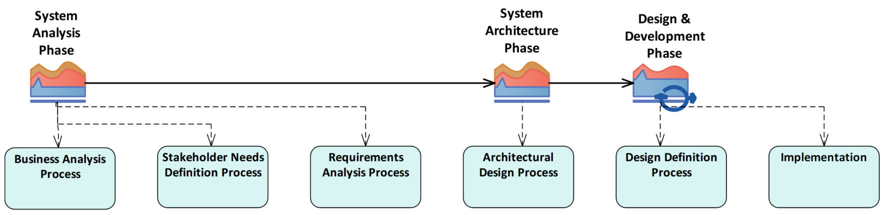

System Analysis Phase
System Architecture Phase
Design & Development Phase
Business Analysis Process
Stakeholder Needs Definition Process
Requirements Analysis Process
Architectural Design Process
Design Definition Process
Implementation

Fig. 3. Development process for RAMI 4.0 models

Firstly, the system is analyzed concerning its functionality and business requirements. This functional architecture should give an overview of the system and be understood by people not familiar with the domain but well known with business aspects. Therefore, the aim is to model the basic conditions the system should follow. The system analysis is the base for building a more detailed viewpoint, the system architecture. It describes the components of the system with their interfaces and connections. The type of connection and technology used to transfer data is the major part designed in this phase of the process. The last step is the detailed modeling of the single components specified by the system architecture. The so-called design of the system displays all elements included and helps dynamically integrating new ones. All in all, this development process deals with an uniform creation of Industry 4.0 models and makes sure that the different abstraction levels of the system are being kept.

3.3 Domain Specific Language

To design a DSL it is important to understand the application domain such as the physical world of Industry 4.0 and CPS. Resulting from this, the behavior and context of the physical domain could be analyzed in order to create a model of the real world. Semantics and structure of this model help defining the abstractions of the Metamodel, dependencies between physical and virtual world formulate the connections of its elements, according to [17]. The Metamodel representing RAMI 4.0 is composed by a conceptual architecture, constituted of the Unified Modeling Language (UML). It describes the conceptional aspects a language needs to contain to model a system based on Industry 4.0. By doing so, the Metamodel is structured in the six layers of RAMI 4.0. On each layer, design elements for describing a viewpoint on a system are provided. The Business Layer therefore consists of elements like business actors, business goals and business cases for representing the cooperation between two actors. With these elements desires of stakeholders can be formulated. High-level use cases are specified to realize business cases on the Function Layer in order to fulfill the defined requirements. Information objects, characterized by a specific data model standard, as well as the connection paths they are exchanged over are being modeled in the lower layers. The Integration Layer offers a representation of the Asset Administration Shell (AAS), a model of the digital twin every physical asset has. Those assets itself are being depicted in the same called Asset Layer.

As the Metamodel is a graphical representation of domain-specific elements and their interconnections, a language is designed for a detailed description of those. Similar to the concepts presented in [4], the conceptual architecture serves as a base to create a specific DSL. This language needs to be utilized throughout the whole development process, from designing the system followed by describing up to modeling it. Consisting of an UML profile, the DSL itself can be designed using well known methods provided by UML. The profile contains all elements previously elaborated from the physical world. Given by UML, the elements itself are consisting of a stereotype and a metaclass. The metaclass is representing the underlying model element where the stereotype is describing the element as it will be used in the DSL.

3.4 Toolbox Implementation

There are several software applications on the market tailored to systems development. Concerning its functionality to extend, the modeling tool Enterprise Architect (EA) developed by Sparx Systems [22] is suitable for providing an environment in order to architect and design Industry 4.0 based systems. To achieve this, the already given general modeling functionalities need to be extended by implementing the DSL. The result is an Add-In called the RAMI Toolbox[1]. The main part of this toolbox is the DSL described in the previous section. It consists of the UML profile and two other profiles for the utilization of a tool-set as well as a suitable UML diagram to describe an industrial model. Adapted from the SGAM Toolbox it also provides demonstration examples showing how to use RAMI 4.0. To make use of this DSL, EA needs to load it during its start-up process to provide a set of tools supporting the modeling of industrial systems.

[1] The RAMI Toolbox is publicly available for download at http://www.rami-toolbox.org/download.

4 Application of the Toolbox

4.1 Case Study Model

The Case Study[2] itself is created by using the development process described in Sect. 3.2. According to these considerations, the Business Layer contains three major actors, the customer, the manufacturer and the supplier. In the system analysis the goals of each actor are elaborated through requirements engineering. Those goals specify the boundary and rules the system should follow. To keep it simple, this scenario identifies one High Level Use Case (HLUC) "Create Custom Shoes" with the three previously mentioned actors. Out of the generic business model a more specific functional viewpoint can be created. The HLUC is decomposed into more detailed Primary Use Cases (PUCs). "Order Processing" or "Factory Maintenance" could be representatives of this kind. In the Function Layer, every Use Case has actors interacting with them. Figure 4 depicts the most detailed functionalities in the development process like forming supplier goods or assembling raw materials. By doing so, the single functions are represented as Use Cases with their related Actors. The resulting Logical Architecture builds the base for the real architecture of the system including all components and parts. The architectural solution is built referring to the results of the system analysis. The modeled processes need to be represented by physical components. Technological speaking, a model transformation introduced by MDA takes place by mapping Logical Actors to their physical components. In the Information and Communication Layer of RAMI 4.0, the interaction of these components is modelled based on the specifications coming from OPC Unified Architecture (OPC UA). The first step of the system architect is to find out which kind of information is exchanged between the elements. This process is followed by designing and specifying the communication paths and interfaces of which the information is sent. During this phase the components are seen as Black-boxes and only those needed for interaction are described.

The decomposition of the components itself takes place on the Integration and Asset Layer. On these layers, the elements are described as physical units like they are in the real world. The Integration Layer generates a Digital Twin out of the physical units. This means, one AAS containing all information and data as well as safety and security aspects is created for each asset or a group of assets working together. Furthermore, the Integration Layer has to deal with Human Machine Interfaces (HMIs) in order to access the needed data. Technologies like Near Field Communication (NFC), Bluetooth, Barcodes and USB find its place on this layer.

4.2 Findings

Although this generic example enabled the evaluation on a superficial perspective, the used concepts worked fine in general. However, the next iteration step

[2] A click-through model is available at http://www.rami-toolbox.org/UseCaseShoes.

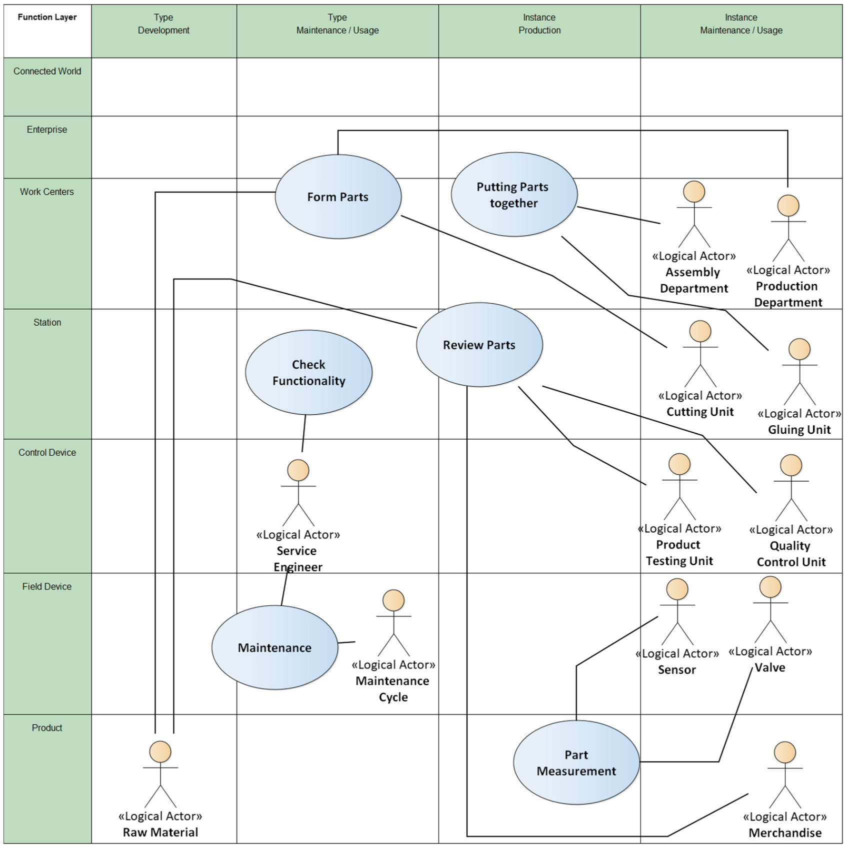

Function Layer
Type Development
Type Maintenance / Usage
Instance Production
Instance Maintenance / Usage
Connected World
Enterprise
Work Centers
Station
Control Device
Field Device
Product
Form Parts
Putting Parts together
Review Parts
Check Functionality
Maintenance
Part Measurement
«Logical Actor» Assembly Department
«Logical Actor» Production Department
«Logical Actor» Cutting Unit
«Logical Actor» Gluing Unit
«Logical Actor» Service Engineer
«Logical Actor» Product Testing Unit
«Logical Actor» Quality Control Unit
«Logical Actor» Maintenance Cycle
«Logical Actor» Sensor
«Logical Actor» Valve
«Logical Actor» Raw Material
«Logical Actor» Merchandise

Fig. 4. RAMI function layer diagram of the case study

of ADSRM needs to deal with more detailed problems. For example, it was shown that some specifications of DIN 91345 need to be refined or adapted to suit for every domain included. Furthermore, an extension of the process model with familiar standards results in the definition of a more detailed development process. Hence, the standard ISO/IEC 42010 [13] provides a formalization of an architecture framework that may well fit for RAMI 4.0, for example deriving viewpoints and views for each layer. In the same step, the concepts of the Unified Architecture Framework (UAF) standard are elaborated on their suitability for the RAMI Toolbox. The general approach and its enhancements need to be validated by an external domain stakeholder providing a more sophisticated case study in the last step.

5 Conclusions and Future Work

An example of how to deal with modeling and analyzing complex energy systems is the SGAM Toolbox. The already established technology for developing Smart

Grid systems has all functionalities needed for system engineering. Due to the similarities between energy and industry domains, the concepts of the SGAM Toolbox [20] may be applicable to Industry 4.0. In this paper, two major concepts have been tested on applicability in the RAMI Toolbox. First, the modeling of Use Cases on basis of an existing reference architecture has been approved by the shoe manufacturing industry example. Although modeling took only place on a superficial perspective, existing concepts and technologies seem to work in the industrial domain as well. The domain specific representation and visualization of components as entry point for discussions or building a common understanding is realized by the DSL. The findings of this paper build a base for the future work of the authors. With the results mentioned above, an application of RAMI 4.0 has been developed for the first time. Now, the results need to be applied to a more sophisticated case study in order to adapt the concept to upcoming domain-specific requirements. In future work, the integration of well-known standards for architectures, processes or industrial specifications and the development of new features may lead the path towards establishing this approach to become a widely used technology for building Industry 4.0-based architectures.

Acknowledgments. The support for valuable contributions of *LieberLieber Software GmbH* and *successfactory consulting group* is gratefully acknowledged.

References

1. Bitkom, VDMA, ZVEI: Umsetzungsstrategie Industrie 4.0, Ergebnisbericht der Plattform Industrie 4.0. ZVEI (2015)
2. Boardman, J., Sauser, B.: System of systems-the meaning of. In: IEEE/SMC International Conference on System of Systems Engineering, 2006, pp. 6–pp. IEEE (2006)
3. Conboy, K., Gleasure, R., Cullina, E.: Agile design science research. In: International Conference on Design Science Research in Information Systems, pp. 168–180. Springer (2015)
4. Dänekas, C., Neureiter, C., Rohjans, S., Uslar, M., Engel, D.: Towards a model-driven-architecture process for Smart Grid projects. In: Digital enterprise design & management, pp. 47–58. Springer (2014)
5. DIN SPEC: 91345: 2016-04. Reference Architecture Model Industrie 4.0 (2016)
6. Drath, R., Horch, A.: Industrie 4.0: Hit or hype? IEEE Ind. Electron. Mag. **8**(2), 56–58 (2014)
7. Hankel, M., Rexroth, B.: The Reference Architectural Model Industrie 4.0 (RAMI 4.0). ZVEI (2015)
8. Hermann, M., Pentek, T., Otto, B.: Design principles for Industrie 4.0 scenarios. In: 49th Hawaii International Conference System Sciences (HICSS), pp. 3928–3937. IEEE (2016)
9. Industrial Internet Consortium and Plattform Industrie 4.0: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper (2017)
10. International Electrotechnical Commission: IEC 61512: Batch control (2001)
11. International Electrotechnical Commission: IEC 62264: Enterprise-control system integration (2016)

12. International Electrotechnical Commission: IEC 62890: Life-cycle management for systems and products used in industrial-process measurement, control and automation (2016)
13. International Organization for Standardization: ISO/IEC/IEEE 42010: Systems and software engineering – architecture description (2011)
14. Kempa, M., Mann, Z.A.: Model driven architecture. Informatik-Spektrum **28**(4), 298–302 (2005)
15. Lightsey, B.: Systems engineering fundamentals. Technical report, DTIC Document (2001)
16. Lin, S.W., Miller, B., Durand, J., Joshi, R., Didier, P., Chigani, A., Torenbeek, R., Duggal, D., Martin, R., Bleakley, G., et al.: Industrial Internet Reference Architecture (IIRA). Industrial Internet Consortium (IIC), Technical report (2015)
17. Mezhuyev, V., Samet, R.: Geometrical meta-metamodel for cyber-physical modelling. In: 2013 International Conference on Cyberworlds (CW), pp. 89–93. IEEE (2013)
18. Morkevicius, A., Bisikirskiene, L., Bleakley, G.: Using a systems of systems modeling approach for developing Industrial Internet of Things applications. In: 2017 12th System of Systems Engineering Conference (SoSE), pp. 1–6. IEEE (2017)
19. Neureiter, C.: Introduction to the SGAM Toolbox. Technical report, Josef Ressel Center for User-Centric Smart Grid Privacy, Security and Control, Salzburg University of Applied Sciences (2013)
20. Neureiter, C., Engel, D., Trefke, J., Santodomingo, R., Rohjans, S., Uslar, M.: Towards consistent Smart Grid architecture tool support: From Use Cases to Visualization. In: 2014 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), pp. 1–6. IEEE (2014)
21. Neureiter, C., Uslar, M., Engel, D., Lastro, G.: A standards-based approach for domain specific modelling of Smart Grid system architectures. In: 2016 11th System of Systems Engineering Conference (SoSE), pp. 1–6. IEEE (2016)
22. Sparks, G.: Enterprise Architect user guide (2009)
23. Uhlemann, T.H.J., Lehmann, C., Steinhilper, R.: The digital twin: Realizing the cyber-physical production system for Industry 4.0. Procedia Cirp **61**, 335–340 (2017)
24. Van Deursen, A., Klint, P., Visser, J., et al.: Domain-specific languages: an annotated bibliography. Sigplan Not. **35**(6), 26–36 (2000)
25. Weyrich, M., Ebert, C.: Reference architectures for the Internet of Things. IEEE Softw. **33**(1), 112–116 (2016)

Assessing the Maturity of Interface Design

Alan Guegan[1](✉) and Aymeric Bonnaud[2]

[1] Sirehna, 5 rue de l'Halbrane, 44340 Bouguenais, France
alan.guegan@sirehna.com
[2] Naval Group, 5 rue de l'Halbrane, 44340 Bouguenais, France
aymeric.bonnaud@naval-group.com

Abstract. It is widely accepted that the way the interfaces between subsystems are designed is a major aspect of system architecture. The task of designing interfaces is made difficult by the technical diversity of subsystems, of interfaces, of functional requirements and integration constraints. Change management processes have long been implemented by the industry to monitor and control interface design (see, e.g. Eckert 2009). In this paper, change request data from several projects completed by Naval Group is analyzed. The Change Generation Index is introduced and a heuristic formula is proposed to link the maturity of interface design with change request generation. This approach comes as a complement to existing results on change propagation patterns within large systems. A promising parallel is established between the design process of a large system and learning processes well known to the social sciences community.

1 Introduction

The complexity of a system is higher when the number of interactions is large with respect to the number of subsystems within the system. Often, when the design of a subsystem is refined or redefined, the subsequent impacts extend well beyond the subsystem itself. It is thus necessary to perform design iterations to take into account unexpected impacts and relax towards a consistent system design.

Due to the large number and the diversity of interactions between subsystems, information about the design is generated *as the design is being worked out.* The design process is not linear; rather, the design follows a "maturation" process distributed over the many subparts of the system. In these conditions it is difficult to assess the maturity of the design. Any part in the system can call for re-work, at virtually any time, due to unforeseen interactions between this part and the rest of the system.

In order to assess the maturity of the design, we propose to view the design process as a *learning experience* in which the design teams accumulate over time a body of knowledge about subsystems, the interactions between subsystems, and the system as a whole.

The maturity of each subsystem can be inferred from subsystem design artefacts such as plans, models, technical reports. The maturity of the design of the system as a whole is assessed with the help of design reviews and technical assessments by experts. From our experience, it is more difficult to assess the maturity of the interfaces between subsystems, which makes most of the system's architecture.

© Springer Nature Switzerland AG 2019
E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 56–66, 2019.
https://doi.org/10.1007/978-3-030-04209-7_5

In this paper we propose to use change requests as a marker for the maturity of the design of interfaces within the system. The analysis of actual data from industrial programmes involving hardware, software, or both shows a clear, reproducible trend for the number of design changes identified over the course of a project. We propose a theoretical explanation for this trend and discuss the agreement with the data. Eventually, we discuss how KPIs can be derived to monitor interface design maturity during the development of complex systems.

2 Design Change Requests Data: Main Characteristics

2.1 Available Data and Characteristic Patterns

In iterative design processes it is common practice to trace *design change requests*. A design change request is issued when the design team has identified that the design of a subsystem should be changed to accommodate constraints from another subsystem. For instance, if system A proves to be larger than expected, it might be necessary to move or reconfigure neighboring system B; if the command and control network is replaced by an ethernet network, it might be necessary to change system C and add a network card to it.

Design change requests are generated throughout the development process. Figure 1 shows the number of design change requests that have been generated weekly over the course of six projects delivered by Naval Group. These projects consist in the design, manufacturing, testing and delivery of five naval systems that range from pure software or mechanical devices to process systems involving embedded software; project number 6 is a software project. The sizes of the projects range from a hundred thousand euros to several millions, with design teams staffed accordingly. The processes vary from pure V-cycle to agile development.

Change request generation varies a lot from one week to the next but it exhibits a consistent overall pattern. Although extremely different in size and scopes, all projects show the same trends:

- Few design changes are generated during the earlier and later weeks of each project.
- A maximum is reached at half of the total project duration, roughly.
- Some projects exhibit deviations from a smooth "bump" shape, like spikes in project 5, uneven distribution over time in project 3, long-lasting change request generation in project 4 or even a holiday-season dip around week 295 in project 6.
- On the whole, the curves of change request generation exhibits the shape of a roughly symmetrical "bump" that extends over the duration of the project.

We propose to call the number of design change requests generated each week the Change Generation Index (CGI), and investigate its general trend across projects. A discussion about the measure of the CGI is provided in Sect. 2.2 and a short theory is proposed in Sect. 3 to explain the trend observed in change request generation over time.

2.2 Potential and Limitations of an Analysis Based on the CGI

The benefits that can be expected from analyzing the Change Generation Index depend on the quality of the available data. The variability that can be expected in CGI data is of two types: 1/ variability in time, 2/ variability across teams.

From our experience, once the change request management process has been implemented within a given project variability is low except for characteristic events such as the ones discussed in Sect. 2.1, Fig. 1. Still, in the early phases of most design projects few change requests are generated. After a short investigation we have been able to trace this to two main reasons:

– early design often focuses on subsystem design driven by rough assumptions on the interfaces. The design of each subsystem is undertaken with these assumptions as input requirements; the assumptions are not questioned until the design has reached enough maturity to justify that interface requirements are refined.
– The first phases of the design process are characterized by intense iterations and the design teams would be slowed down if they traced each and every change request. On most of the projects we have analyzed the change request management process has been implemented a little later in the design process, when the main assumptions about system architecture had been validated.

Variability may also be observed across teams within a project, or across different projects. The latter has little impact on the analysis of the CGI, as Sect. 3 will show. The analysis focuses on the date when the CGI reaches a maximum and on the characteristic time it takes the CGI to decrease. These two parameters are independent from the change request tracing policy of a project. Also, the CGI is a statistical index that smooths out variability across teams, very much like the group learning curves

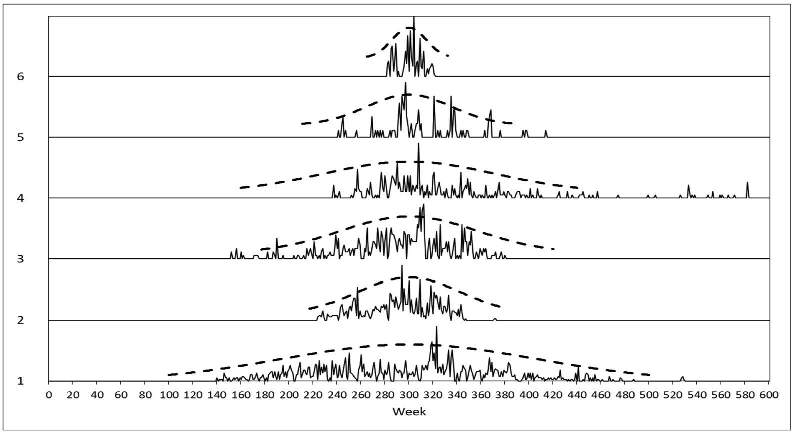


Fig. 1. Number of change requests registered each week during the development process of six hardware/software/mixed systems (solid lines). All of these curves exhibit a similar shape ("bump" shape outlined by dashed lines), with project-specific deviations discussed in more detail in Sect. 4.

addressed in Gallistel et al. (2004) smooth out individual differences in the learning rates and performances. As a consequence, the unavoidable variability between projects has not proven a difficulty in the analysis of the CGI.

It is useful to place the present paper in the perspective of other change request management processes. Conventional change request management usually focuses on the absolute number of change requests that has been generated over a given period of time, the number of changes that have been implemented, or the average time needed to implement change requests. In the present approach the focus is shifted to the *rate* at which change requests have been *generated*, regardless of whether the design changes have been actually *implemented* or not, and regardless of how many still need to be implemented or how many have been implemented in total.

It is not the first time the importance of tracking the rate at which changes are generated is acknowledged (see, e.g. Giffin (2007)). This rate was seen as a consequence of project staffing and project events rather than an indicator of the learning process associated with design activities, as we propose here.

Some key performance indicators have been introduced in the past, for instance the Change Rejection Index (Alabdulkareem et al. 2013) that reflects the rate at which change requests are *rejected* by a subsystem (not implemented), or the Change Propagation Index (Giffin et al. 2009) that reflects the likelihood of a subsystem generating new changes after implementing a modification. These papers focus on change propagation or rejection with a view to assign change requests to those subsystems that are more prone to evolution, thus increasing the efficiency of the change management process. We expect that shifting the focus onto the overall pattern of change requests generation will yield complementary and similarly useful information about the design process.

3 Design Change Request Generation as a Heuristic Measure of Interface Design Maturity

3.1 Change Requests and Interface Design Maturity

In project management, change requests are often viewed as a risk for the project since they incur unforeseen delays and costs. In this view, change requests should be avoided to minimize risks. From a design engineer's point of view, change requests are a sign that, at some point in the design process, *technical decisions being made conflict with decisions made earlier*. A change request is neither good or bad; it might lead to the implementation of an actual change in the design, or it might be rejected owing to unaffordable costs or delays. In this sense, a change request is the indication that the design process has unveiled previously unsuspected interactions between technical decisions made at different times; it is the trace of knowledge being acquired.

In most learning processes – and more specifically, group-learning processes - the amount of knowledge accumulated over time is reasonably well represented by a sigmoïd, or S-curve. A sigmoïd function is a bounded, differentiable, real function that is defined for all real input values and has a non-zero derivative at each point (see e.g., Han and Morag 1995). Gallistel et al. (2004) provide one such example of a group-learning curve (Fig. 1 in cited paper). We chose to use the following sigmoïd function as the reference in our analyses:

$$S(t) = (1 + \mathrm{erf}(t))/2,$$

where erf is the error function and t is time. Function S has a number of characteristics that make it a convenient reference for analyzing the knowledge generated during design processes:

- S goes from 0 to 1, which is consistent with the knowledge about a system's interfaces increasing from 0% to 100% over the course of the design process,
- The derivative of S is a gaussian function,
- S can be fitted to the characteristics of any individual project, by the affine change of variables T = a(t + b).

Function S is displayed in Fig. 2 (solid line). Early in the design process, little is known about the system's interfaces and teams have few opportunities to assess how the parts within the system work together: the learning process is slow. Halfway through the design process, a lot more is known about the system, its subparts, and the interactions between them. Many decisions made during this phase conflict with previous work and induce change requests: design teams generate a lot of knowledge about the interfaces and the S-curve has a steeper slope. By the end of the design process, little remains to be learnt about the interfaces within the system and the S-curve levels off.

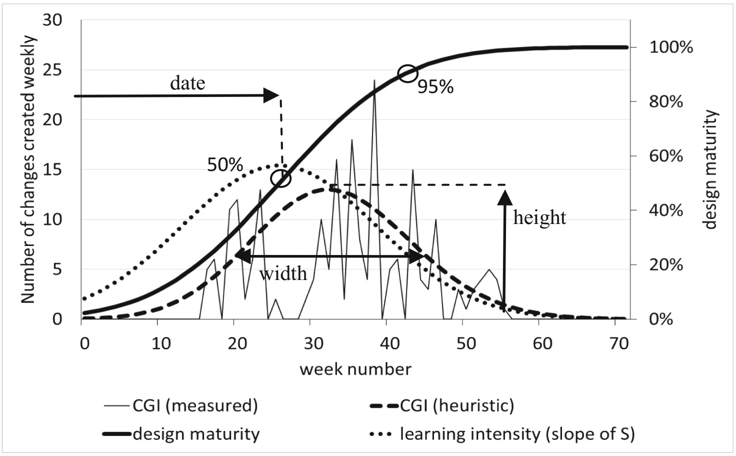


Fig. 2. Number of change requests registered each week for project 6, Fig. 1. The thick solid line represents the learning curve (axis on the right), the dotted line is its derivative, corresponding to the intensity of the learning (gaussian curve). We postulate that the Change Generation Index scales like the product of these two quantities (dashed line, axis on the left). The parameters for the CGI have been fitted on the actual data (thin, solid line). The learning curve can then be used to infer a "maturity" indicator, here pictured at 50% and 95% maturity (circles).

The learning process we describe here is actually stepwise at small scales and continuous (represented by function S) at a global scale. Each time a change request is issued, it is a sign that a piece of knowledge has been acquired that contributes to the global learning curve, very much like the stepwise learning of several individuals within a group contribute to the sigmoïd-shaped knowledge accumulation by the group as a whole (Galistel et al. 2004). The knowledge accumulated weekly is the derivative (slope) of the learning curve: it is the intensity of knowledge acquisition (dotted line in Fig. 2).

As stated in Sect. 3.1, change requests are generated when technical decisions *being made* conflict with technical decisions *made earlier*. Following this interpretation, it is reasonable to assume that the number of change requests being generated at any time in the project scales with both the amount of knowledge *being acquired* and the knowledge *already accumulated*:

$$\mathrm{CGI}\ (\mathrm{t}) \sim \mathrm{Knowledge}\ (\mathrm{t})\ \mathrm{x}\ \mathrm{d(Knowledge)/dt}\ (\mathrm{t})\,.$$

With the assumption that Knowledge (t) scales like S, the Change Generation Index scales like:

$$\mathrm{CGI}\ (\mathrm{t}) \sim \mathrm{e}^{(-\mathrm{t}^2)}\mathrm{S}(\mathrm{t}).$$

The CGI as defined by the above formula is displayed in Fig. 2 (dashed line). The “heuristic” CGI curve resembles the actual data in Fig. 1. The section that follows investigates the relationship between the heuristic and the actual data and shows that the heuristic shows a good match with the actual data.

3.2 Procedure to Evaluate the Maturity of Interface Design

The heuristic CGI devised in Sect. 3.2 is characterized by two parameters that are in direct relationship with the parameters of the S-curve:

– Parameter 1: the date when the maximum is reached. This date is linked with the date when the S-curve reaches its inflexion point, that is, the date when half the total amount of knowledge that will be accumulated eventually has already been acquired.
– Parameter 2: the “width” of the bump. This stems from the steepness of the S-curve, or the standard deviation of the Gauss curve representing learning intensity.

The interpretation of these two parameters is straightforward: a faster design process will lead to a narrower CGI curve, and a design process developed later in time will lead to a CGI curve centered on a later date.

To match the actual data, it is necessary to introduce the “height” of the CGI curve as a third parameter. This parameter is the maximum number of changes that are generated weekly during the design process. It may vary depending on the complexity of the project: complex projects should generate more change requests in absolute value than simpler ones. This parameter acts as a simple “scaling” of the CGI curve along the y-axis, and it won't change the shape of the curve that is given by the date and width parameters.

In what follows, parameters 1, 2, and 3 will be called the "date", "width", and "height" parameters. Practically, the date and width parameters are implemented by replacing variable t by u = a(t + b), with a being the "width" parameter and b being the "date" parameter. The "height" parameter comes as a multiplying coefficient applied to $e^{\left(-t^2\right)}S(t)$.

We propose to apply the following procedure to infer project maturity from the *measure* of the Change Generation Index:

- 1/ plot the actual number of change requests generated each week versus time,
- 2/ plot the heuristic CGI curve, and match the "date" parameter with the date at which the maximum number of change requests has been generated according to the measure,
- 3/ tune the "height" parameter so as to match the maximum number of change requests generated weekly,
- 4/ tune the "width" parameter to fit the measured data,
- 5/ adjust parameters so that the heuristic CGI curve superimposes the actual data in the best possible way,
- 6/ plot function S with the "date" and "width" parameters found at 5/ and use it the assess the maturity of the design of the interfaces.

In this procedure parameter identification is done "by hand", which leaves room to interpretation. However, we have found it to yield consistent results across projects, and to be more resilient to specific (meaningful) deviations in the data than an automated identification algorithm would.

An example is shown in Fig. 2. The development process took approximately 10 months (between weeks 15 and 55); in average, up to 13 change requests were created each week, with a maximum around week 30. Interface design maturity reached 50% around week 25, and 95% at week 45.

Section 4 provides several examples of actual data and what conclusions can be drawn by using this approach.

4 Examples

4.1 The Perfect Project

Figure 3 shows the CGI measured from the change requests database of an industrial project achieved by Naval Group. A number of 1029 change requests were registered over the approximately 9 years of the project. The measured CGI exhibits a general trend similar to the one observed in Fig. 1.

The steeper increase in change request generation in the beginning of the project is a sign that the traceability of change requests has improved progressively between weeks 90 and 120 (see Sect. 2.2). By the time the CGI reaches a maximum, change requests are traced thoroughly and the curve follows the theory nicely.

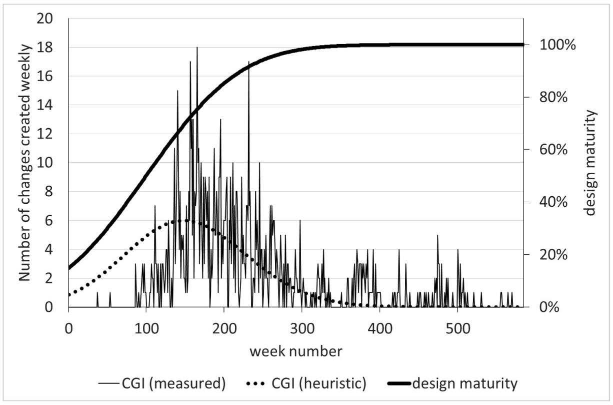


Fig. 3. Change Generation Index as measured for a naval system (thin, jagged line). Heuristic parameters have been tuned to fit the measured data (fitting curve in dotted line), with rather good agreement between weeks 100 and 400.

The project that is displayed in Fig. 3 was a success. Commissioning started around week 280. The learning curve that we have derived from the interpolated CGI indicates a 97% maturity by that time. The large majority of the projects that we have analyzed show a similarly good agreement with the theory, with a maturity comprised between 95% and 98% at comissioning. Some of them, however, depart from the theory quite significantly. Sections 4.2 and 4.3 show two such examples of "unorthodox" behaviours.

4.2 Late Overshoot: End of Design Declared Prematurely

Figure 4 shows the CGI measured from the change requests database of an industrial project similar to the previous one. A number of 925 change requests were registered over 3 years. The measured CGI exhibits a general trend similar to the one observed in Figs. 1 and 3, except for the significant overshoot that can be observed between weeks 180 and 200.

The project was significantly shorter (by a factor of 3) than the one in Sect. 4.1; commissioning started at week 130. A possible explanation for the surge in change requests at week 180 is that the design process missed a number of technical issues that were identified only when the system had been built and tested. In this sense, the change requests generated in excess between weeks 180 and 200 are the visible part of the "technical debt" accumulated over the design process. They are also a – late but backed up by data - sign for project management that the maturity curve shall be questioned (the learning curve depicted in Fig. 4 does not reflect the actual learning accumulated by the design team) and that additional work needs to be done before the system can be declared good for service.

The data in Fig. 4 provides signs that the design process, in the way it was completed on this specific project, needs improvement, to ensure that no design issues are overlooked in future projects.

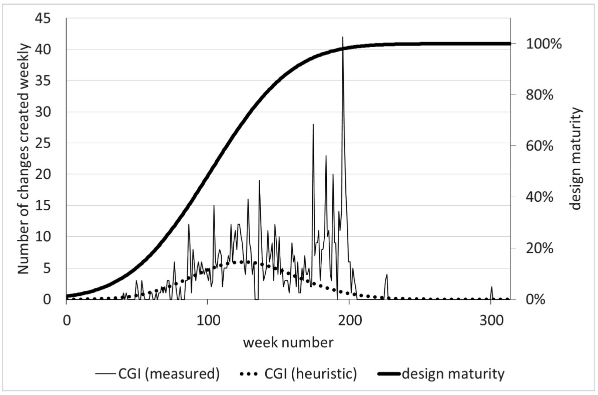


Fig. 4. Change Generation Index as measured for a naval system (thin, jagged line). Heuristic parameters have been tuned to fit the measured data (fitting curve in dotted line), with rather good agreement between weeks 50 and 180. The maturity of the design (thick, solid line), as inferred from the parameters of the heuristic CGI, shall be reconsidered in view of the "technical debt" that shows in the excessive number of change requests observed over the last 20 weeks of the project.

4.3 The Flat Project: A Volatile Environment

Figure 5 shows the CGI measured from the change requests database of an embedded software. A number of 152 change requests were registered over a little more than a year. The measured CGI is "flat": there were between 2 and 4 change requests each week during 70 weeks, with no sign of a peak or decrease in the number of change requests over time.

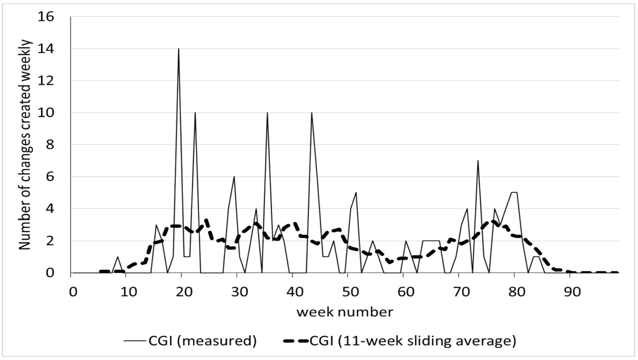


Fig. 5. Change Generation Index as measured during the development of an embedded software (thin, jagged line). A sliding average over 11 weeks is displayed as a thick dashed line, to highlight the flatness of the CGI over time.

This project lasted much longer than originally planned, due to unceasing changes in the specification from the client entity. Detailed software design was initiated early, at a time when the system supporting the software had not yet been designed in enough detail. This resulted in changes originating *outside* the software design process, making it impossible to “drain” the subject of designing a software that fits the customer’s needs. In this case, a flat CGI curve is the sign of a Sisyphean project constrained by an endlessly moving environment.

5 Discussion

The present work suggests that change requests might be used as a marker for design maturity. The heuristic that has been introduced in Sect. 3.2 reproduces the trends observed in change request generation, and preliminary analysis shows that maturity as inferred from parameter identification is in good agreement with project milestones (see 4.1: 97% maturity corresponds to system commissioning).

If confirmed, the approach would provide a way to assess the maturity of interface design based on data, which would be an invaluable complement to expert-based assessment. The scarce data in the first phases of the design does not allow for precise maturity assessment, yet the “peak change” can be identified with little ambiguity – indicating a 70% maturity milestone – and the remaining 30% maturation can be evaluated and tracked with reasonable precision with the CGI.

Further work is needed to assess the statistical robustness of the approach. Still, we believe that this work is a new confirmation that change request management is an essential part of the design processes that shall be applied to complex systems. Section 2.2 raises the question of when during the development process design changes shall be traced. Change management is often viewed as a costly process, at least in the first stages of the design. It might be of interest to establish more informal change management very early, to dispose of change request data as early as possible to the benefit of design maturity assessment.

The heuristic we propose also offers an opportunity to diagnose project weaknesses. In Sects. 4.2 and 4.3, the diagnosis is performed a posteriori to provide return of experience on the development process. It is possible to identify, if not quantify, the technical debt accumulated during the early stages of the design. A late surge in change requests is the telltale sign of an incomplete system design. It is also possible to identify projects that have suffered from inefficient learning.

The most promising aspect of the present work lies in the fact that we explicitly viewed the design process as a *learning process*. This has implications on the metrics associated with the design process - the CGI is one such metric. By pushing this paradigm further, it could also have consequences on the way complex systems are designed, or on the tools that are being used to develop complex systems. If the design team is *a group in the process of learning something by itself*, then maybe education sciences can bring some interesting methods, best practices and tools to improve and organize the learning.

References

Alabdulkareem, A., Alfaris, A., Sakhrani, V., Alsaati, A., de Weck, O.: The multidimensional hierarchically integrated framework for modeling complex engineering systems. In: CSD&M (2013)

Eckert, C., de Weck, O., Keller, R., John Clarkson, P.: Engineering change: drivers, sources, and approaches in industry. In: Proceedings of ICED 2009 (2009)

Gallistel, C.R., Fairhurst, S., Balsam, P.: The learning curve: Implications of a quantitative analysis. PNAS **101**(36), 13124–13131 (2004)

Giffin, M.L.: Change Propagation in Large Technical Systems. Ph.D. thesis, MIT (2007)

Giffin, M.L., de Weck, O., Bounova, G., Keller, R., Eckert, C., John Clarkson, P.: Change propagation analysis in complex technical systems. J. Mech. Des. **131** (2009)

Han, J., Morag, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: From Natural to Artificial Neural Computation, pp. 195–201 (1995)

Tracking Dynamics in Concurrent Digital Twins

Michael Borth and Emile van Gerwen(✉)

ESI, High Tech Campus 25, 5656 AE Eindhoven, The Netherlands
{Michael.Borth,Emile.vanGerwen}@tno.nl

Abstract. The availability of machine-generated data for the management of complex systems enables run-time technologies for diagnosis, predictive maintenance, process control, etc. that find their apex in digital twins. Such model-based replica of cyber-physical assets represent system elements and their behavior within their environment, which is often dynamic. These dynamics of a system's environment can render the underlying model unfit w.r.t. the changing reality and thus cripple the whole approach. We provide the means to detect such a transgression of the operational space of digital twins and similar technologies using a novel combination of probability-of-findings calculations with established process control methods and localize necessary updates to ensure efficient model maintenance.

1 Introduction

Machine-generated data, i.e., data that was produced entirely by machines, e.g., from sensor readings [1], became the backbone of many industrial and societal development. It is mission-critical for Smart Buildings, Industry 4.0, the Internet-of-Things (IoT), and Autonomous Driving [2], but also enables system stakeholders to address business concerns like total-costs-of-ownership with novel applications for, e.g., process control, diagnosis, and predictive maintenance that are driven by data analytics. These means are realized as digital twins in their most versatile and comprehensive form. Digital twins, which were identified by Gartner as one of the current top strategic technology trends [3], are digital replica of assets or systems together with their processes. They are based on models of the knowledge of domain experts as well as the real-time data collected from the systems and their environments. As such, they are subject to real world dynamics that change what the twins are about. We address these dynamics and their consequences for the digital twin in this article and provide a novel approach to detect and cope with them.

2 Digital Twins for Process Control and Analysis

Digital Twins, as originally defined by Grieves around 2001–2002 (see the newer [4]) provide a virtual representation of operational systems or other assets that aims at tightening the loop between design and execution. One of their strengths is the explicit use of expectations w.r.t. a system's behavior according to both domain engineering knowledge and to analysis models derived from data in comparisons with observations about the actual behavior (Fig. 1). The term ‘behavior’ has a wide interpretation in this context: for automated process control, it is typically understood according to the functional specifications of a system’s performance, while diagnosis includes expectations about reliability, mean-time between failures, etc., as does prognosis, especially for the purpose of predictive maintenance.

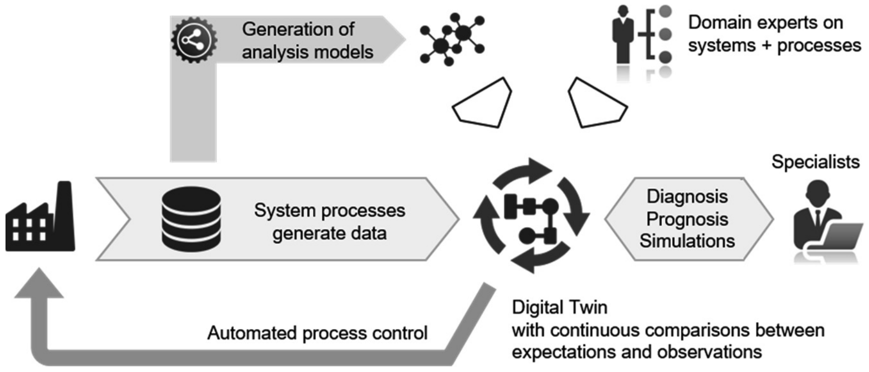

Generation of analysis models
Domain experts on systems + processes
Specialists
System processes generate data
Diagnosis
Prognosis
Simulations
Automated process control
Digital Twin
with continuous comparisons between expectations and observations

Fig. 1. Workflows for digital twins

Simpler twins offer primarily insights into the operations of the systems as well as in the larger interconnected systems-of-systems such as a manufacturing plant that form the environment of the system. Oracle, listing those as ‘virtual twins’ in [5], points out that they go beyond simplistic documents enumerating observed and desired values, but reserves the inclusion of analytics models built using a variety of techniques for ‘predictive twins’. The respective analysis models are typically based on data science, even though domain application experts, system architects and engineers, and data scientists often need to pool their expertise for success.

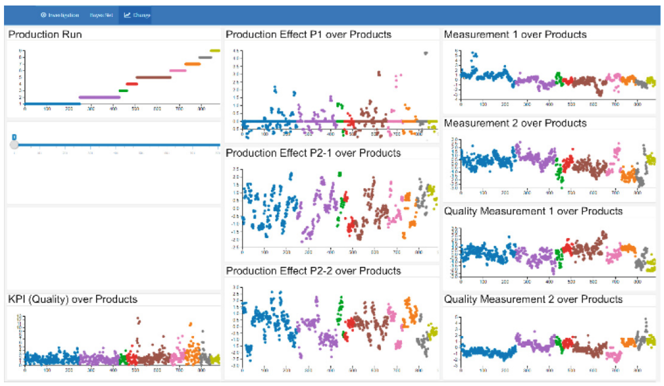
Once established, a digital twin typically serves various purposes. In cooperation with our industrial partners, we normally aim for an operational use that has a direct positive impact on total cost of ownership (TCO), e.g., energy savings with smart lighting controls for buildings, in addition to one or two service purposes, especially model-based diagnosis and prognosis. Furthermore, we strive to capture the causal relationships between systems, their components and functionality, and their state (especially failure modes and effects). As Pearl laid out in [6], this allows to investigate interventions, i.e., changes to the system, in a systematic manner. We described our use of this approach in [7].

2.1 Dynamics: A Challenge for Concurrent Digital Twins

Both virtual and predictive digital twins start out with a set of assumptions about the status quo of the systems they mirror: It is typical that they model the system as realized, using design information among other sources, and integrate insights gained from data analysis over an initial time period, typically one that consists of smooth operations – a so called happy flow – that illustrates which observations are expected if the system works well.

Seen as that, the digital twin is stationary: It mirrors the system in accordance to the knowledge and observations that were available at one point in time. Therefore, their status quo becomes a status quo ante, literally 'the state in which before', meaning here the state of affairs that existed previously – i.e., before the system and its environment changed.

Such change is ubiquitous. Even rather unassuming and mostly mechanical systems like wind turbines, which are monitored to prevent prohibitively expensive damage in case of a catastrophic failure [8], experience differences in the viscosity of lubricants or the stiffness of welded joints based on temperature, and thus season. These differences affect, e.g., the signatures and transfer of vibrations, a prime indicator of bearing issues among other items. The experts who build the digital twin might take that into account to a certain extent, but most likely there is no data and possibly even a lack of understanding for extreme situations, which are thus not adequately covered by the digital twin. The operation of a concurrent digital twin with this limitation will therefore lead to warnings that the condition of the system is degrading or that there is a failure, as the observations are abnormal. This, however, is a false positive, given that the observations can be explained by the impact of the environment. More complex examples illustrate that the dynamics systems face, and thus their digital twins as well, cannot always be foreseen and thus 'modeled in', as one might argue in the example above. Industrial production, realized with systems-of-systems typically orchestrated by a production management system [9], becomes, e.g., more and more flexible, up to the point of the so-called 'lot-size one'. Here, the number of items manufactured in a single production run is one, meaning that each item is made to order, following a unique process. This trend, which is made possible by the concepts and technologies covered under the term Industry 4.0 [10], leads to production processes, settings, and parameters that were unknown in the conception phase of the factory as market demands and insights into process improvements result in constant change.


Production Run
Production Effect P1 over Products
Measurement 1 over Products
Production Effect P2-1 over Products
Measurement 2 over Products
Quality Measurement 1 over Products
Production Effect P2-2 over Products
KPI (Quality) over Products
Quality Measurement 2 over Products

Dashboard showing data from 9 production runs producing ca. 900 products.

The product quality varies, but a correlation between the quality and observable effects of the two major production steps (middle) is difficult due to the variation that stems from diverse impact factors together with changes to the production settings between runs.

Section 5 describes the use case behind this data.

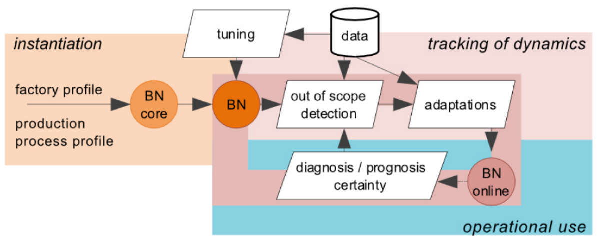
Fig. 2. Dynamics within flexible production setups

In our work, we investigated the range of dynamics in production process settings for flexible production lines and their impact to key performance indications regarding the product quality (see Sect. 5). As Fig. 2 illustrates, there are both stable and unstable process steps observable within individual production runs for specific products, but more importantly, a process setting that causes quality issues for one product can be perfectly fine for another. This is mainly due to the grade of orchestration, as settings need to fit to each other over interlocking control loops – a highly complex matter that leads even to expensive trial and error during setup.

2.2 Safeguarding the Digital Twin's Operations

Given the dynamics that we and others encounter, we see the need to safeguard a concurrent digital twin's operation against them: As changes in the real world will lead to a twin's assessment that new observations, which deviate from expectations, indicate problems, we require mechanisms that separate real issues from underlying dynamics in the environment in order to achieve foremost long-term operations with a constantly low rate of false alarms, but also the insight that the modelled reality changed to an extend that a maintenance operation on the digital twin is both needed and warranted.

We call the process for this endeavor 'tracking of dynamics'. As shown in Fig. 3, it consists of a detection component that confirms whether a computational model of a digital twin is used within or out of its scope or operational space (detailed in Sect. 3) and adaptation mechanisms for maintaining the model, i.e., keeping it aligned to the dynamic reality (Sect. 4). Our tracking process is not computationally expensive, allowing us to run it in parallel with the operational use of the digital twin, e.g., in its prognostic capacity. If there is a feedback loop regarding the quality of the digital twin's findings, e.g., its error rates or (un-) certainty, we can include this information into the detection as well (not detailed in this article).



The core model (BN core) of a twin is instantiated from data and profiles. The resulting model (BN) is used within a tracking loop that monitors for out of scope use, i.e., transgressions of operational space and lack of certainty in its diagnostic or prognostic use.
It is kept valid (BN online) via continuous adaptations.

Fig. 3. Safeguarding a digital twin's operations

3 Detecting Transgressions of a Model's Operational Space

Given our digital twin's purpose of diagnostics and prognostics, we chose Bayesian networks [11] for their underlying computational models, as they excel at these tasks. These graph-based representations of the joint probability distribution over all modeled variables offer causal probabilistic modeling [6], optimal to investigate cause-effect relationships, e.g., what impact a production setting has on product quality. Furthermore, Bayes nets allow sensitivity analyses to investigate the impact of factors [12], e.g., to determine the dependency of a system's performance on the environment. The listed literature provides details on Bayesian networks, whereas the remaining article only assumes familiarity with the core concepts: Bayes nets are graphs with (random) variables as nodes. Directed edges between nodes show their relationships, i.e., probabilistic or causal dependencies, encoded as conditional probability distribution of a variable given all its parents.

Several modeling techniques exist that enable the efficient use of Bayes nets for system modeling, e.g., by supporting re-use with object-orientation [13], and semi-automated construction of networks from knowledge bases [14] or system descriptions [15]. Such techniques allow us to use consistently generated network building blocks, called *network fragments*, as re-usable and maintainable parts of the underlying model of digital twins. Being able to do so is critical for our work, as the networks will grow very large, resulting in disproportional efforts to ensure their correctness otherwise.

3.1 Probability of Findings for Model Fragments

As Bayes nets encode the joint probability distribution over the modeled variables, we can interpret our objective to determine the fit between model and changing reality as investigation if the data that we observe adheres to this distribution. This is possible given the notion of *probability of findings* that also supports situations of partial observability, which we normally encounter, both w.r.t. missing data (e.g. due to sensor failure) and variables that are part of the model but not observable (e.g. variables that encode an inner state of a component).

The probability of finding is defined as $\Pr(E|H)$, where E is the evidence (the observed data) and H the hypothesis (the model).

The probability of finding for a fragment E_1, where E_2 is the evidence not in the fragment, is $\Pr(E_1|E_2, H)$.

We use Bayes Rule to rewrite this to $\Pr(E_1|E_2, H) = \frac{\Pr(E_1|E_2,H)}{\Pr(E_2|H)}$.

This allows us to calculate the probability of finding for a fragment: the numerator is the probability of finding of all evidence, the denominator can be calculated if we retract the evidence in the fragment.

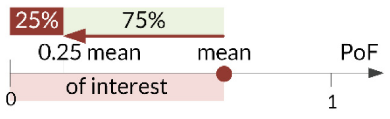
3.2 Monitoring Findings with Western Electric Rules

The probability of findings calculation provides a measurement for the fit of a set of observation for one point in time to the expectations defined by the computational model of the digital twin. To detect relevant dynamics, i.e., real world change that renders our model unsuitable, we use this calculation in an investigation of the respective time-series which follows the approach of statistical process control that establishes a 'normal' series and a criterion to identify 'not normal'.

One of the often-used criteria for these purposes are the Western Electric Rules (WER) that were introduced around 1950 at the Western Electric Company as decision rules for detecting 'out-of-control' or non-random conditions on control charts and thus process instability and the presence of assignable causes. The idea is to measure the mean and variance of a process when it is assumed to be stable and use a series of rules to flag suspicious data. Within the rules, locations of observations relative to the control chart control limits (typically at ±3 standard deviations for symmetrical processes) and centerline defined by the mean indicate whether the process in question should be investigated for assignable causes. The core concept here is that the occurrence of data in certain locations, or of certain series of locations, is too unlikely to be ignored safely. Conceptually, we use the major four Western Electric rules as described in [16]:

1. A single data point is more than 3 standard deviation (sigma) from the mean
2. Two out of three consecutive points are beyond 2 sigma on same side of the mean
3. Four out of five consecutive points are beyond 1 sigma on same side of the mean
4. Eight consecutive points are on the same side of the mean.

Originally, the WER operate on sensor measurement expressed in numbers. We look at probabilities and are only interested in values below the mean, as any value above the mean cannot indicate a decreasing fit between the Bayes net and the data. Given that probabilities form a ratio scale, the WER translate into the appropriate part of the scale from mean towards 0, as shown in Fig. 4 (left) for a ¼ to ¾ ratio. The four major WER are interpreted for probability of finding (Fig. 4 right), wherein the ratios were selected to capture the same likelihoods as the original WER.



- 1 beyond 3 sigma: single PoF<0.003*mPoF
- 2 out-of 3 beyond 2 sigma: 2 out of any 3 with PoF < 0.046mPoF
- 4 out-of 5 beyond 1 sigma: 4 out of any 5 with PoF < 0.32mPoF
- 8 consecutive on one side: 8 out of 8 with PoF < mPoF

with mPoF = mean(PoF) for suitable sample size and PoF = probability of a finding, i.e., a set of observations, according to the joint probability distribution over the respective variables according to the BN.

Fig. 4. Western Electric rules interpreted for probability of findings

To our knowledge, we realized a novel approach with the interpretation of statistical process control for probability of findings. As we describe below, we see several advantages in it. For related work, please refer to [17] and [18] for overviews on adaptation to change and on novelty detection techniques.

4 Tracking for Digital Twin Maintenance

Above, we described our methods to detect and localize discrepancies between the current, observable reality that a system operates in and the computational model that a digital twin uses to monitor and analyze that system and its behavior, and, e.g., to provide diagnosis and prognosis for predictive maintenance.

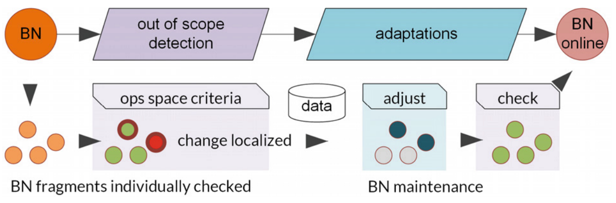
We developed these methods to maintain the computational model, i.e., to keep it and thus the concurrent digital twin up to date w.r.t. the dynamics of the real world. As we face large models that monitor potentially mission-critical systems for which both down-time and non-optimal operations induce high costs, we require that the maintenance of the digital twin occurs timely and that its quality is assured. This leads to a set of requirements for its processes and methods:

- no significant delay in the detection of an operational space transgression (no false negatives)
- no unwarranted maintenance (no false positives)
- efficient mechanisms to update the model:
  localized maintenance without complete re-building or re-training
- low efforts to ensure the updated model's quality:
  only local effects of updates, no needs to re-evaluate the whole.

As presented in Sect. 3, we address the latter two requirements in our approach via the fragmentation of the Bayesian networks. It allows us to adjust only the parts of the model that are outdated. Given the advantages of causal modeling, quality assurance techniques like testing and expert evaluations are also restricted to the respective fragments and the envelope that consists of nodes connecting to them. Consequently, we also require

- localization of change w.r.t. the parts of the model:
  detection only triggers for relevant model fragments.

Figure 5 displays an overview of the resulting workflow for the maintenance of Bayes nets by local adaptations. While its technical details are out of the scope for this article, we show that the methods defined here fulfill the requirements listed above via the experimental validation described in Sect. 6 after an introduction of the underlying industrial use case in Sect. 5.

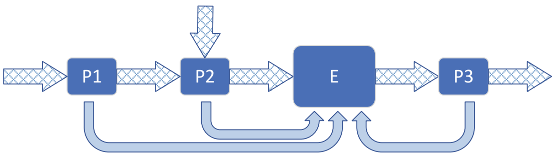


The out of scope detection runs the detection criteria against all fragments and localizes change in those that violate the rules. Learning from data and expert revision adjust the respective fragments. The Bayes nets is rebuilt after a new check.

Fig. 5. Maintenance of Bayes nets model by local adaptations

5 Industrial Use Case

We applied our methods to an industrial use case that we will describe in anonymized terms because of confidentiality. As our work is set to improve complex manufacturing processes that span factory equipment from multiple vendors, our goal is to safeguard a digital twin that monitors production equipment against dynamics that stem either from changes within the factory control, e.g., parameter settings for new products, or from changes to the factory's setup, e.g., updates to equipment and processes. For this article, we consider a simplified production line in which two production steps P1 and P2 precede our own production equipment (Fig. 6).



Production steps P1 and P2 precede our equipment E.
In addition to the material flow (hatched arrows), the data flow (solid arrows) provides partial information on the production processes.

Fig. 6. Production line (simplified example)

A digital twin's purpose in this setup is twofold: it monitors the health of the equipment and it provides information towards process step optimization. The latter acts on partial information, as P1 and P2 are acting as so-called grey boxes, and data from the end of the production line arrives too late to form a feedback that impacts production runs directly. Based on expert knowledge on optimization schemes and fed with historical data, we developed a Bayesian Network for the model core of the envisioned digital twin's second purpose. Figure 7 depicts a variant of this network fitting to the simplified version of our use case.

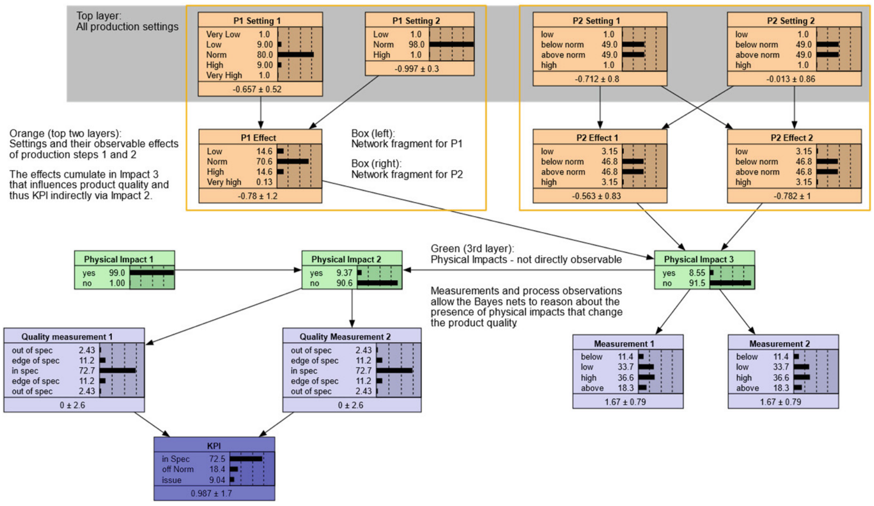
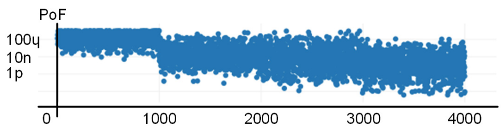

Top layer:
All production settings
Orange (top two layers):
Settings and their observable effects of production steps 1 and 2
The effects cumulate in Impact 3 that influences product quality and thus KPI indirectly via Impact 2.
Box (left):
Network fragment for P1
Box (right):
Network fragment for P2
Green (3rd layer):
Physical Impacts - not directly observable
Measurements and process observations allow the Bayes nets to reason about the presence of physical impacts that change the product quality
P1 Setting 1
P1 Setting 2
P2 Setting 1
P2 Setting 2
P1 Effect
P2 Effect 1
P2 Effect 2
Physical Impact 1
Physical Impact 2
Physical Impact 3
Quality measurement 1
Quality measurement 2
Measurement 1
Measurement 2
KPI

Fig. 7. Bayesian network to monitor process step optimization

6 Experimental Validation

We validated our methods in an experimental setting that simulates our industrial use case, but allows us to control the location of scope of change within the production line, thus providing the ground truth that the tracking needs to cope with.

For this, we generated time-series of 4000 data-points per observable and changed the setup with progressive changes: first altering the machine setup parameters, then replacing production machine P1 with another one having different characteristics, and last introducing an additional source of influence in the P1 production process that cannot be sensed directly but does affect the physical properties of the product leaving machine P1. The changes resulted in a significant decrease of the probability-of-findings (PoFs), i.e., the fit between the model and the data, as Fig. 8 shows.



The visual inspection shows a drop by a order of magnitude for the PoFs after 1000 time steps. This likelihood continues to fall later for the initial Bayes net model, but maintenance of the digital twin would replace the model prior to that.

Fig. 8. Probability-of-findings for all observations

The three changes that we introduced to the setup happened subsequently after 1000 time-steps each. The out-of-scope detection triggered at least with one of its rules basically immediately for each change, as Fig. 9 shows. (There is an inherent delay of one time-step for the first rule, up to eight time-steps for rule 4.)

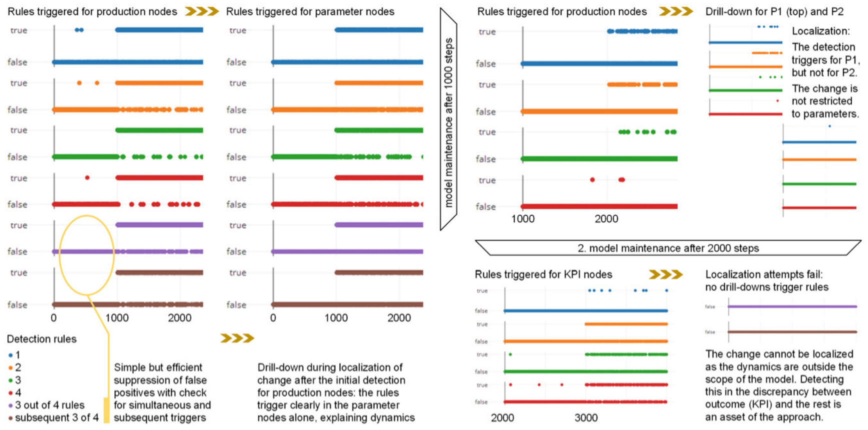


Fig. 9. Experiments on out-of-operational-space detection

In the picture, we see experimental results from the use of the detection rules introduced in Sect. 3 working on three different models: the original one that was generated for the digital twin and those generated from it in two successive maintenance operations in which we updated the parts of the model for which the detection localized the transgression of the models' operational space (top left to top right to lower right).

6.1 Successful Localization of Change

In each section, we show the results for the rules that picked up the real-world dynamics together with possible drill-down steps that facilitate the localization of the change. In the first section, e.g., it was possible to explain the change of the probability-of-findings within the production parameter settings, indicating a new production run, while the second section correctly identifies a change in the P1 step, while the data shows that P2 is still covered correctly by the model. The third section shows an interesting special case in this regard: the detection picks up a difference between the expected and the observed KPI data, but cannot localize a change that would cause the product quality to change. This is consistent with the change we introduced as it was outside the scope of the digital twin.

All in all, we saw that our approach's ability to check both the whole model that underlies the digital twin, but also relevant fragments of the Bayes nets individually enabled an exact localization of the parts of the probabilistic model that required maintenance. This is a major asset to us, as it allows us to keep very large models on track efficiently, a requirement to work with digital twins of industrial scope in dynamic settings.

6.2 Fast and Accurate Detection of Change

Next to that our detection rules trigger on observable change basically without delay, we also see a high precision and accuracy: We have nearly no false positives – instead, we see a clear jump-function in our signal. Furthermore, we have the option to suppress false positives without significant disadvantages, e.g., by requiring that 3 out of 4 rules trigger or by checking for two subsequent observation sets for which the change detection occurs.

We attribute the accuracy to our decision to monitor probability-of-findings instead of the raw data, for which an earlier investigation showed a signal-to-noise ratio so bad that it rendered our detection attempts to be futile. This unusual choice requires a correct interpretation of the results, though: the existence of individual observations that do not trigger the detection rules while we observe a massive set of true positives is inherent to the ideas that we laid out in Sect. 3.2 and Fig. 4 and do not constitute a meaningful false negative, as we only look for the point in time for which the trigger events start. As Figs. 8 and 9 show, we do look for the drop in the mean value of probabilities – but there will always be observations with a higher likelihood than the mean, which results in rules not triggering.

Altogether, both the detection speed and the accuracy visible in our experiments show the suitability of our approach for the tracking of dynamics in concurrent digital twins.

7 Future Work and Conclusion

There are several areas in which we foresee sensible extensions of our work. First and foremost is automation and the generation of a seamless workflow for the continuous tracking of real-world dynamics in digital twins. This is primarily an industry issue, as such a workflow must fit within the operational use of the digital twin and adhere to the respective company's quality assurance procedures.

We believe that the latter requires an additional investigation on how to conduct maintenance operations for digital twins in ways that ensure that automated updates to probabilistic reasoning models do not erase the model's ability to cope with odd but probably important cases just because they did not occur often enough recently to register in the data analysis. Such an examination of coverage could be based on the sensitivity analysis techniques that are available for probabilistic models.

Another, more academic, topic for future work is the integration of expectations w.r.t. the real-world dynamics that is, in certain areas, available to domain experts. The integration of knowledge into data science approaches is typically beneficial and sometimes necessary to successfully support engineering and system analysis tasks, as we pointed out in [19]. In the context of the work we present here, we expect that we can improve our results further by fine-tuning the out-of-scope detection to the origin and nature of dynamics and the behavior of control loops that adapt the monitored systems to the situation at hand.

The need for further developments of this kind is debatable, as efforts need to remain in balance with expected improvements. Drawing only from experiences in industrial R&D within a single domain, we cannot provide a general notion w.r.t. its necessity: our detection of transgressions from the operational space of predictive twins worked very fast and accurate. We were, without mentionable efforts, able to set up our detection mechanisms such that neither false positives nor false negatives impacted results in our controlled experimental setup – an achievement that outperformed the individual technologies which inspired our approach – and could do so based on relatively small time-series, thus fulfilling all our requirements.

Further improvements would therefore need their justification in the demands on the original system, the digital twin, and its operation itself: safety-critical systems warrant extra care w.r.t. the unlikely, but potentially catastrophic event; systems with excessive down-time costs should not be taken offline to maintain their twins, which translates to high demands on a twin's robustness versus manageable dynamics, warranting dedicated handling of foreseeable dynamics; but concurrent twins that monitor systems that are expected to operate in a stable fashion will require only the safe-guards that the work we presented can provide.

Given that digital twins are, by themselves, a new technology for the control and management of complex systems, we hope that our work helps to ensure their continued success, but also that we raised awareness on the challenge that real-world dynamics pose to them.

References

1. Monash, C.: Examples and Definition of Machine-Generated Data. Monash Research Publication (2010). www.dbms2.com/2010/12/30/examples-and-definition-of-machine-generated-data. Accessed Apr 2018
2. Laney, D., Jain, A.: 100 Data and Analytics Predictions Through 2021. Gartner Report G00332376 (2017)
3. Gartner Press Release: Gartner Identifies the Top 10 Strategic Technology Trends for 2017. Gartner (2016). www.gartner.com/newsroom/id/3482617. Accessed Apr 2018

4. Grieves, M., Vickers, J.: Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In: Transdisciplinary Perspectives on Complex Systems, pp. 85–114. Springer International Publishing (2016)
5. Oracle: Digital twins for IoT applications. Oracle White Paper (2017). www.oracle.com/us/solutions/internetofthings/digital-twins-for-iot-apps-wp-3491953.pdf. Accessed Apr 2018
6. Pearl, J.: Causality. Cambridge University Press, New York (2009)
7. Borth, M.: Probabilistic system summaries for behavior architecting. In: Proceedings of the Complex Systems Design and Management 2014 CEUR Workshop, pp. 71–82 (2014)
8. Christensen, J.J., Andersson, C., Gutt, S.: Remote condition monitoring of Vestas turbines. In: Proceedings European Wind Energy Conference, pp. 1–10 (2009)
9. Gupta, S., Starr, M.: Production and Operations Management Systems. CRC Press, Boca Raton (2014)
10. Schwab, K.: The Fourth Industrial Revolution. Portfolio Penguin, London (2017)
11. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, New York (2001)
12. Jensen, F.V., Aldenryd, S.H., Jensen, K.B.: Sensitivity analysis in Bayesian networks. In: Carbonell, J.G., et al. (eds.) Symbolic and Quantitative Approaches to Reasoning and Uncertainty. Springer Lecture Notes in CS, vol. 946, pp. 243–250 (1995)
13. Koller, D., Pfeffer, A.: Object-oriented Bayesian networks. In: Geiger, D., Shenoy, P. P. (eds.) Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI 1997), pp. 302–313. Morgan Kaufmann Publishers Inc. (1997)
14. Laskey, K.B., Mahoney, S.M.: Network fragments: representing knowledge for constructing probabilistic models. In: Geiger, D., Shenoy, P.P. (eds.) Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI 1997), pp. 334–341. Morgan Kaufmann Publishers Inc. (1997)
15. Borth, M., von Hasseln, H.: Systematic generation of Bayesian networks from systems specifications. In: Musen, M.A., Neumann, B., Studer, R. (eds.) Intelligent Information Processing, pp. 155–166. Kluver (2002)
16. Western Electric Rules: From Wikipedia. en.wikipedia.org/wiki/Western_Electric_rules. Accessed May 2018
17. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, H.: A survey on concept drift adaptation. ACM Comput. Surv. (CSUR) **46**, 44 (2014)
18. Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Process. 215–249 (2014)
19. Borth, M., van Gerwen, E.: Data-driven aspects of engineering. In: IEEE SoSE 2018, Paris (2018, accepted)

How to Boost the Extended Enterprise Approach in Engineering Using MBSE – A Case Study from the Railway Business

Marco Ferrogalini[1](✉), Thomas Linke[2], and Ulrich Schweiger[2]

[1] Bombardier Transportation, 1 Place des Ateliers, 59154 Crespin, France
marco.ferrogalini@rail.bombardier.com
[2] Knorr-Bremse Systeme für Schienenfahrzeuge, Moosacher Str. 80, 80809 Munich, Germany
{thomas.linke,ulrich.schweiger}@knorr-bremse.com

Abstract. This paper presents an overview of why Model Based System Engineering (MBSE) is the mandatory solution to engineer complex railway systems and subsystems, giving some insights about the complexity and technical challenges. But its main focus will be on how MBSE can be a key enabler on the implementation of an extend enterprise approach in the engineering domain. The case study is on rolling stock systems and brake subsystems developed by the two companies Bombardier Transportation and Knorr-Bremse which are in a customer-supplier relationship.

1 Introduction

1.1 Railways Business OEM, a System Integrator

OEMs for rolling stock products such as Bombardier Transportation are providing train sets to railway operators (final customers and end users together with the passengers). Except few subsystems like the carbody, the propulsion, the bogies and the train control and monitoring system, all the rest of the twenty to twenty-five subsystems are purchased from external suppliers. The core system engineering activities are:

- Gathering and consolidating all the input requirements (customer, standards and regulations)
- Developing architectural concepts (mechanical, functional, performance) at vehicle level (system of interest) and deriving requirements into the technical subsystem specifications to precisely define the scope of supply of any integrated subsystems
- Maximizing the re-use of existing subsystem solutions
- Verifying and validating the final product (integrated systems) to ensure full compliance of the input requirements and customer satisfaction (Fig. 1).

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 79–96, 2019.
https://doi.org/10.1007/978-3-030-04209-7_7

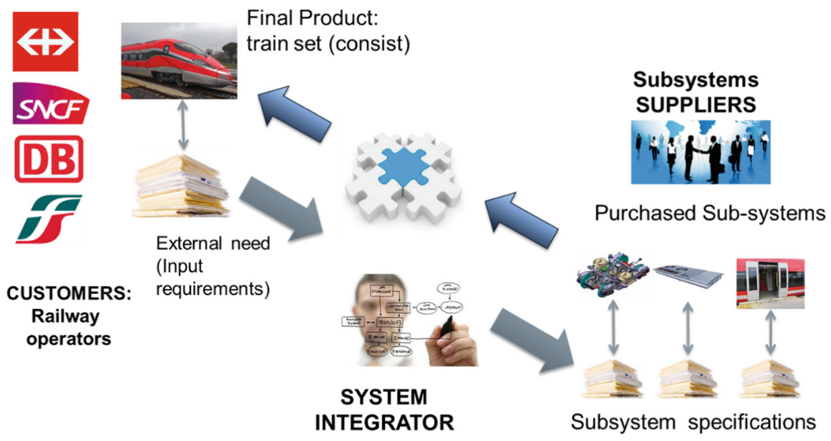


Fig. 1. Role of the system integrator

1.2 Railways Business OEM, a System Integrator

The vision of rolling stock subsystem suppliers such as Knorr-Bremse is to provide comprehensive solutions for a specific functional vehicle domain. Knorr-Bremse targets to free up the system integrator from the hassle of dealing with subsystem specific details by providing pre-developed, approved, high level products with a defined set of functions and variability to satisfy different customer needs. The core system engineering activities are:

- Gathering and consolidating all input requirements (customer, standards, regulations) on subsystem level.
- Advising system integrators in subsystem specific topics and supporting them in defining an efficient solution for their train system.
- Specifying and defining the architecture of the subsystem to fit into the overall vehicle architecture.
- Maximizing the re-use of existing solutions, but customizing it to project specific requirements.
- Integrating, testing and validating the subsystem inhouse and on-site together with the system integrator.

1.3 Rolling Stock Systems Complexity Growth

As for many other systems (e.g. smart phones), there's a general trend to increase complexity for rolling stock systems acting on several axes:

- More and more integrated functionalities for passenger services (public announcement, infotainment and multimedia, video surveillance, internet access).
- IoT (internet of things) makes the train seen as a "connected object", therefore new interfaces with external systems requires new functionalities (e.g. drive from remote control center) and with a very high level of safety and security (cyber-security).

Looking to the technology evolution on the control systems for rolling stock, it's important to note that these systems started as fully mechanical systems (steam locomotives), and once electronics appeared, all the «non-mechanical» functions to monitor and control the system were realized initially with wiring logic (relays logic). It's just much more recently (last decade) that with the introduction of the "train control and monitoring system" the control started to be realized with a combination of software and hardwired logic. Typically, safety functions and reliability performance are granted by hardwired implementation.

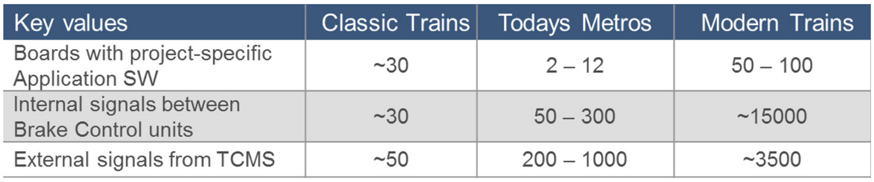
One of the most important factors of this complexity growth is the transformation to reduce more and more hardwired function in favour of software. Typically, around twenty to twenty-five different subsystems (traction, braking, doors, HVAC, passenger info and entertainment, CCTV, signaling, etc.) shall be functionally integrated by using several different types of communication buses to interconnect within themselves and with other legacy rolling stock.

1.4 Rolling Stock Subsystems Complexity Growth

There are different factors which increased the complexity of the subsystems. Technical complexity growth by increased electronic and software driven functionality of the subsystem. Increased performance requirements lead to a more sophisticated and precise control of the subsystem.

For example classical power head driven trains had a central pneumatic brake pipe pressure control and each coach contained a pneumatic distributer valve. Usage of electronic control is limited to wheel slide protection and a limited diagnostic functionality.

Modern trainsets with fully distributed power have a full electronic per bogie control of the brake pressure providing quick response to brake demands and full blending with electrodynamic brake and eddy current brakes to optimize the energy consumption, life cycle cost. Brake performance is improved to maximise the line capacity. Electronic Adhesion Management maintains a constant deceleration rate even in emergency brake (Fig. 2).



Key values	Classic Trains	Todays Metros	Modern Trains
Boards with project-specific Application SW	~30	2 – 12	50 – 100
Internal signals between Brake Control units	~30	50 – 300	~15000
External signals from TCMS	~50	200 – 1000	~3500

Fig. 2. Number of signals vs application

As mentioned above, when the train shall provide all the typical services of a connected object (IoT) then the subsystems have been forced to deeply modify their traditional data communication architectures. Indeed the introduction of Ethernet technology in the rolling stock control domain have been required to support additional services like web based maintenance, improved diagnostic capabilities as well as condition monitoring and predictive maintenance. Ethernet technology is also an enabler for interconnecting different subsystems to provide additional customer valued functionality. Cyber security and safety integrity levels also drive additional engineering tasks. Upcoming demands like "virtual" coupling, wireless TCMS, full drive by wire and autonomous driving will increase the system complexity even further.

Additional to these technical complexity drivers, there is a number of non-technical drivers. In the past years all successful companies had to go global. This is not only affecting the customer interfaces but also internal engineering departments. For example, a global acting company like Knorr-Bremse has more than 100 subsidiaries in 30 countries. To provide an outstanding customer interface, application engineering is located very close to our global distributed customers. All these complexity drivers can be tackled by different technical options, but efficient system engineering methods become a crucial factor to remain competitive in the business.

2 Functional Architecture Concept Process and MBSE at Bombardier

This chapter will provide insights about the engineering phases performed at BT to conceptualize a rolling stock system functional architecture and also some insights about how the concept phase has been implemented by using a model based approach based on the SysML language [2, 4, 11, 17].

2.1 Functional Architecture Engineering Process and Modelling Methodology

The process is divided in the following phases, described here below.

Input Requirements Analysis

This phase consists of analysing and consolidating all the input requirements which can influence the definition of the functionalities of the rolling stock system. The possible sources are the customer specifications, standards and norms, but also Bombardier internal requirements and constraints. The full set of requirements is considered during the concept phase and traceability measures are put in place to ensure that all of them are covered by the design.

Train Operability Analysis

This phase aims to provide full understanding of the operational context, environmental constraints and therefore to define the operational scenarios and to capture the train external interfaces. The analysis is structured following a standard structure called OBS (Operability breakdown structure) made of three levels:

- Level 1 – Operational contexts: define the contexts in which the train performs its mission (they have been standardized into the following: Normal, Restricted, Degraded, Emergency and Maintenance). A state diagram shows all the possible transitions between the different contexts. Each operational context is further decomposed into operational scenarios.

- Level 2 – Operational scenarios: each operational context is broken down into more granular ones. This breakdown is formalized through a set of activity diagrams, one per each operational context. Operational scenarios are modelled as activities.
- Level 3 – Consist use cases: basic "bricks" of the train behaviour which are defined in the next process phase. As for the previous step, each consist use case is modelled as an activity (and as a use case), each operational scenario is formalized trough an activity diagram.

Consist Operational Analysis
Once defined the system operational needs (ConOps), the next step is to perform the "external" functional analysis to determine the rolling stock black box functions which will support the full set of operational scenarios. This phase is mainly based on the following activities:

- Define the environment and actors.
- Define all the possible usages (use cases) of all the actors and environment which will interact with the train in both normal condition and degraded conditions to fully cover the operational scenarios. As mentioned before those use cases are combined into activities diagrams to describe the operational scenarios.
- Characterize the train external interfaces by formalizing all the exchanges (inputs/outputs).
- Define the train states and the related transitions. This analysis is structured following a standard functional breakdown structure (three levels); all the use cases are organized under the functional breakdown structure (FBS) level three. This means that the expected behaviour of the train is defined for each of the level three functions.

Consist Functional Analysis
This phase consists of performing the functional analysis of the system as a white box; all the required behaviour defined from a black box point of view in the precedent step is now decomposed into more granular functions (level four). For each function level three, this phase aims to identify the set of functions which will decompose the main function level three and which other functions which decompose other functions level three will contribute by providing outputs or consuming inputs. This phase is mainly based on the following activities:

- Decompose each function level three to the next level and determine which other functions level four (belonging to other functions level three) are contributing.
- Functional interactions (functional signals) between the functions and with the system external actors (coherently with what defined in the operational analysis step), elaborating the functional architecture (per each function level three)
- Determine the behavior of each function level four and formalize it through:
  - functional derived requirements
  - state machine diagrams (this will enable the model simulation)
- Describe the holistic dynamic behavior of the function level four trough sequence diagrams.

Consist Technical analysis
This phase runs in iteration with the previous one. To enable the implementation of the defined functional architecture, any functions level four shall be allocated in a unique manner to one subsystem. This means that the functional decomposition analysis is normally performed several times until reaching the optimal setup.

This phase is mainly based on the following activities:

- Decompose the rolling stock into the physical subsystems.
- Allocate each function level four to a given subsystem, as mentioned this might require to review the functional decomposition.
- Define for each functional signal a real physical signal (signal type and signal name), fully defining the functional ICDs.

2.2 MBSE Tool Chain and Main Outputs

Bombardier transportation has chosen the SysML language and developed on it a BT owned modelling methodology briefly described in the previous chapter. The modelling tool that is currently in use is Magic Draw with a centralized server which permits concurrent engineering from distributed teams on the same model. Because the tool to manage requirements across the company is DOORS, we interfaced it with MagicDraw thanks to the Cameo Data hub plugin and following a strict synchronization process. Out of the MagicDraw models all the functional specifications (documents) are automatically generated and stored in the PLM system (Siemens TCUA) for the several functional baseline releases. Those functional specifications (documents) are then typically shared with:

- The customer during the project design reviews
- The suppliers (like Knorr-Bremse) because they define their functional scope of supply (functional requirements and functional ICDs).

The model is verified trough static checks (verifying that all the modelling rules have been respected) and trough simulation, permitting to verify in a very early stage the correctness and completeness of the functional architecture (Fig. 3).

2.3 Deployment Status

The large-scale rolling-out of MBSE in Bombardier Transportation started around three years ago with a key pilot project which is now the base of one of the most important product family for the mainline segment. Because of the positive return on experience and investments, it has been decided to set this approach as mandatory for all the future product definitions; the implementation has already started on several other product families beyond the first pilot. The deployment of MBSE has concerned a population of around hundred functional engineers which are required to formalize their concept through the described approach and a population of around three hundred engineers which are the customers of what is formalized in it and therefore they have been trained as model readers.

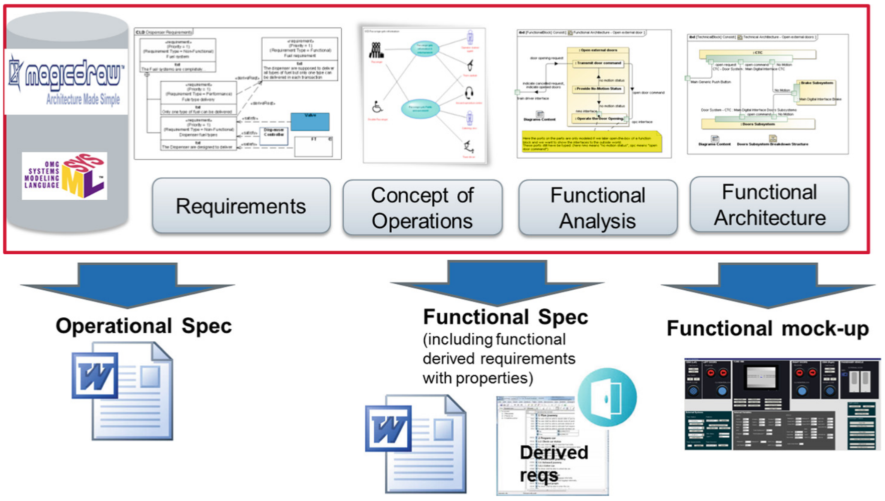


Fig. 3. MBSE tool landscape at BT

3 Functional Architecture Concept Process and MBSE at Knorr Bremse

The key idea of shifting from the classical document based engineering approach to the model based engineering approach is not in discarding the whole set of documents but rather keep all the development artefacts in a common and concise model. Out of this model the typical system engineering process documents of the railway industry can be generated. The model is continuously enriched and detailed with the information created during each design step being the single source of truth.

It helps the engineers to maintain coherency, consistency and provides a “real-time” common view on the system design. The integration of the subsystem into the vehicle requires advanced engineering methods to achieve a seamless integration and an efficient on-site commissioning, testing and vehicle homologation. Mains benefits are:

- Clearly defined and managed interfaces, not only on static interface definition, but as well as behavioural description to improve the quality of the whole process.
- A modular approach with defined variation points enables re-usable building blocks as a product line approach. This allows the system integrator for a quick and easy adaption of their vehicle platform to different operator requirements.

The following chapter deals with the model based process based on SysML language which has replaced the conventional document driven approach.

3.1 Functional Architecture Engineering Process

A closed circular process model (see picture hereafter) describes the key activities of the domain engineering but also of the application engineering as defined by the standard ISO 26550 [8].

The core concept is to feed and re-use a common asset base made of subsystem functions, subsystem complex technical elements which realize system functions, and system architectures which combine different technical elements to a train sub system (Fig. 4).

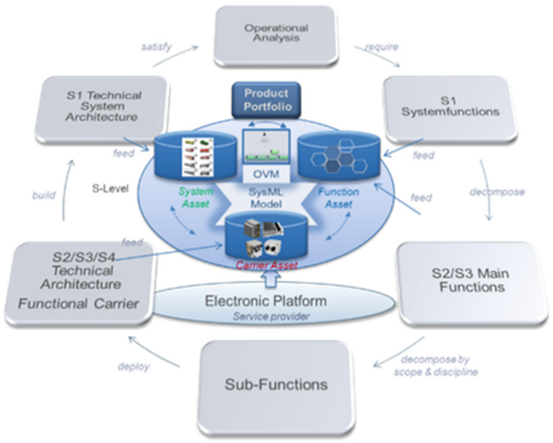


Fig. 4. KB Functional architecture engineering circular process

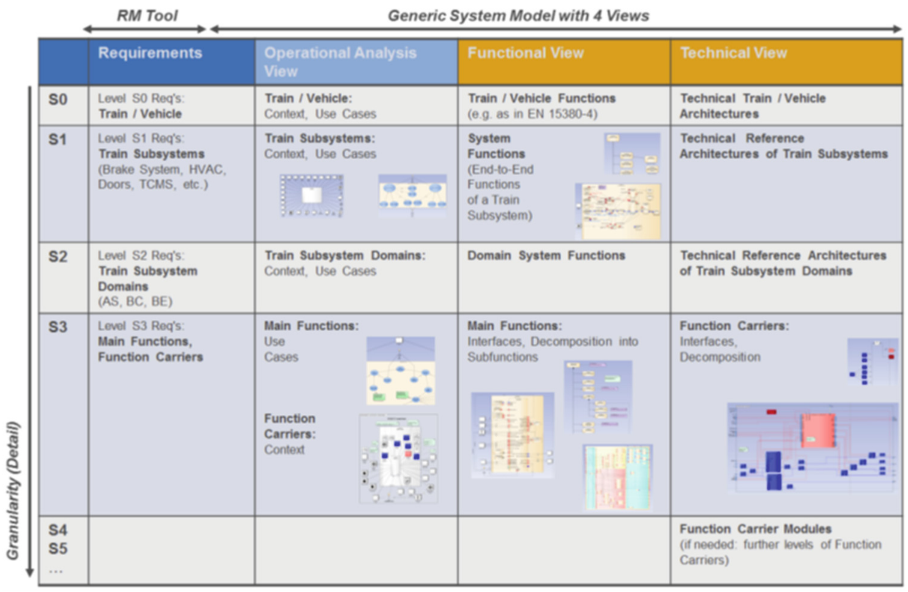
3.2 Modelling Methodology

With SysML, a standardized language for system modelling has been defined by the OMG organization (Object Management Group). However, there is no standardized approach for the methodology of system modelling or the structure of a system model. SysML 1.3 [2, 4, 11, 17] is used as the common modelling language in all parts of the System Model except the Feature Model. The Feature Model was originally formulated in a feature-based ad-hoc notation based on [9], but it has been transferred to OVM [12], the variant modelling notation used in PTC Integrity Modeler. OVM (Orthogonal Variability Model) is not part of the SysML standard. The System Model is the central repository for Domain Assets, i.e. the results of Domain Engineering which are stored in a structured way to be re-used by Application Engineering. The structure of the model is based on the methodology developed in the projects SPES 2020 [13, 15] and CESAR [14].

Views and Levels

Views and levels can be seen as two dimensions defining a matrix in which all model content is represented (see figures hereafter). The relation typically used for connecting elements of different levels within the same view (vertical direction) is the composition (whole-part relation); the allocation relation is used to connect elements of the same level from the Functional to the Technical View (horizontal direction) (Fig. 5).

The most important entities in the model are the Main Functions and the Function Carriers. The Main Functions define the functional breakdown of the complete system.



RM Tool — Generic System Model with 4 Views

Granularity (Detail)

	Requirements	Operational Analysis View	Functional View	Technical View
S0	Level S0 Req's: **Train / Vehicle**	**Train / Vehicle:** Context, Use Cases	**Train / Vehicle Functions** (e.g. as in EN 15380-4)	**Technical Train / Vehicle Architectures**
S1	Level S1 Req's: **Train Subsystems** (Brake System, HVAC, Doors, TCMS, etc.)	**Train Subsystems:** Context, Use Cases	**System Functions** (End-to-End Functions of a Train Subsystem)	**Technical Reference Architectures of Train Subsystems**
S2	Level S2 Req's: **Train Subsystem Domains** (AS, BC, BE)	**Train Subsystem Domains:** Context, Use Cases	**Domain System Functions**	**Technical Reference Architectures of Train Subsystem Domains**
S3	Level S3 Req's: **Main Functions, Function Carriers**	**Main Functions:** Use Cases **Function Carriers:** Context	**Main Functions:** Interfaces, Decomposition into Subfunctions	**Function Carriers:** Interfaces, Decomposition
S4 S5 ...				**Function Carrier Modules** (if needed: further levels of Function Carriers)

Fig. 5. KB MBSE approach – views and levels matrix

The Function Carriers are technical products ready to be used in customer projects; they are located on the same level of the Technical View. Function Carriers and discipline-specific elements (on levels D…) can be grouped into System Families. On the left side of the matrix, the column Requirements shows the connection between the System Model and the textual requirements stored separately in the requirements management tool. The same levels used to structure the System Model are also used to structure the textual requirements which might be linked to model elements of the same level to ensure traceability between requirements and solutions. The light-blue colour used for the Operational Analysis View in the figures below illustrates the special role of that view as a bridge between the textual requirements and the design models in the Functional and Technical View (orange columns). The Feature Model containing information about the variability of the system is orthogonal to both views and levels; feature constraints may be connected with elements of any view and any level. The Feature Model thus represents a third dimension of the Generic System Model and is not represented in the matrix.

The three views in the model are:

Operational Analysis View

The Operational Analysis View falls in two parts: the Context Model and the Use Case Model.

Context Model

The context of a system is the sum of all human, natural or technical entities that surround the system and are relevant for the correct operation of the system in its environment. The context does not belong to the system; in a development project, it must usually be accepted as is, because it may not be changed. Therefore, the context has a strong impact on the requirements for the system. The purpose of the Context Model is to analyze influences and constraints on the system originating in its environment. Such influences may be of physical (e.g. climate conditions), technical (e.g. communication protocols), social (e.g. ergonomics and usability), or legal nature (laws, standards and other regulations). In Context Diagrams, the System of Interest (SoI) is represented as a black box interacting with Context Elements. The Context Model serves as an entry point to the Generic System Model. It includes separate Context Diagrams for all Main Functions. In combination, these diagrams yield a Context Diagram for the complete brake system, or, in a similar way, for any other train subsystem included in the model.

Use Case Model
Use Case Modelling is a high-level method to characterize the functionality of a system from the individual perspectives of the different users. Users in this sense may be humans as well as technical systems that use services of the system of interest. In the KB System Model, there are no special rules for Use Case Diagrams; they simply follow the standard as defined in UML or SysML.

Functional View
The modelling approach at KB is function-based, meaning that modelling activities typically start with a functional analysis that abstracts from technical details. The function of a technical system is the action or purpose for which it is designed. Analysing and decomposing the function of a system is a way to find abstract descriptions of its actions and purposes. Functional descriptions are solution-neutral, which means that they don't anticipate design decisions or technical details.

Function Classification
A model-wide Function Classification is defined for the complete functionality of the systems considered in the model. The Function Classification is a tree with Main Functions as leaf nodes. It is represented as a structure of nested model packages and used as a common structuring backbone in several model parts. The Function Classification is a classification scheme which is similar, but not identical with the system levels (S0, S1, etc.). While the Function Classification is used to classify Main Functions on the same level, and to group them into packages for easier orientation, the system levels are used to model functions that combine several Main Functions in a certain way to realize more comprehensive functionality (called System Functions). Currently, the Function Classification comprises the functionality of the Brake System. In the future, it may be extended to include other train subsystems, such as Propulsion, Doors, Air Conditioning, etc.

The most important diagrams used in the Functional View are Functional Context Diagrams (which define the link to the Context Model), Function Trees (decompositions of functions into Sub-Functions), and Function Networks (showing the interrelations of Main Functions within a System Function or Sub-Functions within a Main Function).

Technical View
Describing how the System of Interest is composed of technical or physical components. Like the other views of the model, the Technical View is organized in multi-discipline system levels (S0, S1,…) and discipline-specific levels (D1, D2,…). Each model element of the Technical View is classified into one of the levels, and for each model element the following information is provided:

- its system boundary;
- its constituents (the model elements of lower levels of which it is composed);
- its interfaces at the system boundary;
- the interfaces of its constituents and how they are interconnected;
- the engineering discipline it belongs to (for discipline-specific elements).

The central model elements in the Technical View are the Function Carriers. A Function Carrier is a standardized subsystem of system level S3 which implements a defined set of standardized functions (Main and/or Sub-Functions), and is composed of standardized units of one or more disciplines (mechanical, pneumatic, electrical, electronic (including software)) realized on physical devices in a defined arrangement. The model elements of the Technical View are described by the following two diagram types:

- Decomposition Diagrams: Block Definition Diagrams (BDD) that define which parts an element consists of;
- Architecture Diagrams: Internal Block Diagrams (IBD) that show how the parts within an element are connected through their interfaces.

These diagram types occur on each of the model levels in a similar fashion:

On level S3, for example, they describe the interfaces and internal architecture of Function Carriers; whereas, on level D1, they describe the interfaces and internal architecture of Function Carrier Elements.

Levels
To structure the model according to abstraction and granularity, the following levels are defined. The levels starting with S… (for “system”) contain mixed-discipline items, those starting with D… (for “discipline”) contain items that are discipline-specific, i.e. purely mechanical, pneumatic, electrical, or electronic. For Requirements Management, the same level numbering is used as in the System Model.

Variability
The Variability Model constitutes a cross-cutting aspect of the model. It is used to model and manage the variation of the System Model and of other development artefacts (e.g. documentation). The Variability Model contains features and their interdependencies. According to ISO/IEC 26550 [8], features are abstract functional or non-functional characteristics of a System of Interest for end-users and other stakeholders. The Variability Model has relationships with all other views: artefact dependencies can be used to describe variability anywhere in the System Model. Thus, the Variability Model is orthogonal to the rest of the System Model and may be regarded as a third dimension added to the Level/View Matrix (see Views and Levels). In the

Knorr-Bremse System Model, variability is expressed in a graphical notation called Orthogonal Variability Model (OVM), which may be considered an “add-on” to SysML.

Tools and Infrastructure

With SysML, a standardized language for system modeling has been defined. Knorr Bremse has chosen SysML 1.3 as the common modeling language in all parts of the System Model except the Feature Model which is expressed in OVM [12]. The MBSE approach was started with Enterprise Architect, a simple user friendly general-purpose modeling tool. It was important to focus on the SysML language and the modeling methodology and keep the obstacle of a complex design tool very low. Enterprise Architect served this approach very well.

In 2017 the MBSE tooling for the System Model has been changed as large-scale enterprise requirements and IT strategies had to be considered when rolling out the MBSE approach to a large-scale community. The modelling tool PTC Integrity Modeler took the place of Enterprise Architect and the model content had been migrated. Requirements have been already gathered with PTC Integrity Lifecycle Manager. Lately the OSLC interface between Lifecycle Manager and Modeller has been introduced. As document generation engine KnowDocs from KnowGravity is going to be deployed.

3.3 Deployment Status

KB’s MBSE Methodology has been developed and applied during a pilot project focusing on functional standardisation. Following the positive outcome of the pilot, the method has been extended to cover also the technical deployment of the functional elements. By beginning of 2018 the method and also the tools have reached a level of maturity where it has been decided to promote it as a company standard approach for the brake system domain engineering. The roll out is currently ongoing by the implementation of brake subsystem product lines under development as a major product involving around 50 product system engineers acting as model architects. A much higher number of engineers will be trained as “readers” of the model with particular focus on getting the buy-in of an audience which is not used to a model based approach and to extensive modelling tools and abstract languages like SysML.

4 Rolling Stock Integrator Versus Subsystem Supplier – A Key and Complex Relationship

This chapter will provide insights on the main focus of this paper: the customer-supplier relationship between a system integrator and a subsystem provider. This is a key area where the extended enterprise approach can be successfully implemented, especially when boosted by the application of the MBSE on both sides.

4.1 State of the Art

As it's explained in the introduction, the most important element of this interface is the set of requirements which are provided from the rolling stock integrator as the result of the architectural/integration work. This set of requirements shall span all the integration aspect, mechanical, functional and performance and shall be managed in configuration and changes.

This set is the main input for the subsystem supplier. According to the life cycle stage, different feedbacks on each requirement shall be provided:

- During the bid phase: a clause by clause, stating compliance status
- During the detailed design phase:
  - A list of satisfaction arguments which explain how the requirements will be covered by an intended design solution
  - List of means of proof (the way the requirements will be validated)
  - The technical documentation which describes the solution
  - The ICDs (Interface control documents)
  - The software release notes
- The validation evidence that the requirements are effectively implemented into the solution.

All those exchanges back and forth are normally based on traditional Microsoft Office documents (Word, Excel). As state of the art of the integrator-subsystem supplier relationship this approach has shown some typical inconvenients, causes of inefficiency, which have many commonalities with the document based engineering approach:

- Misunderstanding/interpretation of requirements (text based)
- Lack of visibility/understanding of the integration environment for the subsystem concerned
- Low level re-use of existing off-the-shelf solutions – the state of art approach is typically top-down and doesn't support a structured re-use methodology
- Very-long life cycle stages, each step requires several cycles to get the same understanding/alignment between the two parties
- Ambiguous interface definitions, on functional level and dynamic behaviour.

4.2 Extended Enterprise, the Next Generation Supply Management

What Is Extended Enterprise? The term "extended enterprise" represents a new concept where a company goes beyond its traditional perimeter and may include its business partners, its suppliers, and its customers in it. When we focus on the suppliers, the notion of extended enterprise can be translated into virtual integration, outsourcing, joint global R&D programs, partnership agreements and preferred supplier approach.

The traditional way of thinking of an enterprise considers a linear value chain from marketing to service all along the product life cycle where suppliers are thought to be "outside" the organization's domain. The railway market is currently evolving having the product development life cycle duration dramatically reducing, the competition intensifying and the level of risk of not delivering on time consequently increasing. This trend has pushed smart organizations to rethink the way of delivering value to customers, for instance by reshaping the relationship with suppliers. A way to pave this is improving collaboration and communication to share the end goal and the related risks.

According to Jan Duffy and Mary Tod, IDC, the authors of the article "The Extended Enterprise: Eliminating the Barriers" [3], the extended enterprise can only be successful if all the component groups and individuals have the information they need to do business effectively.

We came here to the core part of this paper, where it will be described how two companies in a customer-supplier relationship in the railway business, respectively Bombardier Transportation and Knorr-Bremse, have defined a model based methodology to support the extended enterprise approach for the functional integration of a brake subsystem in the rolling stock product.

5 The BT-KB MBSE Cycle

This chapter will give insights on a model-based collaboration concept between Bombardier Transportation (BT) and Knorr-Bremse (KB). The process is a cyclic process passed through as long as necessary to achieve a solution accepted by both partners. It is built on system models as they are used in Model-Based Systems Engineering (MBSE). Both partners possess system models that have been developed separately and in different tools, yet based on the same modelling language (SysML). The BT-KB MBSE Cycle proposes to connect these models and to use them in a well-defined way to foster model-based collaboration. The goals of the BT-KB MBSE Cycle are:

- Simplifying the definition of interfaces
- Generating interface control documents based on consistent models
- Traceability of functional requirements and design decisions across company borders
- Facilitating iterative refinement and change management
- Effect analysis across company borders
- For subsystems (e.g. Brakes, Doors, etc.): Using standardized products taken from a portfolio of standard system functions, function carriers and reference architectures
- On vehicle level: Integration of standardized subsystem products into standardized functional building blocks.

5.1 The BT-KB Model-Based Systems Engineering Cycle

The following figure illustrates the steps involved in the BT-KB Engineering Cycle. Each step is explained in a separate section below (Fig. 6).

Step 1 (BT to KB): Forward Requirements and Map to Functions

The functional model of BT is structured into Functional Contexts (Functional Breakdown Structure Level 3) and Functional Blocks (Functional Breakdown Structure Level 4). The derived requirements which are forwarded to KB are corresponding to the Functional Blocks of Level 4. KB receives requirements for each Functional Block which is allocated to the Brake subsystem ("SBA Brakes"). KB maps the requirements

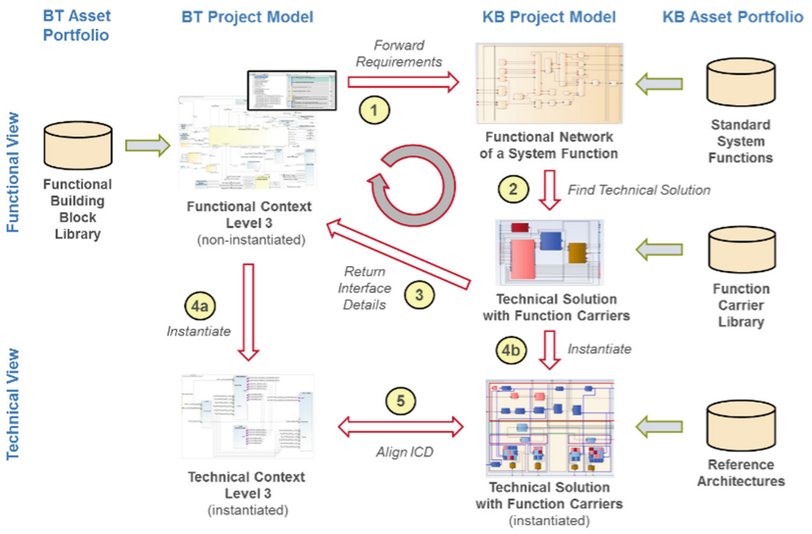

BT Asset Portfolio
BT Project Model
KB Project Model
KB Asset Portfolio
Functional View
Technical View
Functional Building Block Library
Functional Context Level 3 (non-instantiated)
Forward Requirements
1
Functional Network of a System Function
Standard System Functions
2
Find Technical Solution
Return Interface Details
3
Technical Solution with Function Carriers
Function Carrier Library
4a
Instantiate
4b
Instantiate
5
Align ICD
Technical Context Level 3 (instantiated)
Technical Solution with Function Carriers (instantiated)
Reference Architectures

Fig. 6. BT-KB model-based systems engineering cycle

received from BT to functions of its functional model. The BT Functional Contexts (Level 3) correspond roughly to the KB System Functions, the BT Functional Blocks to the KB Main Functions and Sub-Functions used in a System Function. Since the BT requirements are scoped to Functional Blocks, they will eventually be mapped to Main Functions and Sub-Functions on the KB side. This process is repeated for all Functional Contexts which contain any Functional Blocks allocated to the Brake subsystem.

Step 2 (KB): Find a Technical Solution Based on Standardized Function Carriers
On the KB side, the KB Asset Portfolio of Standard System Functions and Function Carriers is used to find a technical solution which satisfies the requirements. In the KB generic system model, Standard System Functions and Function Carriers are connected. For a specific OEM project, a reduced branch of the generic system model is generated that contains only project-specific content. This is accomplished by selecting features (variability items) describing the OEM project. After the project-specific selection of features a model transformation is triggered which results in a project-specific model. This model contains a set of Function Carriers appropriate for designing a technical solution tailored to the specifics of the OEM project.

Step 3 (KB to BT): Return Interface Details
For each Functional Block allocated to the Brake subsystem on the BT side, KB provides a Technical Solution Block satisfying the requirements. The technical solution for a Functional Block includes:

- A complete specification of the interface,
- A White Box view showing how the Function Carriers are interconnected to realize the technical solution.

To make the interface alignment traceable on both sides, the KB technical ports are mapped to the BT functional signals, because the BT functional view has roughly the same level of technical detail as the KB technical view. This mapping should be supported by a tool linking the corresponding elements of the two models together. Once the model elements are linked, this information can be used in further iterations and will speed up the process of alignment in the BT-KB MBSE Cycle. There is a special reason why a White box view of a Technical Solution Block is delivered by KB in addition to the interface specification: Some of the connections between the Function Carriers in the Technical Solution Block might be realized by BT as part of the train infrastructure (bus systems, wiring, pipes). In the model, the connections for which this is true may be marked with special attributes. By analysing these attributes of the connections within the Technical Solution Blocks, requirements to the BT train infrastructure (e.g. bandwidth and performance of bus systems) can be derived.

For simplicity, the technical solution exchanged between the partners in this step is not instantiated yet. This means that groups of technical elements which need to be repeated several times in the train architecture are only represented once in the models exchanged. During the mapping process, discrepancies between both models become evident and can be analysed. Because this might lead to changes of requirements or of interface elements in the model, the process described here is a cyclic process comprising as many iterations as necessary to achieve a solution accepted by both partners.

Step 4 (BT and KB): Instantiate the Technical Solution
After BT has received a non-instantiated technical solution from KB, both sides independently work on instantiating it. Instantiating means adding information on how often the technical elements are repeated in the train architecture. The system of interest for this design step is a consist. On the KB side, a portfolio of reference architectures supports the instantiation. The results of the instantiation are project-specific technical models on both sides. The KB model will contain information on the correct number of Function Carriers needed to realize the technical solution for a complete consist, together with the correct multiplicity of bus signals needed for the communication between the Function Carriers. The BT model contains similar information about the communication between the different subsystems (e.g. Brakes, Propulsion, etc.).

Step 5 (BT and KB): Align ICDs
After the instantiated models have been developed independently by both partners, the results need to be aligned. To this end, corresponding model elements of the instantiated models are mapped. This mapping will draw on the results of step 3, i.e. the mapping of the non-instantiated models will be refined adding information about multiplicity. During this process, inconsistencies in the interface definitions of the two partners become evident and can be corrected. This might even cause another cycle of the complete MBSE process if the inconsistencies can't be solved by just correcting the instantiations made in step 4.

Just as in step 3, the mapping results will be stored in a tool. This ensures traceability between the two models, allowing further cyclic refinement and simplifying change management. Due to slightly different modelling approaches on both sides, technical signals on the BT side will be mapped to technical ports on the KB side.

Based on this mapping, Interface Control Documents (ICDs) may be generated, combining the interface information of both models. These documents contain the corresponding interface elements with names following the respective naming conventions of either side. They represent the authoritative interface definition agreed upon by the two partners.

6 Conclusion

Railway market evolution is challenging the current status quo of how the companies working on this business are organized and are interacting with each other. Improving collaboration and communication to share the end goal and the related risks between rolling stock integrator and subsystems suppliers is becoming a must to remain competitive.

The extended enterprise approach answers to that need, but it requires some enablers, while MBSE can definitively be the one for the functional content of the product.

Unfortunately, the only available and widely adopted standards in the MBSE domain are the generic system modelling language SysML giving basic elements and diagrams to depict system structure and behaviour plus some generic modelling methodologies which remain quite abstract. No common industrial standard on modelling methodology for the railway industrial sector has been developed yet.

In some other engineering domains like software, there are coding standards like MISRA which give a guideline how to apply the language, or in other industrial sectors like automotive, standards like AUTOSAR (Automotive Open System Architecture) [1] have established an open and standardized software architecture for automotive electronic control units (ECUs), permitting the scalability to different vehicle and platform variants, transferability of software, the consideration of availability and safety requirements, as well as collaboration between various partners and maintainability throughout the whole “Product Life Cycle”.

The lack of such standards requires companies that want to implement an MBSE approach (like Bombardier and Knorr-Bremse) to develop by their own specific profiles of the SysML language and related modelling methodologies. As major consequences it’s very difficult on one hand to interlink models across companies, especially when different tool environments are used, and on the other hand to extend the usage on the functional models linking down to behavioural simulation environments like Modelica.

The case study presented in this paper is an exceptional lucky case where the lack of those standards hasn’t been impeding Bombardier and Knorr-Bremse to explore the feasibility of implementing a real MBSE based extended enterprise approach and to appreciate all the possible benefits of it. Both the companies strongly believe in MBSE and in the potentialities to enable a strong extended enterprise approach, and wish that an open modelling methodology standard will be developed providing a common MBSE framework across the railway industry.

References

1. AUTOSAR: Standards. https://www.autosar.org/standards. Accessed 10 July 2018
2. Delligatti, L.: SysML Distilled. Addison-Wesley, Upper Saddle River (2015)
3. Duffy, J., Tod, M.: The Extended Enterprise: Eliminating the Barriers. IDC
4. Friedenthal, S, Moore, A., Steiner, R.: A Practical Guide to SysML, 3rd edn. Morgan Kaufmann (2014)
5. Grady, J.O.: Universal architecture description framework. Syst. Eng. **12**(2) (2009)
6. International Council on Systems Engineering (INCOSE): Systems Engineering Vision 2020, version 2.03. INCOSE-TP-2004-004-02, Seattle (2007)
7. International Council on Systems Engineering (INCOSE): Systems Engineering Handbook, version 3.2.2. INCOSE-TP-2003-002-03.2. San Diego (2012)
8. International Organization for Standardization (ISO): Software and systems engineering – Reference model for product line engineering and management. ISO/IEC 26550 (2013)
9. Kang, K.C., et al.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Carnegie Mellon University, Technical report CMU/SEI-90-TR-2
10. Krob, D.: Eléments d'architecture des systèmes complexes. In: Gestion de la complexité et de l'information dans les grands systèmes critiques, Alain Appriou editor, CNRS (2009)
11. Object Management Group: OMG Systems Modeling Language (SysML), version 1.3, https://www.omg.org/spec/SysML/1.3. Accessed 10 July 2018
12. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering. Springer, Berlin (2005)
13. Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.): Model-Based Engineering of Embedded Systems. The SPES 2020 Methodology. Springer, Berlin (2012)
14. Rajan, A., Wahl, T. (eds.): CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems. Springer, Berlin (2013)
15. Ratiu, D., Schwitzer, W., Thyssen, J.: A System of Abstraction Layers for the Seamless Development of Embedded Software Systems. SPES 2020 Deliverable D1.2.A-2. Technische Universität München (2009)
16. Van Gaasbeek, J.R.: Model-Based System Engineering (MBSE), presented in the INCOSE L.A. Mini-Conference. INCOSE (2010)
17. Weilkiens, T.: Systems Engineering mit SysML/UML, 2nd edn. dpunkt.verlag (2008)
18. Wymore, A.W.: Model-Based Systems Engineering. CRC Press, Boca Raton (1993)

Model-Based System Reconfiguration: A Descriptive Study of Current Industrial Challenges

Lara Qasim[1,2](✉), Marija Jankovic[2], Sorin Olaru[3], and Jean-Luc Garnier[1]

[1] Thales Technical Directorate, 1 avenue Augustin Fresnel, 91767 Palaiseau Cedex, France
{lara.qasim,jean-luc.garnier}@thalesgroup.com
[2] Laboratoire Génie Industriel, CentraleSupelec, 3 rue Joliot Curie, 91190 Gif-sur-yvette, France
lara.qasim@centralesupelec.fr, marija.jankovic@ecp.fr
[3] Laboratoire de Signaux et Systemes, CentraleSupelec, 3 rue Joliot Curie, 91190 Gif-sur-yvette, France
sorin.olaru@centralesupelec.fr

Abstract. System Reconfiguration is essential in management of complex systems because it allows companies better flexibility and adaptability. System evolutions have to be managed in order to ensure system effectivity and efficiency through its whole lifecycle, in particular when it comes to complex systems that have decades of development and up to hundreds of years of usage. System Reconfiguration can be considered and deployed in different lifecycle phases. Two significant phases are considered for configuration management and System Reconfiguration: design-time – allowing system performances by modifying the architecture in early stages – and run-time – allowing optimization of performances during the in-service operations. This paper gives an overview of a field research currently ongoing to capture the strengths and the shortages in the current industrial landscape. It also discusses possible future management strategies with regard to identified issues and challenges.

1 Introduction

In today's competitive market, companies are concerned with developing systems effectively while reducing cost and time overruns. The primary objective of Systems Engineering is to develop systems that are operational regarding defined contexts and environments. Systems Engineering sustains complex systems activities with the aim to satisfy internal and external stakeholders requirements [1, 2]. Configuration management, technical management and life cycle model management are formalized as a set of processes during design-time and run-time for management of systems through their lifecycles [1–3].

"System configuration" is defined in Systems Engineering as a set of elements that compose a system in terms of hardware devices, software, interfaces, human profiles and processes [2]. System configuration is one of the important aspects addressed by the system management. The objective of system configuration management is to ensure effective management of an evolving system during its lifecycle [2]. System configuration can be characterized with regards to economic, environmental, legal, operational, behavioral, structural, and social aspects that are necessary to demonstrate a capability. As a counterpart, "System Reconfiguration" is defined in this paper by subsequent changes of the system configurations with the objective of maintaining or improving the capabilities provided by the system. System Reconfiguration is in particular necessary in two major system life-cycle phases: development (or "design-time") and in-service phase (or "run-time").

At design-time, one can identify several reasons for configuration changes. Systems may evolve to improve the performance by taking into account information coming from operational data. Changes are introduced to system configurations in order to correct errors and mismatches during the development, testing and deployment of the system. Stakeholders' requirements for system evolution may also drive to changes.

At run-time, the objective is to optimize system performances according to the context or the mission. Configuration optimization in terms of capabilities and available resources is needed to cope with environment or mission evolutions. In case of starvation of resources, the end-user needs must be dealt with by optimizing the remaining resources.

This paper aims at presenting current challenges that are related to system configuration based on industry research. Section 2 gives the state of the art of the domains related to System Reconfiguration. Section 3 describes the methodology used in this research. Section 4 presents the industrial challenges and the issues. Section 5 analyzes and discusses the industrial observation and also gives an insight into future works. Finally, Sect. 6 draws conclusions.

2 State of the Art

To keep the system functioning correctly, companies need to manage evolutions of system configurations. System Reconfiguration has been treated in several research domains with different levels of maturity. The concept of system configuration has been initially developed in the fault detection, isolation, and reconfiguration related studies. Reconfiguration in fault tolerant control (FTC) is mainly addressing the dysfunction of interconnected dynamic systems that are more or less complex (see Sect. 2.1 addressing both considerations). These two considerations shall be addressed in Systems Engineering and during configuration management at run-time (see Sect. 2.2). Configuration management at design-time can be yielding from different development activities with consideration of system configuration and evolution in the case of change management and propagation (see Sect. 2.3).

2.1 Fault Detection, Isolation and Reconfiguration

In operations, it is important to supervise and control component or sub-system operations, for example via a feedback loop, to maintain the system desired behavior. Once a fault in one component has been identified within a supervision activity, the control activities must react by reconfiguring the system to cope with these abnormal behaviors. Literature is addressing these concerns as fault detection, isolation, and reconfiguration or FTC. The primary purpose of FTC functionalities is to overcome the malfunctions while maintaining desirable stability and performance properties [4, 5].

Passive and active FTC functionalities exist, depending on their management of detected faults. Passive FTC functionalities sustain robust control activities that handle faults within a predefined quality of service. On the other hand, active FTC functionalities allow reaction to a detected fault and perform reconfiguration so that the stability and the performances can be maintained [6]. In active FTC functionalities, the fact that the control activities are reconfigurable means that one can adaptively address non-predefined faults.

A typical active FTC functionality relies on two fundamental mechanisms: fault detection and isolation (FDI) sometimes referred to as "fault diagnosis" [7], and reconfiguration control mechanisms (RC) [4]. The reconfiguration control aims at masking the fault either by switching to a redundant system/component or by revising the controller structure. In some cases, the available resources do not allow counteracting fault effects. In such cases, the best solution is to allow system degradation when the performance is accepted to be out of the optimal area [5].

There are different techniques used in fault detection and isolation. They are classified into model-based and data-based techniques [4]. Model-based techniques use system models to estimate the system states and parameters. Data-driven techniques, on the other hand, rely on classifiers and signal processing [4]. In this paper, the interest lies in changes and deviations in the system state addressed by model-based techniques while data-driven techniques fall out of interest.

Reiter [8], in his theory of diagnosis, proposes a method that requires a model describing the system. Given observations of a system, diagnosis compares the observed system with the expected behavior (modelled system) to determine the malfunctioning components. Reiter's theory has been extended to deal with the model-based diagnosis of different kinds of systems in different domains of applications [9, 10]. Identifying faults in malfunctioning systems is important but repairing these systems so that they can continue their missions is an essential problem to be addressed. Reiter's theory of model-based diagnosis has been extended to a theory of reconfiguration [11]. Much research has been conducted to use the model-based analysis concepts in the reconfiguration control design and analysis algorithms [12–15].

2.2 Configuration Management and System Adaptability

Adaptability can be understood by the fact that systems have to face continuously evolving situations and must be able to reconfigure their structure or their behavior to maximize their ability to accomplish required functions [16].

Moreover, research efforts have been conducted to design flexible and agile systems supported by a reconfiguration agent. Boardman and Sauser [17] define agility as the ability of the system to quickly detect and destroy unintended behaviors. In Systems Engineering, flexibility means the ability of the system to respond effectively and efficiently to potential internal or external changes [18]. AlSafi [19] proposes an approach based on reconfiguration software agents that allows manufacturing systems to adapt to changes in the manufacturing requirements and the environment by generating an alternative of a new feasible configuration. An approach of designing reconfigurable systems using a multi-agent system is described in [20]. A model-based oriented approach that supports the adaptation processes based on a run-time transformation of the system architecture is proposed in [21].

2.3 Change Prediction and Propagation in the Conceptual Design and Basic Engineering Phase

Most design activities can be considered as modifications of previous designs. New product or system development is then an incremental process involving modifications (changes) to existing designs where innovation and ideas are only used in some parts of the products while other parts remain relatively unchanged [22]. Literature underlines several strategies addressing system configuration and reconfiguration at design time including system modularity, reusability (Components of the shelf - COTS), system platform design and change management.

As products or systems are often based on past designs rather than designing from scratch, change management and change propagation are very important to manage the development of complex systems. Giffin [23] defines change propagation as the process where a change to one part of an existing system configuration can lead to changes that are not always wished.

To manage the risk of changes early in the design process, there is a need to assess which systems are most likely to be affected by a change, and what the impact of such a change would be. Clarkson [24] proposes a change prediction method to calculate the risk of direct and indirect change propagation. In [25], an efficient engineering change management (ECM) is presented as a key enabler for the agile product development of physical products.

After analysis of the academic state of the art, no research has currently been found on integrated approach aiming at describing reconfiguration process and actions for both design-time and run-time; even if such an approach should provide a global strategy to enhance flexibility, efficiency, reusability, modularity, and configurability.

3 Methodology

The research presented in this paper is action-based research [26]. This means that at least one of the researchers is also an engineer in Industry. This paper is addressing the first stages of this research. The research methodology (Fig. 1) is based on the exploration of current literature through the examination of papers supported by data collection. According to Blessing and Chakrabarti [27], observation and data gathering are essential to analyze and understand the industrial context and to propose a descriptive study that covers both empirical studies and their analysis to form new hypotheses.

In the first phase of this action-based research, the aim is identifying the current challenges in management of system configurations and overall System Reconfiguration process with regard to existing literature, including norms and standards. This

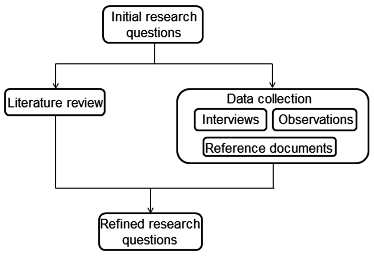


Fig. 1. Research methodology

research is based on triangulation of the data: interviews, direct observations and company reference documents. To understand where challenges, limitations, and opportunities lie, interviewing can be used to support engineering design research [28]. To ensure objectivity, the interview has been designed according to a structured list of questions. Objective of the interviews is to find out the definitions related to system management (including system configuration and System Reconfiguration), in terms of artefacts and processes that govern system or product life cycle activities, like Systems Engineering, manufacturing and in-service operations. Moreover, questions about the different methods and tools used in the configuration management and System Reconfiguration processes have been included in our survey.

For the interviews, 17 Thales experts have been identified with different levels of involvement in system management. Since Thales deals with different types of systems in various operational contexts, the identified persons are classified into two categories: people working in transversal activities and subject matter experts. Currently, interviewed persons belong to the first category. The results of these interviews are presented in Sect. 4.

4 Analysis of Current Industrial Challenges

The objective of this research is to propose integrated model-based support for instantiation of system configurations and System Reconfiguration addressing both design-time and run-time. In this context, a field study is conducted to identify existing data, process, issues and challenges in order to better understand the type of needed support.

With a systemic approach, System Reconfiguration is considered as a mean to improve the system performance and quality of service. An essential benefit of System Reconfiguration is to reduce the redundancy that comes along with different problems like increased cost, space limitations, weight, and high energy consumption. To allow system management, including instantiation of system configurations and reconfiguration, interviewed experts highlight the importance of concurrent element management, i.e., resources, functions, (Fig. 2) in different lifecycle phases of safety and security-critical systems. Reconfiguration allows improving systems in terms of performance and effectiveness to ensure an increased availability and a continuity of service.

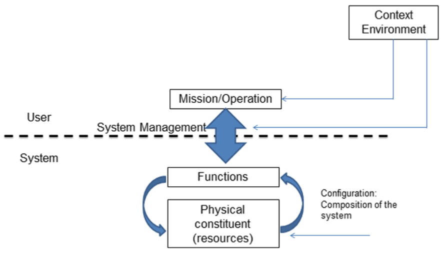

Context Environment
Mission/Operation
User
System Management
System
Functions
Physical constituent (resources)
Configuration: Composition of the system

Fig. 2. Optimization of the concurrent elements (resources, functions)

Experts interviewed during this research have emphasized the need to support systems evolution during each lifecycle phase. According to them, "Reconfiguration is an everyday question." For instance, during dismantling one should think of reconfiguration because dismantling a service or a product might have an impact on the overall services provided by the system. However, as for the perimeter considered in the enterprise, 2 phases of systems lifecycle seem to be critical: design and operations (Fig. 3).

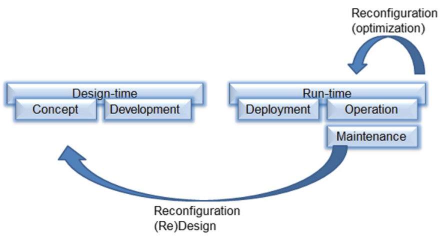


Fig. 3. Reconfiguration in different lifecycle stages

At design-time, a system has to be studied in terms of functions permitting to accomplish specific operational capabilities. Reconfiguration at design-time means optimizing the implemented resources to achieve capabilities demanded by the stakeholders. At deployment, before starting the mission, instantiation of a configuration aims at ensuring that the total resources provide the functions needed to accomplish the mission.

At run-time, the functions and resources essential for the mission are monitored as the reconfiguration process relies mainly on the awareness of the system state concerning the health of available resources with regards to the functional modes to be guaranteed. When a function becomes unavailable because of a faulty resource, reconfiguration may pick up the capability from other available resources (P1 to P2 in the example of Fig. 4). If the function is no more accessible, then reconfiguration aims at steering the system into a degraded mode (P3 to P4 in the example of Fig. 4).

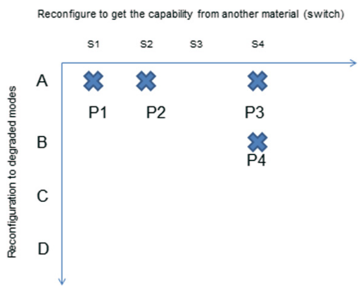

Reconfigure to get the capability from another material (switch)
S1
S2
S3
S4
Reconfiguration to degraded modes
A
B
C
D
P1
P2
P3
P4

Fig. 4. Reconfiguration during operation to optimize the resources

Configuration management and System Reconfiguration are needed at both design-time and run-time. These processes rely on different **data** and **models** to be collected, reused, generated and managed during the whole relevant system or product life cycle, according to formal and unformal **contracts** committed with stakeholders. Some challenges pertaining to configuration management and reconfiguration have been identified: challenges related to data, modeling issues, contracting and certification, system and context taxonomy. The following sub-sections discuss these issues.

4.1 Data Related Challenges

The instantiation of configurations and reconfiguration processes rely fundamentally on data. Data need to be collected and verified from day zero up to the retirement. Indeed, these data can originate from different phases of system or product lifecycle and can have different structure and management systems. This sub-section discusses the issues related to these data.

(a) Data availability and accessibility

Data availability and accessibility can be a real issue at the technical or operational levels. For instance, in some applications, data collection cannot be possible due to harsh working environments; for example, fuel rod temperature measurement within a nuclear reactor in operation. In other cases, measured data cannot be transmitted directly; therefore communication technologies are needed to give access to these data. However, when communicating data, one should take into account all the measures to secure these data. When dealing with operational data, privacy, integrity and confidentiality become a real issue. Consequently, secured data processing for strategic and tactical applications (e.g. military systems) may become an essential requirement.

(b) Data shared across stakeholders

Different stakeholders are involved in complex systems, such as system designers, developers, customers and end-users. In addition to that, a specific team can be responsible for the system at each of the system lifecycle stages. Therefore, it could be difficult to collect data shared between the different stakeholders because of complex organizational interfaces and intellectual property.

(c) Data storage

The quantity of data needed for instantiation of configurations and reconfiguration can be considerably depending on data saving strategy. In that case, data storage for instantiation of configurations and reconfiguration is also being considered as a critical issue. To avoid storing data continuously, front-end pre-processing can be implemented as a way to lessen data storage. In this case, pre-processing typically relies on detection of thresholds and hence the problem of threshold definition can arise.

(d) Uncertainty and data verification

Data necessary in the instantiation of configurations and reconfiguration process can have different sources. In particular, system monitoring data are collected from sensors installed either as external interface of the system or in embedded components. External observation of the system; from operators or maintenance teams, is also considered a source of data. Indeed, the uncertainty about the collected data is variable. Data verification is necessary to address this uncertainty. However, data verification is not evident because it is linked at least to the level of knowledge and completeness of the data.

(e) Data combination

Instantiation of configurations and reconfiguration rely on a priori and a posteriori data. These data need to be combined to allow instantiation of configurations and reconfiguration at both design-time and run-time. These data can be of different nature and are collected in one phase and later used in other phase of the life cycle. For example, during design-time, data from past systems can be used to modify the system design. At run-time, data from maintenance or the current operational situation can lead to system reorganization. Data combination is not trivial and requires in-depth data analysis to consider the degrees of uncertainty.

4.2 Modeling Issues

Different types of models can be used for the instantiation of configurations and reconfiguration. Depending on the system and its context, models can be continuous or discrete. Systems are built of constituent sub-systems leading to nested modeling. In instantiation of configurations and reconfiguration, different kinds of systems are involved: the system of interest (the system fulfilling the operational mission); but also enabling systems for development, manufacturing, maintenance, health monitoring, supervision and control. In order to achieve instantiation of configurations and reconfiguration, multi-level modeling is needed to combine and conjointly manage these levels. In addition to that, data to be modeled can be of different nature: internal (technical), external (environmental and operational). There is a difficulty in modeling these data. The modeling and analysis techniques need to be adapted to address different types of data.

4.3 Contracting and Certification

In the industry, developers and solution providers are usually concerned with contracting and certification. Contracts include information about usage profiles, configuration alternatives, operational contexts, quality of service, reliability, availability, safety, security, etc. The contracted configurations are tested, validated and certified in advance. However, when considering dynamic operation, new alternatives can be instantiated during operations; therefore they are not initially approved. The efforts needed to cope with this situation are not negligible because of lack of metrics required for the certification process. This activity may last for a long time leading to penalties due to a schedule overrun. Consequently, challenges related to certifying, assessing and selecting the newly emerging configurations have to be taken into account.

4.4 System and Context Taxonomy

The Thales Group develops a very large set of system and product types. Most of the time, these systems and products do not stand alone and are integrated into different operational contexts (e.g., platforms or infrastructures) and larger systems (or systems of systems). Consequently, the concerns related to these systems are extremely different. For example, reconfiguring a closed system (an assembly or a platform) is needed to propose new configurations while integrating new technologies. When considering distributed System Reconfiguration, there is a need to address the problem of connectivity between the system elements. Moreover, in a system of systems (SoS), constituent systems are mostly independent and this can lead to emerging effects. In addition to that, the interfaces between these constituent systems of a SoS may evolve. Consequently, the System Reconfiguration of SoS must be considered as an agile capability. The methods and mechanisms for System Reconfiguration might be slightly different according to the system types. Indeed, this variation leads to complexity when trying to build the support framework progressively with the aim of overall generalization across systems and industry. A holistic method for System Reconfiguration is an essential challenge because it must be as abstract as possible to adapt to system and context taxonomies.

5 Discussion and Future Work

This research seeks to build on previous work and aims at proposing an integrated approach for model-based instantiation of system configuration and reconfiguration addressing both design-time and run-time. In particular, the approach aims at using models for representing system configuration and health monitoring system to harness complexity. A system management framework will be studied in order to propose a configuration manager (Fig. 5) including an engine and a knowledge base. This knowledge base will contain models representing the system configurations and reconfiguration rules. The configuration engine will be in charge of applying relevant configurations.

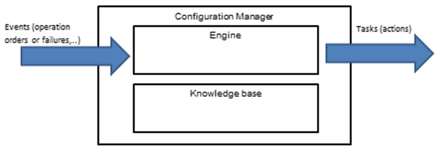


Fig. 5. Configuration manager implementation

As foreseen, instantiation of configurations and reconfiguration problems are at the intersection of different domains. Indeed, the knowledge base needs to include models related to technical, contractual and operational domains. When events occur, the engine part of the configuration manager generates reconfiguration actions. This process is possible by referring to policies included in the knowledge base as models.

Future work should concentrate on identifying models and data to be included in the knowledge base. Models and data can be clustered in two main categories: (1) models and data allowing reconfiguration (giving the system configurations with states and modes), (2) models and data for performance of reconfiguration (including rules and conditions).

This research seeks to address action models representing the mechanisms of reconfiguration in both phases: design and operation. Methods from complexity science sound promising to address reconfiguration in the design phase. Change prediction method can be used to predict how changes propagate in the system linkage model (Design structure matrix DSM) while addressing the risks related to propagation to allow evaluation of possible alternatives and redesign. The Model Predictive Control (MPC) can be used to address the on-line (run-time) optimization of systems. This method relies on enriching values for process input (feedback from operation), and the control action is determined as a result of the online optimization problem. Future work should focus on demonstrating the utility and underlying limitations of such methods. With regard to the challenges identified in Sect. 4, the application of the proposed approach within industry requires careful consideration.

A better understanding of the reconfiguration process and the characteristics of related data and mechanisms are necessary to improve the proposed approach.

In next steps of the research, interviews will be completed with case studies in order to capture the specifics related to system and context taxonomy. Attention is required to identify the mechanisms of reconfiguration for both design-time and run-time. This allows addressing life cycle processes in an overall manner with the goal of generalization across systems and Industry.

6 Conclusion

System configuration and System Reconfiguration are essential as they support system evolutions to ensure affectivity and efficiency of systems through their life cycles. This paper identifies current industrial challenges related to system configuration and System Reconfiguration at design-time (development) and run-time (in-service phase). This paper discusses identified challenges – with regard to the literature review and interviews of experts – and gives initial proposals to address the reconfiguration problem. Some difficulties such as modeling issues and system taxonomy have been identified. An initial reflection on configuration manager is presented. Future work addressing change prediction (CPM) and model predictive control (MPC) methods have been discussed allowing for definition of further actions.

References

1. ISO/IEC/IEEE/15288: Systems and software engineering–system life cycle processes (2015)
2. INCOSE: Systems engineering handbook: a guide for system life cycle processes and activities. In: Walden, D.D., Roedler, G.J., Forsberg, K., Hamelin, R.D., Shortell, T.M., (eds.) International Council on Systems Engineering, 4th edn. Wiley, San Diego (2015)
3. NASA: NASA Systems Engineering Handbook, vol. 6105 (2007)
4. Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems. Annu. Rev. Control **32**(2), 229–252 (2008)
5. Stoican, F., Olaru, S.: Set-Theoretic Fault Detection and Control Design for Multisensor Systems (2013)
6. Eterno, J., Weiss, J., Looze, D., Willsky, A.: Design issues for fault tolerant-restructurable aircraft control. In: 24th IEEE Conference on Decision and Control, pp. 900–905 (1985)
7. Isermann, R.: Supervision, fault-detection and fault-diagnosis methods–an introduction. Control Eng. Pract. **5**(5), 639–652 (1997)
8. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
9. Kuntz, F., Gaudan, S., Sannino, C., Laurent, É., Griffault, A., Point, G.: Model-based diagnosis for avionics systems using minimal cuts. In: Sachenbacher, M., Dressler, O., Hofbaur, M., (eds.) DX 2011, Oct 2011, pp. 138–145. Murnau, Germany (2011)
10. Ng, H.T.: Model-based, multiple fault diagnosis of time-varying, continuous physical devices. In: Proceedings 6th Conference on A. I. Applications, pp. 9–15 (1990)
11. Crow, J., Rushby, J.: Model-based reconfiguration: toward an integration with diagnosis. In: Proceedings of AAAI 1991, pp. 836–841 (1991)
12. Provan, G., Chen, Y.-L.: Model-based diagnosis and control reconfiguration for discrete event systems: an integrated approach. In: Proceedings of the 38th IEEE Conference on Decision and Control, vol. 2, pp. 1762–1768 (1999)
13. Russell, K.J., Broadwater, R.P.: Model-based automated reconfiguration for fault isolation and restoration. In: IEEE PES Innovative Smart Grid Technologies (ISGT), pp. 1–4 (2012)
14. Cui, Y., Shi, J., Wang, Z.: Backward reconfiguration management for modular avionic reconfigurable systems. IEEE Syst. J. **12**(1), 137–148 (2018)
15. Shan, S., Hou, Z.: Neural network NARMAX model based unmanned aircraft control surface reconfiguration. In: 9th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, pp. 154–157 (2016)
16. Ludwig, M., Farcet, N.: Evaluating enterprise architectures through executable models. In: Proceedings of the 15th International Command and Control Research and Technology Symposium (2010)
17. Boardman, J., Sauser, B.: System of systems–the meaning of of. In: 2006 IEEE/SMC International Conference on System of Systems Engineering, pp. 118–123 (2006)
18. Nilchiani, R., Hastings, D.E.: Measuring the value of flexibility in space systems: a six-element framework. Syst. Eng. **10**(1), 26–44 (2007)
19. Alsafi, Y., Vyatkin, V.: Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing. Robot. Comput. Integr. Manuf. **26**(4), 381–391 (2010)

20. Regulin, D., Schutz, D., Aicher, T., Vogel-Heuser, B.: Model based design of knowledge bases in multi agent systems for enabling automatic reconfiguration capabilities of material flow modules. In: IEEE International Conference on Automation Science and Engineering, pp. 133–140 (2016)
21. Rodriguez, I.B., Drira, K., Chassot, C., Jmaiel, M.: A model-based multi-level architectural reconfiguration applied to adaptability management in context-aware cooperative communication support systems. In: 2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA, pp. 353–356 (2009)
22. Otto, K., Wood, K.L.: Product design: techniques in reverse engineering and new product development, September 2014 (2001)
23. Giffin, M., de Weck, O.L., Bounova, G., Keller, R., Eckert, C., Clarkson, P.J.: Change propagation analysis in complex technical systems. ASME J. Mech. Des. **131**, 1–14 (2009)
24. Clarkson, P.J., Simons, C., Eckert, C.: Predicting change propagation in complex design. J. Mech. Des. **126**(5), 788 (2004)
25. Schuh, G., Riesener, M., Breunig, S.: Design for changeability: incorporating change propagation analysis in modular product platform design. Procedia CIRP **61**, 63–68 (2017)
26. Ottosson, S.: Participation action research. Technovation **23**(2), 87–94 (2003)
27. Blessing, L.T.M., Chakrabarti, A.: DRM, a Design Research Methodology, vol. 1 (2009)
28. Summers, J.D., Eckert, C.M.: Design research methods: interviewing. In: Workshop in ASME Conference 2013, Portland, Oregan, USA (2013)

A Domain Model-Centric Approach for the Development of Large-Scale Office Lighting Systems

Richard Doornbos[1](✉), Bas Huijbrechts[1], Jack Sleuters[1], Jacques Verriet[1], Kristina Ševo[2], and Mark Verberkt[2]

[1] Embedded Systems Innovation (ESI), TNO, Eindhoven, The Netherlands
richard.doornbos@tno.nl
[2] Professional Lighting Systems, Signify (Formerly Philips Lighting), Eindhoven, The Netherlands

Abstract. The high-tech system industry faces many challenges, such as continuously increasing system complexity, scale and customer demands. We address these challenges using a domain model-centric approach. This approach focuses on clear and formal system specifications, connected to a chain of automatic transformations for system analysis, including virtual prototyping, and system synthesis, e.g. code generation. We have applied the approach to the development of large-scale office lighting systems in order to reduce development effort and handle the complexity of system control.

1 Introduction

The development of high-tech systems is challenged by the ever-increasing complexity, signified by various trends such as the integration of IT, mobile and embedded systems into smart cyber-physical systems. The scale of systems increases tremendously. It is not exceptional for a high-tech system to have hundreds or thousands of processing units, more than 100 million lines of code, and requiring many design documents. A second major trend is the increasing speed of development, required by rapidly changing business models and increasing customization demands. The industry is trying to cope with these trends by adopting continuous engineering, and model-based engineering approaches.

In system development exchanging knowledge and information is crucial. Mainly written documents are used to convey in-depth technical information; these are static, and often in practice incomplete and out-of-date. Especially for the specification of dynamic behavior, textual documents are often vague and ambiguous [1]. A typical source of confusion is the description of the required system behavior (in the problem domain) using knowledge and terminology from the solution domain. Next to that, in our experience often only happy flow behavior is described.

An important assumption in developing systems is that requirements and early specifications correctly describe what is needed. However, there are two complications in industrial practice: uncertainty what the specifications actually should be, and secondly, there is ambiguity due to lack of formality in the specifications [2]. Therefore, the correctness of the specifications is very difficult to assess. Only in the later stages of development, system elements will be implemented and can be checked and tested. However, in practice system-level testing is often immature, and very limited due to time pressure.

In general, errors are introduced during the manual steps from specification to coding and configuring high-tech systems, impacting the quality of the product. Especially in testing software, finding errors (and their solutions) may prove time consuming [3].

From a business point of view these delays in development lead to an increase in time to market for products, which is a definite adverse effect. The perceived quality of products will strongly be influenced by the occurrence of failures in the field, which are the result of missed faults in development, installation or commissioning phases. In many cases, repair, reconfiguration, or even recall of products in the field has proven to be extremely expensive [3]. Obviously, the resulting low overall development efficiency will lead to a high development cost of the product.

2 Domain Model-Centric Approach

Given the mentioned issues and challenges, we aim at devising an approach to improve the development process in a fundamental way. The two main goals are: (1) reduce development effort and (2) handle complexity of control.

The aim is to reduce the effort for the full lifecycle: specification, development, validation, installation, commissioning, upgrade, etc. A specific goal is to replace the costly on-site testing of a physical system by off-site analysis of *virtual prototypes*.

Our focus is on the improvement of the specification process and the subsequent processes during development. This leads to several benefits, such as easy validation and verification of specifications and proposed solutions, and alignment and harmonization of operations along the development path (e.g. handovers between development phases).

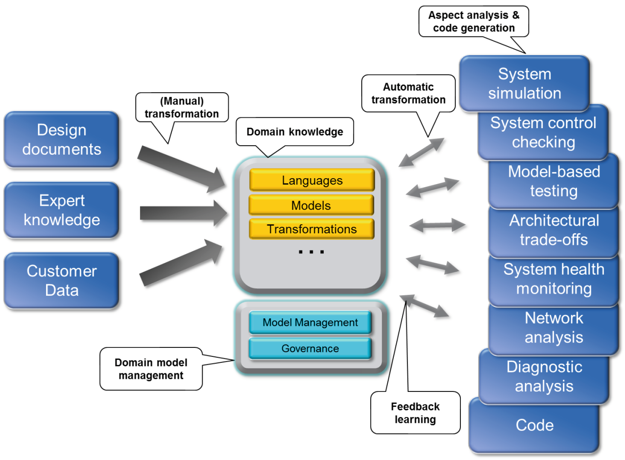


Fig. 1. High-level diagram of the domain model-centric approach. DSLs form the central domain model for system specification and information.

The basic idea behind the domain model-centric approach is the use of a set of languages (capturing the stakeholders' terminology) that allow specifying system information. In essence these languages and the specifications expressed in these languages replace the documents that currently capture the knowledge of experts and system information, as depicted on the left side of Fig. 1.

2.1 Domain-Specific Languages for System Specification

Domain-specific languages (DSLs) are a well-known way to define and separate system concerns [4–6]. The approach aims at a precise, fully declarative system specification; this requires a collection of DSLs, each of which describes one aspect of a system. These DSLs will provide for each stakeholder a natural and comprehensible way to describe the concerns, requirements and constraints with respect to the system. For instance, an office lighting system's topology structure of physical devices is described independently of the system's behavior.

In defining a DSL it is crucial to restrict oneself to the essentials, thus keeping the language concise. This makes very clear what is allowed and what not, therefore a lot of discussions are avoided. Also, the relations between DSLs have to be defined to enable cooperation and handover of work artefacts. This is usually done by defining automatic transformations, in which the semantics become explicit.

Next to DSLs specifying the system, the approach requires DSLs specifying the system's environment. These DSLs describe the interaction of a system with its users and other environmental interactions. For instance, for lighting systems, these DSLs describe how sensors are being triggered by the building's inhabitants and the natural light coming through windows. To allow tests of realistic system behavior, the environment is described in terms of usage profiles that are calibrated by data collected in existing buildings.

Having a formal specification enables powerful and advanced features:

- Automatic and aspect-specific checking of specifications. This is typically built into the language editor used to create a specification. Given the DSL and the checking rules, this language editor is usually automatically generated by the tooling.
- Automatic transformations to various aspect models and targets, see Fig. 1. Two main outlets exist: analysis and synthesis. For analysis purposes a variety of checks, simulator files, etc. can be generated. For synthesis purposes, the same (analyzed) system information can be used to generate code, configurations, tests, etc.

2.2 Aspect Analysis and Virtual Prototyping

In virtual prototyping a system model is created that exhibits all relevant features of a real prototype, except its physical presence. It is a computational model that allows exploitation of the flexibility of models: one can investigate performance, functional behavior, and other system characteristics with changing scale, different components, settings, and features by simply changing the DSL-based system specifications.

A specification of a system and its environment allows a system's behavior and performance to be analyzed by transforming the specification into a simulation model. Two kinds of simulation can be considered: *closed simulations* that fully capture the system's environment to analyze certain cases (e.g. for testing specific scenarios) and *interactive simulations* that allow for more explorative analysis (e.g. a user may trigger the system and observe its response). A visualization of the simulation state has proved to be very helpful to understand system behavior, especially for the interactive simulation case. Clearly, it is also very beneficial in the communication to stakeholders, especially to customers.

Next to simulation, system specifications can also be used in model checking. Where simulation allows a single scenario to be analyzed, model checking offers exhaustive analysis of all possible scenarios. Summarizing, simulation and model-checking can be used to guarantee desired system behavior or to identify unlikely failure scenarios already early in the system development (which saves costs).

2.3 System Configuration and Code Generation

To create an implementation, the (validated) system specification can be used in the transformations to generate code, configurations, test scenarios, etc. In these transformations usually a template implementation is adapted at certain variation points. In these variation points the specification information is inserted.

2.4 Industrial Application

It is important to know when to apply such a domain model-centric approach [7]. We have identified a few criteria that will help to decide. (1) DSLs can best be applied in the domains where knowledge is stable and mature. Continuously changing domain knowledge leads to continuously updating DSLs and transformations, which typically involves a large effort. (2) There should be a reasonable amount of reuse, over components, variants, system configurations, deployment environments, or products. (3) Having not too many variants (e.g. system component types, device types, control rules) is helpful; commonality increases the value of the approach. With these generic criteria the approach seems to be applicable to many industries: telecom, transportation, logistics, lighting, energy, and more.

The following list of concrete steps summarizes the approach:

1. *Identification of specification needs:* (a) Identify the goals and intentions in the development organization (e.g. show behavior to clients, check system KPIs). (b) Create overview of the overall development workflow and its stakeholders. Identify aspects, aspect interactions, roles, etc.
2. *Definition of aspects:* (a) Define for each aspect the relevant information and concepts, and the required transformation targets. (b) Develop for each aspect the domain languages and transformations.

3. *Analysis*: (a) Create specifications using the languages, typically starting with simple ones and gradually increasing in complexity and size. (b) Validate the specification of the system statically (performed by the language editor). (c) Create and validate the transformations. (d) Perform the transformations to the various analysis targets. (e) Validate system behavior dynamically in various ways: simulation, model checking, model-based testing, etc.
4. *Synthesis:* When step 3 has proved successful one can safely generate system configurations or code. (a) Create and validate the transformations. (b) Transform specifications to the various synthesis targets. (c) Use the generated artefacts in the final product environment. (d) As a last step usually acceptance tests of the generated system are performed.
5. *Feedback and improvement*: Wider introduction in the organization and further usage of this approach leads to a secondary process of feedback and improvement of the specification process, languages and transformations. It may become part of existing improvement activities within the organization.

The introduction of the domain model-centric approach leads to new ways of working and new functional roles. We separate language creation from language usage, as there are clearly different capabilities involved. The *language creators* perform 'language engineering' and should be able to isolate domain concepts and capture them into abstract language elements. The experts working in the company will be heavily involved in this process to provide their domain knowledge, and direct the language and its usage. The users of the languages, mainly the *system specifiers*, are typically focusing on one aspect, or one domain.

The *transformation experts* create the bridge between the abstract specification and the target, therefore they should be able to span this gap. The users of the transformation results can be split into *system analyzers* (e.g. system architects) and *system implementers* (e.g. engineers). For the latter, work may change significantly as many implementation activities become automated.

Of course, the languages, specifications, and transformation code as well as tools, need to be versioned and maintained [8, 9]. This may require a dedicated role: e.g. *modeling manager*.

3 Office Lighting

A smart indoor lighting system (e.g. for an office building) is a prime example of a complex system as described in the Introduction: it is a large-scale distributed system, containing thousands of sensor, controller, and actuator components, connected via a network. The system can have very many configurations, but is comprised of only a limited number of types of components (see Fig. 2). Typical for modern lighting systems is the complicating factor of having to cooperate with other systems (HVAC, network, power, security, building and cloud services), often leading to conflicting requirements.

To complicate things further, office lighting systems have a lifespan of several decades. Because market requirements and legislation change over time, lighting

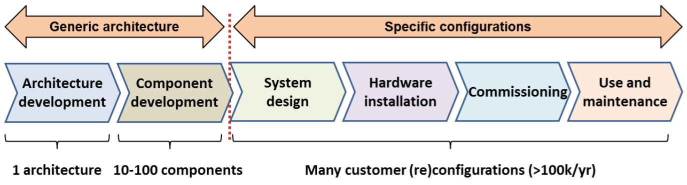


Fig. 2. Development process in the (office) lighting area.

systems go through multiple development cycles consisting of system design, installation, commissioning (i.e. software configuration), and operational use (see Fig. 2). *Installation* involves the set-up of a lighting system's physical components in its environment (e.g. an office building) which is a very costly phase in the lifecycle. *Commissioning* involves the configuring of the system's software to guarantee the required system behavior, which currently requires a lot of manual testing. Detecting and solving errors in this phase is expensive, especially because this is done on-site [3, 10]. In *maintenance* one has to deal with the complexity of component stocking and compatibility issues.

The key requirements for an office lighting system are: low response times, high synchronicity, low energy usage, and high availability. The first three requirements are typically addressed in the architecting and design phases. To fulfil the availability requirement, errors that are either made during installation and commissioning or occur during operational use, should be detected, diagnosed, and repaired as quickly and effortlessly as possible.

3.1 Office Lighting Systems

The lighting systems we consider in this paper are deployed in large office buildings (e.g. 2 wings, 15 floors), consisting of up to 10,000 luminaires (light points). These luminaires each contain a LED group (that acts as one light point), zero or more sensors (light intensity, presence, etc.) and a controller board. A luminaire uses a wired or wireless network connection to communicate data or control messages.

The control setup of the lighting system consists of 100's to 1,000's of controlled areas in the building, that interact with each other. The behavior of one area, which contains multiple luminaires, is usually defined in a number of so-called 'scenes' (e.g. 10 different scenes, designated as 'concentration', 'presentation', 'relaxation', etc.). Each scene has a specified light intensity, color, timing, intra-scene behavior (e.g. daylight compensation) and scene-to-scene transition behavior for each luminaire. The control configuration depends on (1) the topology of the building (location of rooms and corridors, but also windows and doors), (2) the functional assignments of the rooms and corridors, and (3) specific user wishes. The lighting behavior is determined by triggers originating from manual switches, presence of users (in the current or neighboring areas), daylight, and the control schedule system.

The complexity of office lighting control originates from the large number of luminaires, the physical distribution, the control interactions and the interaction with other systems.

The development challenges of office lighting systems can be distinguished for the various phases of the life cycle. Some typical questions during system development are: How to specify system behavior in the language of the end user, and how to translate it towards test cases? How to prove the correctness of the lighting behavior for a given building *before* the actual installation in the building? How to determine the system performance, e.g. energy usage, response times, *before* installation? Can we perform design space exploration to optimize system performance, and find the best system configuration for a specific customer?

For the operational phase there are different challenges: Can we add new features to the system, when the system is already deployed and operational, and be sure the addition will not break current behavior? How to easily reconfigure the system to address changes in its usage or environment?

4 Application in Office Lighting

For office lighting systems, the domain model-centric approach is very beneficial since there is a vast amount of system configurations, and these systems are built up from a small set of component types. It is clear that for lighting systems there is a significant amount of inherent reuse.

In the application of our approach we have focused on the control behavior. We have addressed four main outlets (see Fig. 3): (1) the static validation of specifications, (2) the dynamic validation of system behavior, (3) analysis of system performance, and (4) system configuration and code generation.

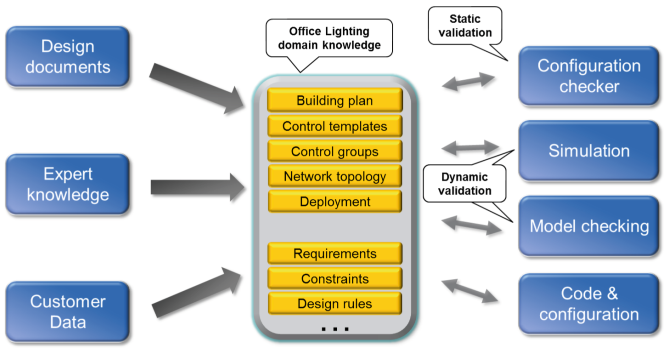


Fig. 3. A set of office lighting specific languages form the central domain knowledge.

We have performed the steps mentioned in Sect. 2.4, and we will focus in the remainder of this section only on the notable results.

4.1 Language Modularity

A strong complexity-reducing feature is the modularity in our office lighting languages. The separation is made along the various aspects identified for these systems, see Fig. 4.

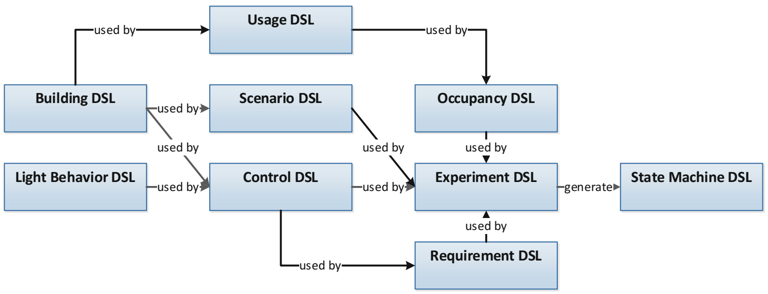


Fig. 4. The set of languages describing the lighting control domain.

The *Light Behavior DSL* describes the lighting behavior of the system, which can be applied to rooms with any number of luminaires. The behavior is parameterized in scenes; some examples: ‘concentration’, ‘presentation’, ‘relaxation’. The dynamic behavior is expressed in terms of a set of lighting-specific state machines with sensor and actuator groups.

The building in which the lighting system is to be deployed is specified in the *Building DSL*. This DSL describes the site, its topology (rooms, corridors, staircases, etc.) and the locations of lighting equipment (luminaires, buttons); information typically available in building information models (BIMs) [11].

The *Control DSL* is a specification that binds the actual building model to the light behavior specification. It maps the behavior specification onto rooms and corridors. With this specification it is clear which sensors and actuators are involved in a given control structure.

As mentioned in Sect. 2.1 the environment specification is needed for the analysis activities. The *Scenario DSL* allows the specification of simple usage scenarios; it describes sensor events occurring in specified sensors at given moments in time. For more elaborate scenarios, we have defined an *Occupancy DSL* that describes the user’s activities, and their mapping on locations in the building. The interaction of the environment with the system is (obviously via sensors and buttons) therefore specified. One level of automation was added by the definition of a *Usage DSL,* which describes user profiles. The automatic transformation to Occupancy specifications allows to generate many office user instances, to be deployed in the virtual prototype.

The *Requirement DSL* describes system-level properties that have to be verified in the analysis step. This language consists of a generic and an office lighting specific part [12], allowing the generic part of the language to be reused in other business areas.

The *Experiment DSL* couples all the mentioned specifications needed for analysis. It does not describe details about the analysis as such.

The *State Machine DSL* describes the state machines used to express the behavior of sensors, controllers, actuators, and office users. This generic language is fully lighting independent, and is a starting point for transformations towards analysis models, see Fig. 1.

We have used Xtext in combination with Xtend in an Eclipse development environment as language technology [13]. This choice is not critical, but in our experience it is easy to use, has powerful features, and has a large community for development and support.

4.2 Virtual Prototyping

As mentioned in Sect. 2.2, the system behavior and performance can be analyzed by transforming the specification into a simulation model. The *interactive simulation* allows a user to trigger the system and observe its response in a dedicated visualization, see Fig. 5.

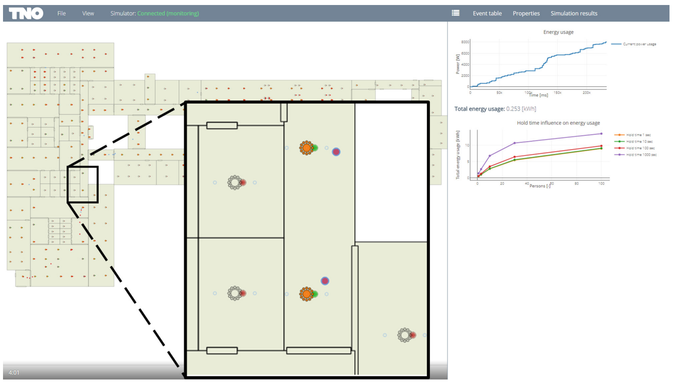


Fig. 5. Interactive lighting system visualization showing the floor layout. The inset zooms in on (orange) light points, (red, green when activated) sensors, and (purple) office users. On the right energy usage graphs are shown.

We implemented the simulation via a Java-based co-simulation framework, which couples the simulators of sensors, controllers, actuators, and environment. An energy monitor and the system visualization were also connected. Using this modular framework, models of actual (or future) buildings and their lighting system can be created and investigated *at full scale*. This allows analyzing system aspects such as KPIs and other performance parameters (e.g. latency), scalability, resource usage, functional correctness, and feature interaction.

As an example of such architectural investigation, we show the results of our analysis about the dependency of the energy usage on the hold time parameter (the time the light in a room stays on after leaving the room). This dependency changes with the number of persons in the building. For this analysis we used the information of one floor of a real building, containing 367 luminaires and more than 1300 behavioral functions. In Fig. 6 the curves are shown, created by 20 simulations.

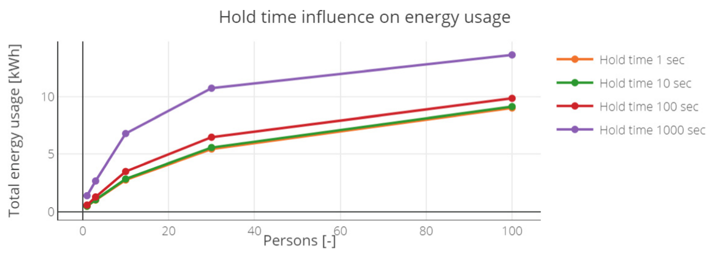


Fig. 6. Simulation results of the energy usage of the entire floor for various hold times and occupancy levels. Each experiment simulates one hour of typical behavior, the energy values are not calibrated.

This investigation provides valuable new insights to make trade-offs on acceptable hold time settings and energy usage. In contrast, performing this type of investigations with installed systems is simply too time-consuming, too costly, and too late.

Another virtual prototype application is to find behavior anomalies and their causes. Simulating system behavior is in general not sufficient, especially for systems with large state spaces. Therefore, we generate input models for a model-checker (in our case we use Uppaal [14]). This enables us to find unwanted behavior in the large number of parallel running systems. Fortunately, the scope of the typical behaviors is limited to a small number of rooms, so we can avoid state space explosions. The mentioned Requirement DSL is used to express the wanted behaviors. These behavioral properties are transformed into monitor automata that are added to the modelled system. If a violation of a specified system property is detected by the monitors, a failure trace will be available to diagnose the violation [12].

4.3 Generation of Configuration Settings and Code

We have generated configuration settings for hardware (firmware) for three different Philips Lighting product implementations and have demonstrated the value of the approach. The automatic transformations were using specification data to insert at certain locations in existing code fragments. This approach improved the efficiency and flexibility of the development process: a significant speed up, very limited manual coding, more flexibility in configuring, and certainty of generating the correct configuration.

5 Discussion and Conclusions

The advantages of the domain model-centric approach occur in various areas: organizational, business, architectural, and technical. Having formal and unambiguous system specifications leads to a more predictable development process. Note that the behavior specification only has to be made once and can be deployed across multiple product solutions. A shared ‘domain language’ connects engineers from different domains and leads to less misunderstanding. In most cases it leads to a simpler, more user-centric but formal specification. Another advantage is that the marketing department is enabled to show product characteristics and behavior via virtual prototypes in the customer’s environment.

On the technical side it is very important to have clear specifications in an understandable, solution-independent language. The enabled automation leads to a significant speed up, via automatic checks, code generation, and generation of virtual prototypes. Design-time simulation and analysis provides a tremendous risk reduction by more reuse of (proven) building blocks, and easy architectural exploration of novel solutions. Less on-site commissioning work is needed as better off-site checks are performed, leading to considerable cost savings.

Summarizing, the domain model-centric approach is a key step towards virtualization of product development. It allows model-based architectural validation and provides early insight in consequences of architectural decisions and configuration settings. It provides better control over system complexity and reduces significant cost and effort in system development in industrial practice.

Acknowledgement. The research is carried out as part of the Prisma programme and H2020 OpenAIS project under the responsibility of Embedded Systems Innovation (ESI) with Philips Lighting as the carrying industrial partner. The Prisma programme is supported by the Netherlands Ministry of Economic Affairs, the OpenAIS project is co-funded by the Horizon 2020 Framework Programme of the European Union under grant agreement number 644332 and the Netherlands Organisation for Applied Scientific Research TNO.

References

1. Akesson, B., Hooman, J., Dekker, R., Ekkelkamp, W., Stottelaar, B.: Pain-mitigation techniques for model-based engineering using domain-specific languages. In: Proceedings of MOMA3N 2018 (2018)
2. Hooman, J.: Industrial application of formal models generated from domain specific Languages. In: Theory and Practice of Formal Methods, pp 277–293 (2016)
3. Westland, J.C.: The cost of errors in software development: evidence from industry. J. Syst. and Softw. **62**, 1–9 (2002)
4. Mooij, A.J., Hooman, J., Albers, R.: Gaining industrial confidence for the introduction of domain-specific languages. In: 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW) (2013)
5. Schuts, M., Hooman, J.: Industrial Application of domain specific languages combined with formal techniques. In: Proceedings of Workshop on Real World Domain Specific Languages, The International Symposium on Code Generation and Optimization, pp. 2:1–2:8 (2016)
6. Bettini, L.: Implementing Domain-Specific Languages with Xtext and Xtend. Packt Publishing Ltd., Birmingham (2016)
7. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, Boston (2004)

8. INCOSE: Systems engineering handbook: a guide for system life cycle processes and activities, version 3.2.1. International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2, San Diego, CA, USA (2012)
9. Bernstein, P.A.: Applying model management to classical meta data problems. In: Proceedings of the 2003 CIDR Conference (2003)
10. Stecklein, J.M., Dabney, J., Dick, B., Haskins, B., Lovell, R., Moroney, G.: Error cost escalation through the project life cycle. In: Proceedings of the 14th INCOSE Annual International Symposium, June 2014
11. Eastman, C., Teicholz, P., Sacks, R., Liston, K.: BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. Wiley (2011)
12. Buit, L.J.: Developing an Easy-to-Use Query Language for Verification of Lighting Systems. Master’s thesis (http://essay.utwente.nl/74020/), University of Twente (2017)
13. Mooij, A.J., Hooman, J.: Creating a Domain Specific Language (DSL) with Xtext. http://www.cs.kun.nl/J.Hooman/DSL, ESI/Radboud University (2017)
14. Uppaal. http://www.uppaal.org/

Through a Glass, Darkly? Taking a Network Perspective on System-of-Systems Architectures

Matthew Potts[1(✉)], Pia Sartor[1], Angus Johnson[2], and Seth Bullock[1]

[1] University of Bristol, Faculty of Engineering, Bristol, UK
{matt.potts, pia.sartor, seth.bullock}@bristol.ac.uk
[2] Thales Research and Technology UK, Reading, UK
Angus.Johnson@uk.thalesgroup.com

Abstract. A system-of-systems architecture can be thought of as a complex *network* comprising a set of entities of different types, connected together by a set of relationships, also of different types. A systems architect might attempt to make use of the analytic tools associated with *network science* when evaluating such architectures, anticipating that taking a "network perspective" might offer insights into their structure. However, taking a network perspective on real-world system-of-systems architectures is fraught with challenges. The relationship between the architecture and a network representation can be overly simplistic, meaning that network-theoretic models can struggle to respect, *inter alia*, the heterogeneity of system entities and their relationships, the richness of their behavior, and the vital role of context in an architecture. A more mature conceptualization of the relationship between architectures and their network representations is required before the lens of network science can offer a usefully clear view of architecture properties.

1 Introduction

System designers are increasingly faced with the challenge of architecting complex systems that may have to operate within the context of a wider System of Systems (SoS). It can be natural to think of an SoS architecture as a network of entities connected together by relationships, and to regard this network as complex in the sense that it exhibits interesting structure. Architects require tools that can support architecture analysis. Network science provides a toolbox of techniques developed to shed light on complex networks of all kinds. However, in applying networks science analyses to real-world architectures, several challenges and limitations become clear. The purpose of this paper is to highlight these challenges and suggest ways to help bridge the gap between academic theory and industrial practice.

The paper is structured as follows; first a brief summary of prior work applying network science techniques to real-world SoS architectures is provided before cautions surrounding the use of graph-theoretic approaches to assess such architectures are described. These address difficulties in correctly identifying important architectural entities, evaluating architecture structure, and anticipating the response of architectures

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 121–132, 2019.
https://doi.org/10.1007/978-3-030-04209-7_10

to perturbations, but also identify more general conceptual challenges in aligning a network perspective with an understanding of systems architecting. The paper then discusses challenges that may impede progress towards more effective graph-theoretic evaluations of architectures, suggesting open research questions for the community.

1.1 Prior Work

Prior work [1] considered a network perspective on a Search and Rescue (SAR) Architecture developed by Thales and Airbus in order to inform systems architecture training and inform the development of the NATO Architecture Framework (NAF) v4 [2]. Several Architecture Views (AVs) were modelled as a directed graph of vertices, each of which represented a capability, service or physical entity, connected by edges representing communications connectivity and dependencies. The modelled graph was interrogated to determine which entities were the most important in the architecture by examining the degree (the number of connections in and out) of each vertex. The structure of the architecture was also evaluated in terms of edge density, D, a metric employed as an indicator of the likely integration and dependency management challenge presented by the architecture, and calculated as per Eq. (1), where E is the number of edges in the graph and V is the number of vertices.

$$D = \frac{|E|}{|V|(|V| - 1)} \tag{1}$$

The structure of the architecture was also evaluated in terms of strongly connected components (groupings of entities within which a path exists linking each pair of entities in both directions [3]) to reveal the presence of a core and periphery structure. Finally, a community detection algorithm [4] was used to suggest an alternative approach to partitioning a complex SoS architecture by grouping entities into communities within which connectivity was relatively strong by comparison with connectivity to entities outside the community.

Here we briefly report additional results drawn from the analysis of a second use case, a tactical military communications enterprise architecture (MComms). The architecture was created in accordance with the Ministry of Defence Architecture Framework (MODAF) [5], to enable Thales and their customer to have a shared understanding of the complex environment within which a tactical military communications solution would have to interoperate. The MComms use case describes the challenge of enabling effective tactical communications between soldiers. Systems, services, functions, artefacts (components), software and capability configurations were modelled as vertices and the communications and dependency relationships, such as systems fulfilling functions and hierarchical compositions, modelled as edges in a directed graph.

In the following section we use these two real-world use cases to highlight the challenges facing an architect trying to take advantage of a network perspective.

2 Cautioning Network Perspectives on System Architectures

2.1 Identification of Important Entities

A network perspective on complex SoS architectures might be expected to enable the identification of important entities in an architecture. Perhaps highly connected entities could be expected to play a crucial role, or entities that are in some sense located in the "core" of the system. Alternatively, an entity that is located such that it mediates between many parts of an architecture might be identified as a key asset. Finally, entities located such that they dominate flows of resource in a network might be assumed to be key. The implication here is that an architect must be clear on what constitutes importance for their architecture – there is no single network science metric or concept that will do this job for the architect. Important entities may be those that are geometrically close to many other entities in a graph, according to metrics such as Closeness, Harmonic Closeness and Betweenness Centrality, or those that benefit from favourable connectivity, according to measures such as Degree and Eigenvector Centrality. It is beyond the scope of this paper to detail the plethora of metrics available, an interested reader is directed to [6], instead centrality measures used selected that several authors used to determine which entities are important in an architecture [7–10], often using one with no consideration of other alternatives.

The Closeness Centrality, C_i, of a vertex i is the reciprocal of the sum of the distances of the shortest paths from i to all other $n-1$ vertices in the graph, normalized (to enable comparison between architectures with different numbers of vertices) by multiplying this reciprocal by the sum of the minimum possible distances, $n-1$ [11].

$$C_i = \frac{n-1}{\sum_j d(j,i)} \tag{2}$$

The Harmonic Closeness Centrality, H_i, of a vertex i is the sum of the reciprocal of the shortest path distances,$d(j,i)$, from i to all other $n-1$ vertices in the graph (Eq. (3)) [12].

$$H_i = \sum_{j \neq i} \frac{1}{d(j,i)} \tag{3}$$

The Betweeness Centrality, B_i, of a vertex i is given by Eq. (4) where σ_{st} is the total number of shortest paths from vertex s to vertex t and $\sigma_{st}(i)$ is the number of those paths that pass through i [13].

$$B_i = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{4}$$

The Eigenvector Centrality,E_i, is given by Eq. (5) where $a_{j,i}$ is the adjacency matrix of the graph (the adjacency matrix is a square matrix representing the graph, where $a_{j,i} = 1$ if vertex j is connected to vertex i and $a_{j,i} = 0$ otherwise) and $\lambda_{\neq 0}$ is a constant [14].

$$E_i = \frac{1}{\lambda}\sum\nolimits_{j \in G} a_{j,i} \tag{5}$$

While each of these measures can be applied to a network representing a system architecture, there are three main concerns when using them to determine important entities within it. First, there are practical limitations in their calculation. Second, they do not agree on which entities are most important. Finally, and most importantly, they neglect the rich context inherent to a complex SoS architecture, potentially allowing an architect to be misled by a numerical analysis that ignores more significant determinants of importance.

A limitation in calculating these measures is the treatment of disconnected entities and entities with connectivity in only one direction (e.g., a node that has one incoming edge from its sole neighbor but no out-going edges linking it to the network). Both the SAR and MComms architectures have a large number of entities with relationships in only one direction (e.g., node i depends on node j, but not vice versa) which results in many entities being assigned an importance score of zero by some of the network metrics, due to the effectively "infinite" distance between the node and parts of the network that it cannot reach. Calculating the shortest path distances, $d(j,i)$, takes account of directionality in a directed graph, noting it is *from i to* all other $n - 1$ vertices. For example, 74% of the entities in the MComms architecture are given a Closeness Centrality of zero and 82% are given a Betweenness Centrality of zero, similarly the 27% of the SAR entities have an Eigenvector Centrality score of zero. It may be the case that high scoring entities are the most important within an architecture, but the inability of the measures to cope with the directed, tree-like structure of the networks is not encouraging. Confidence in these results can only be obtained through further analysis of what each centrality scores measure, and how this corresponds to a notion of importance that makes sense to an architect. Rather than expecting a single network science measure to identify key entities, a more useful approach for an architect would be to iterate between two questions: 'what makes an entity important in this complex SoS architecture and how might that be reflected in the network representation of it?' and 'what network properties are these measures capturing and what are their implications for my understanding of the architecture itself?'. Note that rather than conflate the network and the architecture together as though they are the same thing, these questions explicitly separate the architecture (the thing we are interested in) from the network (a particular abstraction of that thing that may help us understand it).

Unsurprisingly, there is little agreement between the centrality measures regarding which entities they identify as most important (Fig. 1). This reinforces the point that caution must be exercised in their use. Despite employing similar concepts such as geometric distance or connectivity, there is little significant correlation between them.

Most importantly, however, it may be that network measures of the type described above are not capable of tracking importance in a sense that is relevant to a system architect's needs. For example, in the case of the SAR and MComms architectures, what makes an entity important may be less to do with its location in the network and more to do with its contribution to mission effectiveness. Unless this contribution is

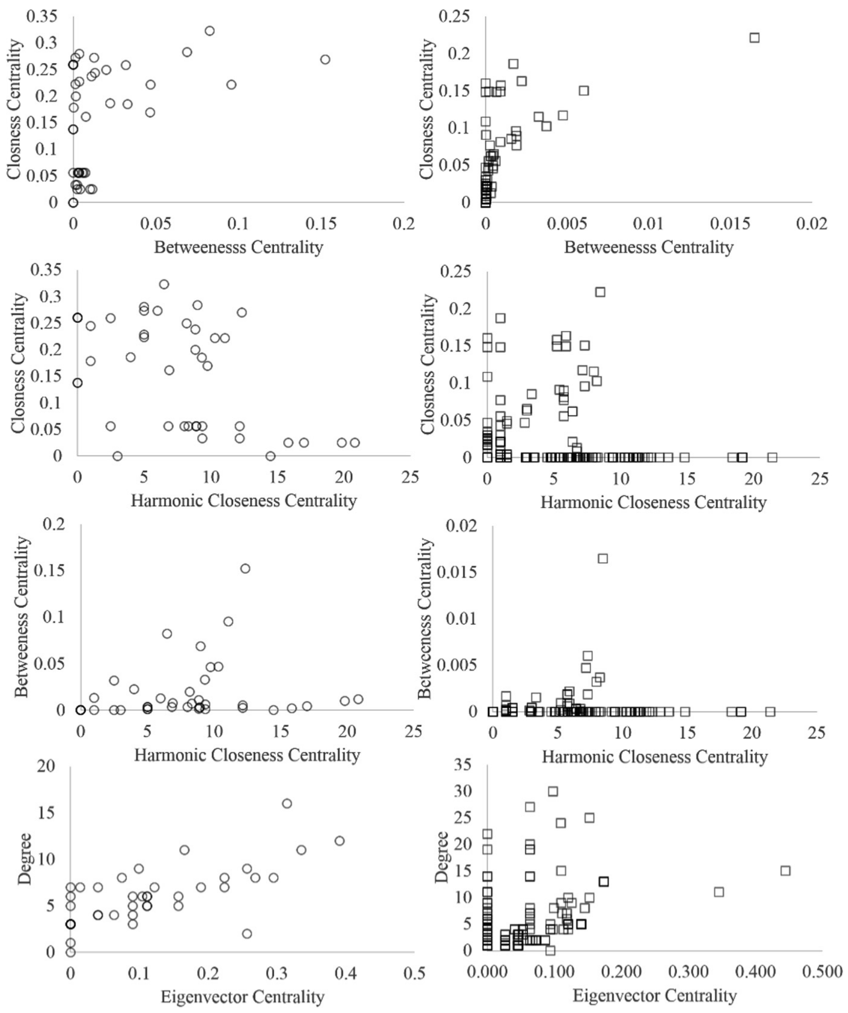


Fig. 1. Relationships between five importance metrics for the 41 entities represented in the SAR architecture network (left), and the 235 entities represented as vertices in the MComms architecture network (right).

either explicitly coded in the network as a node property, or implicitly reflected in some aspect of network structure, this importance criterion will be invisible to graph-theoretic analysis. The graph-theoretic signifiers of importance do not necessarily even have any meaning in the real-world system, an architect would need to determine what terms like the 'shortest path' correspond to in a heterogeneous architecture abstracted into a simple directed graph of vertices and edges.

2.2 Evaluating Structure

Edge density has previously been suggested by the authors as a proxy measure indicating the degree to which integration or dependency management will be a challenge for an architecture [1]. In reality, however, the metric is too simplistic to be useful for architecture evaluation [15]. The metric neglects context specific information that is likely to be more important to an architect such as the geographical separation between entities, the frequency of communication or the format of information, data or resource transfer [16–18]. Network perspectives do provide an opportunity to quantify the complexity of an architecture, by quantifying the number of components, the number of interfaces and various graph topology measures [19]. However, there is little agreement on complexity definitions or how complexity should be measured [20–22].

Another structural evaluation available to an architect who adopts a network perspective is exploring whether a core and periphery structure exists. Detecting such a structure within a SoS may be desirable to determine which entities are more important to overall functionality and hence where intervention effort should be directed. Existing literature highlights an approach using path lengths [23], and previous work by the authors used the detection of strongly connected components [1]. The MComms use case had only 6% of the architectural entities identified as belonging to the core, which could indicate that this architecture does not demonstrate a significant core and periphery structure. Perhaps more likely, however, is that strongly connected components are too naïve a measure of core and periphery structure. The MComms use case has a tree-like structure, with several entities that can only be reached in one direction. What such a structure means for the real-world system is currently difficult to assess due to the heterogeneity of entities modeled and depends on what has or has not been modelled in the directed graph. One can imagine the identification of a core and periphery of the architecture by more sophisticated and context specific measures, such as say the frequency and criticality of communication between entities, or their contribution to mission objectives [16, 17, 24, 25].

A similar problem occurs when using community detection to partition an architecture. Just because a set of architectural entities have strong connectivity between them does not guarantee that they form a “community” or that it is sensible to treat them as a distinct architectural “unit”. It is the failure of the network view to capture the relevant contextual information that causes this limitation.

Architecture structure can also be examined in terms of how reciprocal connections are, that is the tendency for entities within an architecture to have mutual directed connections [26]. In theory, such an approach may provide a numerical means to characterize SoS type, for example “collaborative”, which would have a high reciprocity, or “directed” which would not exhibit significant reciprocity [27]. The SAR architecture reciprocity is on average over six times that of the MComms architecture. Both architectures have more reciprocity than would be expected by chance in an architecture with the same number of vertices and edges, and the same in- and out-degree distribution. This was demonstrated by comparing the real-world architectures with 1000 null models, each containing the same number of vertices and edges but where the in- and out-degrees are shuffled at random. The SAR architecture reciprocity was greater than every random architecture generated, and the MComms architecture’s reciprocity was greater than 99.6% of random architectures. However, the characterization of a SoS may need to go further than comparison with an entirely structureless null model. In order to determine if an architecture's reciprocity values are high *for an architecture of its type*, it would need to be compared against a null model that respects the constraints under which architectures are arrived at. Since these constraints, whether fundamental or conventional, are to a large extent unknown, such a null model is hard to arrive at. Making progress will require a closer appreciation of the role of reciprocity in real-world architectures. Indeed, it might be argued that one positive associated with taking a network perspective is pressure to formulate the thinking that is required by such null models – thinking that has been painstakingly undertaken in other fields that have benefitted from networks science approaches [28, 29].

2.3 Network Perturbation

A system architect may be concerned with several dynamic properties such as flexibility and resiliency [30–33]. In networks arenas, it is common to explore the resiliency of networks, often measuring the impact of the removal of vertices on some proxy measure of network effectiveness such as diameter (the number of nodes involved in the longest of the shortest paths between all pairs of nodes) or the size of the largest component (the numbers of nodes in the largest fragment of the network) [34–36], but others have noted that more detailed simulation is needed rather than simply assessing the result of single vertex removal [37, 38].

When subjected to perturbations that removed vertices, both use case architectures were found to have a significantly greater vulnerability to vertex removal when the highest degree vertices were removed (compared to targeting vertices for removal at random). However, this vulnerability was measured in terms of changes in average centrality scores, such as average Closeness Centrality, or average Betweenness Centrality. These averaged centrality scores may not be suitable proxies for the effectiveness of the remaining architecture as they do not consider the topology of the surviving architecture, nor do they capture the differential importance of entities and their relationships. Similarly, network diameter and the size of the largest component are also naive measures of architecture effectiveness; while the size of the largest surviving network component may make sense as a proxy for effectiveness of a communications network, is this also true for a complex SoS? Perhaps a relatively small but critical subset of architecture entities and their connectivity are required for the overall operation of an SoS – or, in extremis, perhaps the loss of any entity in the architecture would compromise system operation? Conversely, many system entities may have some capability to resist perturbation through inherent resiliency and adaptability. This type of capability may be captured in the full representation of a system architecture, but could be difficult to represent in a simple graph.

2.4 Evaluating Vulnerability to Failure Cascades

Taking a network perspective also enables an architect to inquire about their architecture's vulnerability to cascades, simulating single or multiple nodes being degraded, to represent perturbations of the real-world that may have several uncorrelated issues, but could result in cascading failure. The fundamental limitation with trying to characterise this vulnerability is that achieving sufficiently secure knowledge about failure dynamics is likely to be extremely hard at the early stages of a system lifecycle. Further, an architect has a significant challenge in conceptualizing how failures and failure cascades could affect a SoS architecture that encompasses a diverse, potentially autonomous and independent collection of systems [39–41]. Whilst the explicit cascade dynamics may not be known at the early stages of a design lifecycle, it may be useful to consider the *possibility* of local failures creating global, SoS wide failures. It then may be possible to lean on network science approaches to explicitly design against cascading failures [42] by introducing protection to the nodes with the highest degree or implementing control mechanisms to distribute the functionality a failed node provided. However, their effectiveness hinges on understanding the cascade dynamics for the system, for example a protection strategy that assumes cascades follow a percolation model may not provide the same benefits for epidemic models of cascades.

The next section highlights the challenges to making progress in using a network perspective to evaluate complex SoS architectures and details the other open research questions.

3 Challenges to Progress

There is a challenge in modelling the heterogeneity of entities and relationships present in a complex SoS; in the SAR architecture entities as different as a Command and Control capability and a communications service coexist, and relationships as different as hierarchical dependencies and service to capability mapping coexist. One solution is to assign attributes to each entity and each relationship in the graph-theoretic model to capture the autonomy, diversity and richness of behavior present. These attributes could be numerical or textual and used to model issues relating to resiliency such as robustness, adaptability and recoverability. Other authors have used this approach and tried to model integration challenges by using an attribute for System Readiness Levels (SRLs) (similar to Technology Readiness Levels (TRLs)) [43]. However, there is a danger in attempting to improve the fidelity of graph-theoretic models by adding an increasing number of attributes to entities and relationships; how can confidence in these values be determined given the early stage of a lifecycle architecting focuses on? More fundamentally, however, a network perspective is exactly that, a lens through which to view a system, not a detailed model that provides direct answers about to how robust, complex or desirable an architecture is.

The more sophisticated the network representation, in terms of multiple types of vertex, multiple types of edges, edge weights, nested, layered, or interdependent subnetworks, etc., the more complicated the interpretation of network properties. Simple network concepts such as clustering or path length have relatively straightforward interpretations in, e.g., networks of Facebook pages, where every vertex is a page and every edge is a friendship relationship. The same cannot be claimed for a network in which some nodes represent physical resources, some are software services, and some are governance structures, where some nodes have geographic locations, some are multi-site and some are not localized at all, where some edges are dependencies, some are communication channels, and some represent flows of resource. Interpretation becomes even more challenging for more complicated properties such as betweenness or assortativity, also dynamics on such networks, e.g., cascades, flows, fragmentation, etc.

There is another feature of SoS and their complexity that creates a challenge for an architect; their scale and scope. With many views already available to an architect following a methodology like NAF or MODAF; how can an architect choose which areas of the SoS to explore using a network perspective? It is feasible that some architecture views are more amenable to graph-theoretic analysis, for example logical node interactions or resource connectivity. However, a full investigation has not been completed. Similarly, one can imagine certain architecting activities, e.g., architecture partitioning, as being particularly amenable to a network perspective but again a thorough evaluation has not been completed [27, 44]. A compounding challenge is how an architect can validate their graph-theoretic models given the scarcity of data likely available at the start of a system lifecycle. We thus stop short of recommending a formal, dedicated 'Complexity View' or 'Network View' when architecting complex SoS but further work is needed to explore the utility of such a view.

The search for a widely applicable importance metric or complexity metric is perhaps unwise given the role that context plays in complex SoS architectures. Instead, perhaps utility comes from the opportunity to not only reflect on what network perspectives do and do not provide, but also on why they were offered as a potential tool in the first place. The scale, diversity and connectivity of a complex SoS suggested network science tools could signpost important entities in an architecture. The answer from these tools is that it depends on what modelling assumptions importance is defined by. This suggests the community needs to tackle the challenge of identifying what makes an entity important in an architecture; is it their connectivity, the effect removal has, the role they play in a mission, their utility in brokering services from diverse and geographically separated entities? Although a network perspective may not provide the answer, it promotes consideration of what can be understood as important to a complex SoS. A similar argument exists for the evaluation of desirable non-functional requirements such as resiliency and flexibility; if they cannot be assessed numerically with confidence using a network perspective, how can we evaluate these properties? Network terms such as the size of the largest component, or average shortest path, currently have no easy translation to a complex SoS architecture but that does not mean the search for a corresponding term or application of such terms is a futile endeavor.

4 Conclusion

Given the impact of network science on fields as diverse as neuroscience [45] and archaeology [46], it is highly likes that there are insights to be gained from taking a network perspective on SoS architectures; which entities in an architecture are most important to one another, is one architecture likely to be more robust, efficient, or manageable than another, to what degree and in what ways might an architecture be vulnerable to failure? In seeking these insights, however, it becomes clear that the tools from network science cannot straightforwardly be applied without developing a more sophisticated understanding of how they map onto the diversity, richness and context sensitivity characteristic of complex SoS architectures. The social sciences spent several decades developing a set of interpretations and conceptualizations that allowed the effective mobilization of networks concepts. Developing an equivalent set of conceptual tools for the analysis of complex SoS architectures remains an open research challenge.

References

1. Potts, M., Sartor, P., Johnson, A., Bullock, S.: Hidden structures: using graph theory to explore complex system of systems architectures. In: Paper presented at the International Conference on Complex Systems Design & Management. CSD&M, Paris, France, December 2017
2. North Atlantic Treaty Organization: NATO architecture framework v4.0 documentation (draft) (2017). http://nafdocs.org/
3. Diestel, R.: Graph Theory, Electronic. In: Graduate Texts in Mathematics, vol. 173. Springer, Berlin (2005)
4. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **10**, P10008 (2008)
5. Biggs, B.: Ministry of defence architectural framework (MODAF) (2005)
6. Newman, M.: Networks: an Introduction. Oxford University Press, Oxford (2010)
7. Okami, S., Kohtake, N.: Transitional complexity of health information system of systems: managing by the engineering systems multiple-domain modeling approach. IEEE Syst. J., 1–12 (2017)
8. Bartolomei, J.E., Hastings, D.E., de Neufville, R., Rhodes, D.H.: Engineering systems multiple-domain matrix: an organizing framework for modeling large-scale complex systems. Syst. Eng. **15**(1), 41–61 (2012)
9. Santana, A., Kreimeyer, M., Clo, P., Fischbach, K., de Moura, H.: An empirical investigation of enterprise architecture analysis based on network measures and expert knowledge: a case from the automotive industry. In: Modern Project Management, pp. 46–56 (2016)
10. Iyer, B., Dreyfus, D., Gyllstrom, P.: A network-based view of enterprise architecture. In: Handbook of Enterprise Systems Architecture in Practice, p. 500. PFPC Worldwide Inc., USA (2007)
11. Freeman, L.C.: Centrality in social networks conceptual clarification. Soc. Netw. **1**(3), 215–239 (1978)
12. Boldi, P., Vigna, S.: Axioms for centrality. Internet Math. **10**(3–4), 222–262 (2014)
13. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **25**(2), 163–177 (2001)
14. Newman, M.E.: The mathematics of networks. In: The New Palgrave Encyclopedia of Economics, 2nd edn., pp 1–12 (2008)
15. IEEE/ISO/IEC Draft Standard for Systems and Software Engineering - Architecture Evaluation, pp. 1–76 (2017). ISO/IEC/IEEE DIS P42030/D1, December 2017
16. Kossiakoff, A., Sweet, W.N., Seymour, S.J., Biemer, S.M.: Systems Engineering Principles and Practice, vol. 83. Wiley, London (2011)
17. Buede, D.M., Miller, W.D.: The Engineering Design of Systems: Models and Methods. Wiley, London (2016)

18. Bullock, S., Barnett, L., Di Paolo, E.A.: Spatial embedding and the structure of complex networks. Complexity **16**(2), 20–28 (2010)
19. Sinha, K., de Weck, O.L.: Structural complexity metric for engineered complex systems and its application. In: Gain Competitive Advantage by Managing Complexity: Proceedings of the 14th International DSM Conference Kyoto, Japan, pp. 181–194 (2012)
20. Lloyd, S.: Measures of complexity: a nonexhaustive list. IEEE Control Syst. Mag. **21**(4), 7–8 (2001)
21. Sheard, S.A.: 5.2. 1 systems engineering complexity in context. In: INCOSE International Symposium, vol. 1, pp. 1145–1158. Wiley Online Library (2013)
22. Fischi, J., Nilchiani, R., Wade, J.: Dynamic complexity measures for use in complexity-based system design. IEEE Syst. J. **11**(4), 2018–2027 (2015)
23. MacCormack, A.: The architecture of complex systems: do "core-periphery" structures dominate? In: Academy of Management Proceedings, vol 1, pp. 1–6. Academy of Management (2010)
24. Rechtin, E.: Systems architecting: Creating and building complex systems, vol. 1. Prentice Hall, Englewood Cliffs (1991)
25. Sillitto, H.: Architecting Systems: Concepts, Principles and Practice. College Publications, London (2014)
26. Newman, M.E.: Mixing patterns in networks. Phys. Rev. E **67**(2), 026126 (2003)
27. ISO/IEC/IEEE International standard - systems and software engineering – system life cycle processes, pp. 1–118 (2015). ISO/IEC/IEEE 15288 First edition 2015-05-15. https://doi.org/10.1109/ieeestd.2015.7106435
28. Freeman, L.: The Development of Social Network Analysis. A Study in the Sociology of Science 1. Empirical Press, Vancouver (2004)
29. Gilbert, N., Bullock, S.: Complexity at the social science interface. Complexity **19**(6), 1–4 (2014)
30. Crawley, E., De Weck, O., Magee, C., Moses, J., Seeringk, W., Schindall, J., Wallace, D., Whitney, D.: The influence of architecture in engineering systems (monograph) (2004)
31. De Weck, O.L., Roos, D., Magee, C.L.: Engineering Systems: Meeting Human Needs in a Complex Technological World. Mit Press, Cambridge (2011)
32. De Weck, O.L., Ross, A.M., Rhodes, D.H.: Investigating relationships and semantic sets amongst system lifecycle properties (Ilities) (2012)
33. De Neufville, R., Scholtes, S.: Flexibility in Engineering Design. MIT Press, Cambridge (2011)
34. Newman, M.E.: Complex systems: a survey (2011). arXiv preprint arXiv:11121440
35. Newman, M.E.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
36. Albert, R., Jeong, H., Barabási, A.-L.: Error and attack tolerance of complex networks (2000). arXiv preprint cond-mat/0008064
37. Khoury, M., Bullock, S.: Multi-level resilience: reconciling robustness, recovery and adaptability from a network science perspective. Int. J. Adapt. Resil. Auton. Syst. (IJARAS) **5**(4), 34–45 (2014)
38. Khoury, M., Bullock, S., Fu, G., Dawson, R.: Improving measures of topological robustness in networks of networks and suggestion of a novel way to counter both failure propagation and isolation. Infrastruct. Complex. **2**(1), 1 (2015)
39. Boardman, J., Sauser, B.: System of systems-the meaning of of. In: Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering Los Angeles, CA, USA, pp. 118–126, April 2006
40. Maier, M.W.: Architecting principles for systems-of-systems. In: INCOSE International Symposium, vol 1. Wiley Online Library, pp. 565–573 (1996)

41. ISO/IEC/IEEE Draft international standard - systems and software engineering - systems of systems considerations in engineering of systems, pp. 1–43 (2017). ISO/IEC/IEEE P21839, April 2017
42. Fu, G., Dawson, R., Khoury, M., Bullock, S.: Interdependent networks: vulnerability analysis and strategies to limit cascading failure. Eur. Phys. J. B **87**(7), 148 (2014)
43. Marvin, J.W., Garrett Jr., R.K.: Quantitative SoS architecture modeling. Procedia Comput. Sci. **36**, 41–48 (2014)
44. ISO/IEC/IEEE DIS 42020 Enterprise, systems and software - architecture processes (2017)
45. Barnett, L., Buckley, C.L., Bullock, S.: Neural complexity: a graph theoretic interpretation. Phys. Rev. E **83**(4), 041906 (2011)
46. Brughmans, T.: Connecting the dots: towards archaeological network analysis. Oxf. J. Archaeol. **29**(3), 277–303 (2010)

Generation and Visualization of Release Notes for Systems Engineering Software

Malik Khalfallah(✉)

AIRBUS DS, 31 rues des Cosmonautes, Toulouse, France
malik.khalfallah@airbus.com

Abstract. Designing complex systems such as satellites requires delivering design data with the associated list of updates regularly in a form of release notes. Creating release notes for system data delivery is challenging for system architects because it requires analyzing all resolved issues during a particular period of time and then summarizing them in a readable format for other architects. In this paper we present an approach for creating release notes. We performed an empirical study involving large amount of satellite projects data to categorize the content of release notes delivered by architects. We have identified the main patterns. We developed an algorithm that discovers these patterns given the history of project data. We applied our approach to system data managed in configuration.

1 Introduction

1.1 Paper Scope

Developing systems engineering software applications requires providing features such as systems modeling, systems simulation and systems data definition. Nevertheless, system engineering process is a collaborative effort that involves many engineers creating different kinds of data and thus all these data should be managed in configuration [1]. Many software applications for engineers rely on version control systems such as SVN or Git to manage in configuration their data [2]. This enforces the traceability, the coherency and the consistency of the data. Nevertheless, this is not sufficient to obtain a rigorous data definition process. Indeed, the configuration management responds to the "what question" of system data definition. It keeps track of all defined data. However, we need also to respond to the "why question" of systems data definition. Any system data update should be an answer or a part of an answer to an *issue*. If this is not done, then during the data inspection or utilization, we cannot know clearly why a system data update has been made.

To answer this problem, it is possible to interconnect the version control system into an issue management system. Then, every system data update will be associated to an issue to provide clearly the reason of the update.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 133–144, 2019.
https://doi.org/10.1007/978-3-030-04209-7_11

1.2 Addressed Problems

System engineering developments rely more and more on agile processes [3]. This means the delivery of intermediate versions of the system data at different milestones. Accordingly, it is necessary to provide release notes at each milestone. Release notes are valuable assets in system development projects. Not only do they serve as a communication medium for issue resolution – an activity that accounts for as much as 40% of software and system development effort [4] - but they are also often consulted even after the issues have been resolved [5].

In this paper we aim at conducting a study in collaboration with different system architects in order to:

1. Analyze different satellite projects data as well as the delivered release notes during the projects development
2. Identify system data definition patterns that are mostly repeated and how architects have mapped them into release notes prior to their delivery.
3. Develop an adaptable model-based approach that discovers these patterns and generates the release notes automatically.

1.3 Motivating Example

Although our study aims to be generic and applicable to different system definition software, we provide an example using an industrial software application to present the study's results. Airbus DS has developed a software application of type SRDB (System Reference Database) [8] to define satellite systems structures and all their related data (TM/TC, Electrical, Functional) called RangeDB [6, 7, 9]. Working with an SRDB is a collaborative effort involving system architects that locally define their data and then share them with other engineers using a version control system.

In RangeDB, the concept of a *dataset* refers to a folder that contains particular system data. A dataset has a *version* and can *depend* on one or more versioned datasets. In configuration, a dataset can belong to a *trunk* but also to a *branch* or to a *tag*.

Generating release notes was a manual task. Starting from the dataset representing the system concerned by the release note, the architect browses the modifications that have been made since the last milestone. He notes down every resolved issue from redmine and the associated modifications.

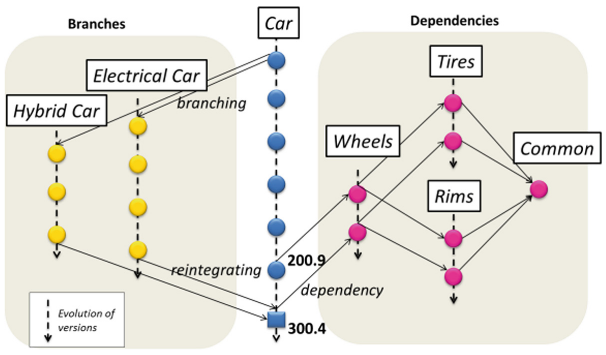


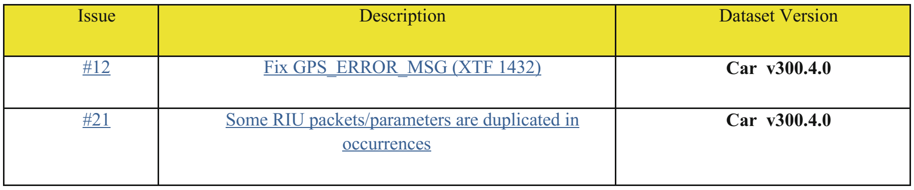
Fig. 1. Car data example in RangeDB

Let’s consider multiple engineering teams working on the design of a car. We chose a car example because it is simpler to illustrate our approach using a car rather than using a satellite system. Figure 1 shows how the dataset of the *Car* and its branches and dependencies are evolving. Basically the dataset *Car* has been branched at the first version and starting from version 200.9, it has started to have dependencies with other subsystems’ datasets.

At the end of the milestone, the architect wants to generate the release note of all resolved issues between the last version 300.4 (depicted by a square) and 200.9 of the trunk. The details of this history contain two important pieces of information:

- Information about the modifications of the dataset *Car* between the versions 300.4 and 200.9
- Information about the modifications made on the dependencies and branches of this dataset during the interval [200.9, 300.4].

The release note is presented as an excel document that shows the issue number, its description and the version of the dataset impacted by the modification. An example of such a table is given in the following figure.



Issue	Description	Dataset Version
#12	Fix GPS_ERROR_MSG (XTF 1432)	**Car v300.4.0**
#21	Some RIU packets/parameters are duplicated in occurrences	**Car v300.4.0**

Fig. 2. Manually developed release note in excel file

2 Framework to Generate Release Notes

2.1 Data Model to Capture Release Notes Building Blocs

The release notes built by architects have a uniform structure that can be captured in a data model and that can be populated by an automated process. The *release note* captures intervals of resolved issues between two versions of datasets. This is materialized by a class *release object*. Each release note contains one root release object that captures the starting point of release note computation. In our running example, this root release object concerns the dataset *Car* with its two versions 300.4.0 and 200.9.0. Then the release object itself contains *child release objects* that concern the intervals associated to dependencies and reintegrated branches of the current dataset (*Car*). In our example child release object concern the *Wheel* datasets. The challenge to automatically generate release notes concerns the determination of the content of the release note when datasets properties have changed between one version and another in terms of dependencies and branches.

2.2 Determination of Patterns

2.2.1 Empirical Study

The goal of the study is to analyze the datasets and their associations on three different satellite projects. One of these projects has already been finished which means that it is quite rich in terms of datasets definition and manual release notes generation. The other projects are nearly finished as well. We aim at answering the following research question:

RQ—Datasets Association Patterns: *What are the patterns of relationships between datasets that architects have faced when they generated release notes*?

To address this research question we have conducted an empirical study. The context of this study consists of the projects' datasets and their associations. We had access to the three projects and their datasets. In the SRDB, we had also access to the history of all commits, their comments and the issues IDs.

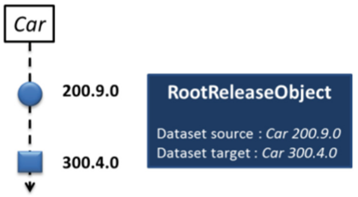
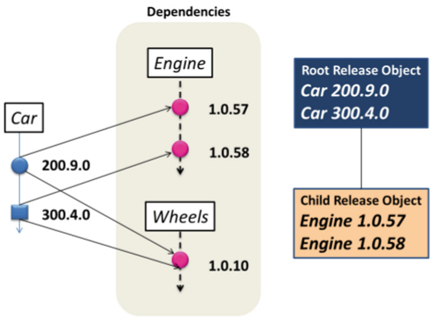
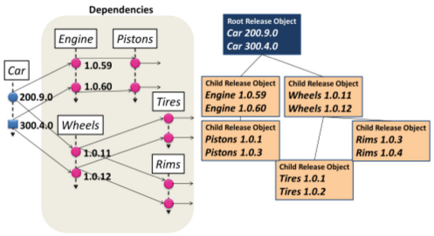
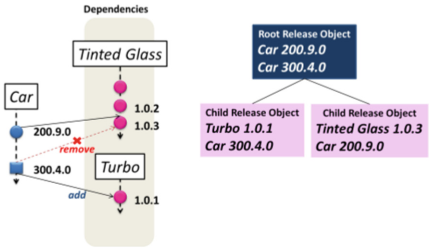
In order to cover a maximum number of situations that architects might face when generating a release note, we have simulated the need to generate release notes for the couples $(n+1, n)$ of the dataset versions representing the satellite of each project. For example for the first satellite project we have simulated the generation of a release note between the couples of versions (3.14.2, 3.14.1), until (1.0.1, 1.0.0).

In addition we have simulated the need to generate the release notes for the couples $(n+1, n)$ of the dataset versions of all satellite's systems and subsystem. Hence, knowing that the dataset for each system has evolved though numerous versions, we deem that we have covered enough cases to say that the patterns found are representative for the incoming projects data. We classify the faced cases into 4 categories: *(i) No Dependencies* exist for the current dataset version and its preceding dataset version, *(ii) Similar Dependencies* exist for the current dataset version and its preceding dataset version, *(iii) Different Dependencies* exist between the current dataset version and its preceding dataset version and *(iv)* there exist *Reintegrated Branches* in the current dataset version that were not reintegrated in its preceding version.

Figure 2 shows the distribution of these categories compiled for the three satellite projects and for all systems and subsystems of these satellites:

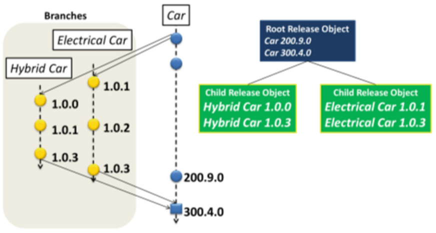
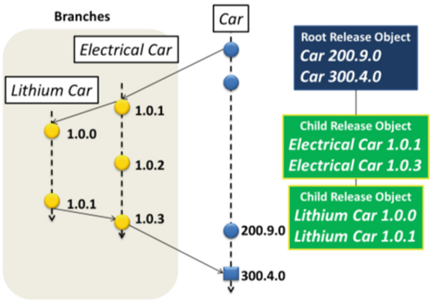
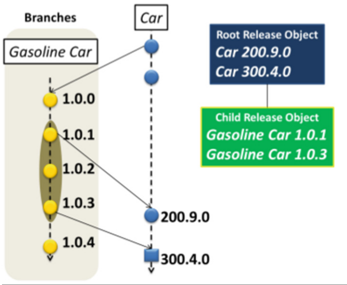
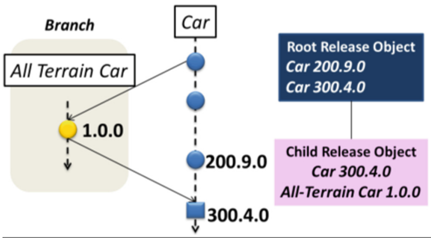
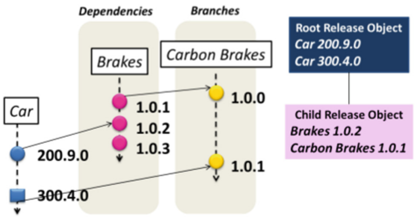
From these categories of datasets association, we classified them into 11 patterns that can be discovered automatically. We present each pattern and give the graphical view illustrating an example of that pattern in order to simplify its understanding. In addition, for each pattern, we depict the release objects and their relationships that capture the history of resolved issues for that pattern.

2.2.2 Study Results: Patterns Identification

Graphical representation and the created hierarchy	User defined object
 Car 200.9.0 300.4.0 RootReleaseObject Dataset source : Car 200.9.0 Dataset target : Car 300.4.0	**No Dependency, No Branch**: In this pattern we have one dataset in the trunk and we need to find the resolved issues for the last release. The dataset versioned 300.4 is represented as a square in order to say that it is the last release
 Dependencies Engine 1.0.57 1.0.58 Car 200.9.0 300.4.0 Wheels 1.0.10 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Engine 1.0.57 Engine 1.0.58	**Multiple Comparable Dependencies**: This pattern concerns the case where the source and target datasets have dependencies to different versions of the same dataset. Since these dependencies belong to the same axis, this means that they are comparable. In this case the result of the history analysis should list: (1) Issues resolved between 300.4 and 200.9 of the dataset *Car* and (2) Issues resolved between 1.0.58 and 1.0.57 of the dataset *Engine*
 Dependencies Engine Pistons Car 1.0.59 1.0.60 200.9.0 Tires 300.4.0 Wheels 1.0.11 1.0.12 Rims Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Engine 1.0.59 Engine 1.0.60 Child Release Object Wheels 1.0.11 Wheels 1.0.12 Child Release Object Pistons 1.0.1 Pistons 1.0.3 Child Release Object Rims 1.0.3 Rims 1.0.4 Child Release Object Tires 1.0.1 Tires 1.0.2	**Nested Comparable Dependencies**: This pattern is similar to the pattern number 2, but with multiple number of dependencies. We should notice that the number of dependencies is not infinite. At a certain level we reach the *commons* dataset that has no dependency. This allows to the search process to end
 Dependencies Tinted Glass Car 1.0.2 1.0.3 200.9.0 remove 300.4.0 Turbo add 1.0.1 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Turbo 1.0.1 Car 300.4.0 Child Release Object Tinted Glass 1.0.3 Car 200.9.0	**Tailored Dependencies**: This pattern concerns the removal and/or addition of dependencies to and from dataset. In other terms, the version *n* of the dataset has references to datasets that the version *n* + 1 has lost and dually, the version *n* + 1 has acquired new dependencies that the version n did not have For this pattern, it is not always possible to create a coherent history of resolved issues. Indeed, since the dataset *Car* in the version 300.4.0 has no longer a dependency to the dataset *Tinted Glasses*

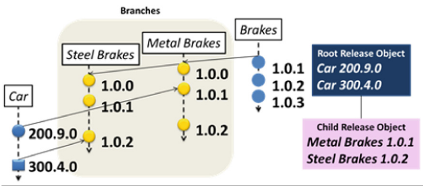
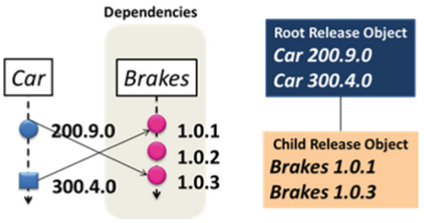
(*continued*)

(*continued*)

Graphical representation and the created hierarchy	User defined object
 Branches Car Electrical Car Hybrid Car 1.0.0 1.0.1 1.0.3 1.0.1 1.0.2 1.0.3 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Hybrid Car 1.0.0 Hybrid Car 1.0.3 Child Release Object Electrical Car 1.0.1 Electrical Car 1.0.3	**<u>Multiple Reintegrated Branches</u>**: This pattern concerns the reintegration of different branches created before or from the target dataset into the source dataset The result returned for this pattern concerns the issues resolved in all reintegrated branches plus the issues resolved between versions 300.4 and 200.9 of the dataset *Car*
 Branches Car Electrical Car Lithium Car 1.0.0 1.0.1 1.0.1 1.0.2 1.0.3 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Electrical Car 1.0.1 Electrical Car 1.0.3 Child Release Object Lithium Car 1.0.0 Lithium Car 1.0.1	**<u>Nested Branch Reintegration Branches</u>**: Following the same idea of dependencies having other dependencies (c.f. pattern 3), we can consider the case where branches have themselves other branches. In this case, the final result is constituted by the issues resolved in all reintegrated branches (recursively) plus the issues resolved between 300.4 and 200.9 of the dataset *Car*
 Branches Car Gasoline Car 1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Gasoline Car 1.0.1 Gasoline Car 1.0.3	**<u>Several Reintegration of the Same Branch</u>**: We can face this case where a single branch is reintegrated several times into the original trunk. For this pattern it is necessary to return only the resolved issues for intervals whose history has not been returned in previous results
 Branch Car All Terrain Car 1.0.0 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Car 300.4.0 All-Terrain Car 1.0.0	**<u>Reintegrated Branch without Previous Release</u>**: In this case it is not possible to create a history for the branch since it has no release. Accordingly, the issues retuned in the final result are those resolved for the dataset *Car* between versions 200.9 and 300.4
 Dependencies Branches Brakes Carbon Brakes Car 1.0.1 1.0.2 1.0.3 1.0.0 1.0.1 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Brakes 1.0.2 Carbon Brakes 1.0.1	**<u>Multi-Origin Dependency</u>**: This particular case is not possible to have in the SRDB since a dataset can have one and only one dependency to another dataset. Nevertheless if it was allowed, the result would have been to make the union of issues resolved in the branch and issues resolved in the dependency

(*continued*)

(*continued*)

Graphical representation and the created hierarchy	User defined object
 Branches Brakes Metal Brakes Steel Brakes Car 1.0.0 1.0.1 1.0.2 1.0.3 200.9.0 300.4.0 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Metal Brakes 1.0.1 Steel Brakes 1.0.2	**Multi-Origin Dependency [Different Depth]**: This pattern is a particular case of the previous one
 Dependencies Car Brakes 200.9.0 300.4.0 1.0.1 1.0.2 1.0.3 Root Release Object Car 200.9.0 Car 300.4.0 Child Release Object Brakes 1.0.1 Brakes 1.0.3	**Unchronological Dependency**: This pattern could occur when the user prefers switch to an old dependency in comparison to the previous version of the dataset. The result that is retuned for this case is empty regarding the issues resolved for the dependency

2.2.3 Model-Based Patterns Discovery Algorithm

The pattern discovery algorithm is modeled using Colored Petri Nets (CPN). Using CPNs has two main advantages: (1) It allows architects to define different behaviors of the discovery algorithm and thus making it generic and adaptable for different projects. (2) It allows us to prove certain properties of our algorithm.

In the following we elaborate on this algorithm:

- *dataset* is defined by the tuple: $\langle dataset_{name}, version_{number}, \{trunk_{name}/tag_{name}\}\rangle$.
- $hasNoDep(dataset_i)$ is a function that checks whether the dataset $dataset_i$ has no dependency. In RangeDB there are no cyclic dependencies and hence there is always at least one dataset in the dependencies chain that has no dependency.
- Two dataset tokens $\langle d_1, v_1, t_1\rangle$ and $\langle d_2, v_2, t_2\rangle$ are comparable iff: $d_1 \sim d_2 \wedge t_1 \sim t_2$. The symbol $\sim$ between $d_1$and $d_2$means that the two datasets belong to two different tags but these tags originate from the same trunk. Similarly, the symbol $\sim$ between t_1 and $t_2$means that these two tags are different but originate from the same trunk.
- A guard G in the CPN on two dataset tokens d_{source}, d_{target} is formalized by the following condition: $G = \left[hasNoDep(d_{source}.d) \wedge hasNoDep\left(d_{target}.d\right)\right] \vee \left[d_{source}.d \sim d_{target}.d\right] \wedge \left[d_{source}.t \sim d_{target}.t\right]$

To build the CPN that computes the release objects of two datasets, we first create a place that will contain the token representing the dataset source and a place that will contain the token representing the dataset target. In addition we create a guarded transition using G that computes the resolved issues between these two datasets and creates the corresponding release object. Figure 3 depicts the CPN.

The transition is fired if and only if G returns TRUE. In this case, the function $r_{comparable}\left(d_{source}, d_{target}\right)$ is called. This function creates the release object corresponding to the couple $\langle d_{source}, d_{target}\rangle$ and puts it into the place R. In addition, it places that release object into the right place in the hierarchy of release objects. Placing a

Projects	(i)	(ii)	(iii)	(iv)	Mean number of resolved issues in an interval of two versions of a dataset	Mean release time (day)
Project 1	3	525	31	15	3,34	21,61
Project 2	24	443	13	12	3,14	19,27
Project 3	0	193	8	7	2,99	33,38

Fig. 3. Categories of datasets associations for the three projects

newly created release object under its parent is performed if and only if among all release objects there is one that has a dependency between its source dataset and $p1.dataset$ and a dependency between its target dataset and $p2.dataset$.

When all token elements satisfying *r_comparable* condition have been consumed, at the end the places p1, p2 will contain only datasets that are not comparable ($\sim$) and thus should be treated depending on the patterns identified above. These details are implemented in the function *r_not_comparable*. Nevertheless, since each project could decide to create different release objects, in this case the implementation of the function *r_not_comparable* can be updated to manage the project specificities.

To obtain the complete history of resolved issues we need to include the issues resolved in the dependencies and the issues resolved in branches as well. Accordingly the CPN of Fig. 3 computes the dependencies and the reintegrated branches of the given couples $\langle dataset_{source}, dataset_{target} \rangle$ and then compute the resolved issues for them. Nevertheless, to compute the dependencies and the reintegrated branches for the initial datasets we need to use their associated tokens. Hence to make the tokens of the initial datasets available for both history computation and also to compute branches and dependencies, we need to create duplicates for these tokens. This duplication has the advantage to allow the parallel computation of reintegrated branches, dependencies and the history of resolved issues too.

Proposition1: Computation made by this CPN ends.

Proof: Since this CPN contains cycles, we need to prove that these cycles end at a certain time. In other terms, we need to prove that the generation of tokens representing branches and dependencies ends at a certain time. More formally, we consider two functions:

A function that computes the reintegrated branches of a given dataset:

$$reintegrated\_branches\colon Dataset \rightarrow 2^{Dataset}$$

A function that computes the dependencies of a given dataset:

$$dependency\colon Dataset \rightarrow 2^{Dataset}$$

In order to prove that the computation made by our CPN ends, we put the two following hypotheses:

1. There is always in the graph of dependencies the last dependency into which all datasets have a direct or indirect dependency called the common dataset that satisfies the following condition:

$$dependency(common_{dataset}) = \emptyset$$

2. The number of reintegrated branches for any dataset is limited. More formally:

$$\forall dataset_i : |reintegrated\_branch(dataset_i)| < \infty$$

Using the first hypothesis, we conclude that when the data source place contains $dataset_i = common_{dataset}$ then no dependency will be generated. Using the second hypothesis, we conclude that we reach a dataset that has no branch at that time and consequently the place containing branches will be empty. Moreover, the common dataset will be reached as a last dependency. Accordingly, the computation ends since the dataset source and dataset target will no longer be populated. ■

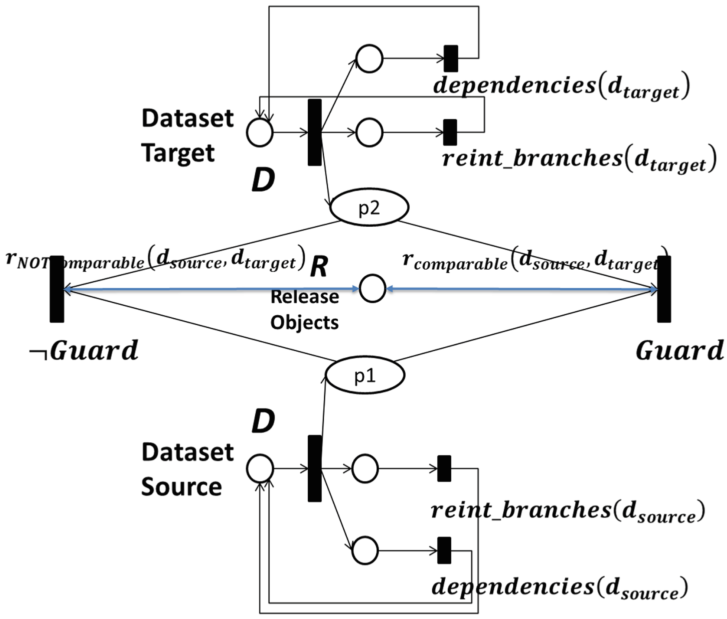


Fig. 4. CPN-based pattern discovery algorithm

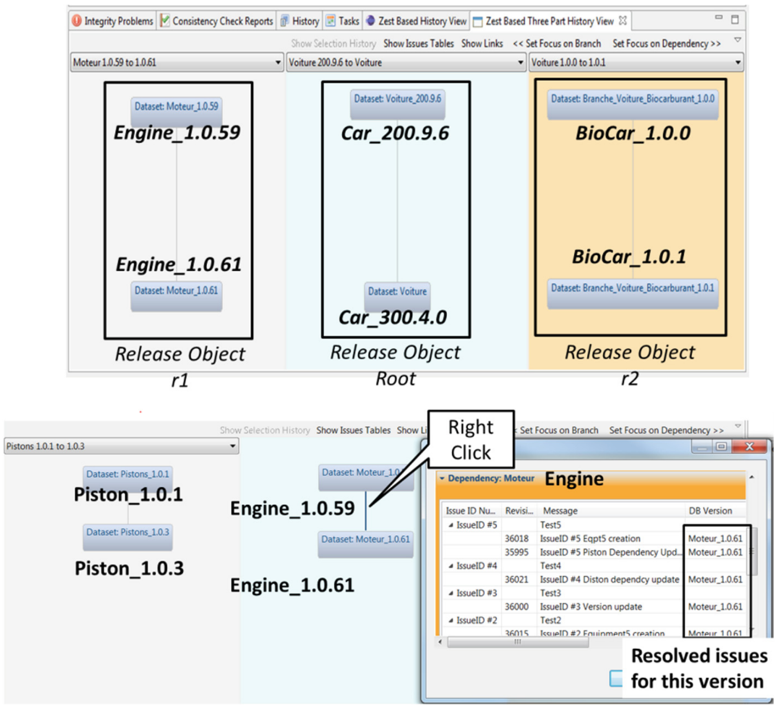
Proposition2: A child release object cannot have two parents.

Proof: Let r_1, r_2 two release objects. The release r_1 is a child of r_2 iff r_1.*datasetSource* and r_2.*datasetSource* have a dependency as well as r_1.*dataSetTarget* and r_2.*datasetTarget* have a dependency and are of the same nature. Or they are branches. Since a branch can have only one origin then there will be one parent.

The proof continue by enumeration on the different patterns. ∎

2.2.4 Implementation

We have developed a prototype for this approach. We have built a CPN model in EMF and its visualization in GMF. Architects can update the behavior of that CPN according to their needs. Additionally, we have developed the visual representation of the release note in Eclipse RCP that constitutes a plugin to RangeDB. Figure 4 depicts an excerpt of the release note visualization corresponding to the example above: (Fig. 5).



Issue ID Nu...	Revisi...	Message	DB Version
IssueID #5		Test5	
	36018	IssueID #5 Eqpt5 creation	Moteur_1.0.61
	35995	IssueID #5 Piston Dependency Upd...	Moteur_1.0.61
IssueID #4		Test4	
	36021	IssueID #4 Diston dependcy update	Moteur_1.0.61
IssueID #3		Test3	
	36000	IssueID #3 Version update	Moteur_1.0.61
IssueID #2		Test2	
	36015	IssueID #2 Equipment5 creation	Moteur_1.0.61

Fig. 5. Generated release note for the car example

3 Related Work

Generating and visualizing release note is an active research field. Klepper et al. [10] developed a semi-automated approach for the generation of release notes. The principle of their approach consists in identifying the changes in the commits between two releases of a project. They summarize the code changes and link it to information commit notes and issue tracking system. This approach would provide relevant information of interested people but authors do not elaborate how they automatically gather all relevant information. In our case we provided a workflow designed by a CPN that elaborates on how we perform the generation of the release note.

[11] developed a tool called ChangeScribe. The purpose of ChangeScribe is to assist users to generate the right message to associate to commits. Its principle consists in extracting and analyzing the differences between two versions of the source code and generates a commit message. The analysis of the source code relies on code summarization techniques, stereotype detection and impact analysis. The generated commit message provides an overview of the changes. Basically, it describes the why and what of a change using natural language. For simple source code files this approach could work as demonstrated by the authors. However, when committing many files this tool could fail. Moreover, the generated messages do not contain the issue number which is fundamental to show other developers which issue has been resolved by the commit.

[12] developed RCLinker that aims at predicting if a link exists between a commit message and an issue defined in a system such as redmine. It relies on ChangeScribe to automatically generate commit messages. Then it tries to extract features from the automatically generated commit messages and issues description. These features allow RClinker to link the generated commit messages with their corresponding issues.

RCLinker made the hypothesis that in many cases software developers could perform a commit without associating it to an issue [13–15]. In our case (system development) it is always mandatory that database architects associate an issue to their commits. This is due to the difference between systems and software. In software we could refactor the source code and commit it without informing the final user. At the end, the behavior of the software remains the same. However in system development the final users are system architects and every data update will probably impact their design. Therefore every data update should be backed by an issue.

4 Conclusion

In this paper, we have presented the foundations of a release note generation framework for systems engineering software. We first determined the need for such framework that results from a business need expressed in AIRBUS DS. Second we have performed an empirical study on a large sample of satellite projects data in order to categorize the entries of a release note. Third we have defined patterns corresponding to these categories and we have developed a CPN based discovery algorithm to generate the release note. We finally proved important properties of that algorithm.

In the future we aim at enriching that framework by developing a generic interface between System engineering software and issue management systems.

References

1. Madni, A., et al.: Model-based systems engineering: motivation, current status, and needed advances. In: Disciplinary Convergence in Systems Engineering Research (2018)
2. Lanubile, F., et al.: Collaboration tools for global software engineering. IEEE Softw. **27**(2), 52–55 (2010)
3. Schindel, B., et al.: Introduction to the agile systems engineering life cycle MBSE pattern. In: INCOSE International Symposium (2016)
4. Boehm, B.: Software defect reduction top 10 list. IEEE Comput. J. **34**(1), 135–137 (2001)
5. Lotufo, R., et al.: Modelling the 'Hurried' bug report reading process to summarize bug reports. J. Empir. Softw. Eng. **20**(2), 516–548 (2015)
6. Eisenmann, H.: MBSE has a good start; requires more work for sufficient support of systems engineering activities through models. INCOSE Insight **18**(2), 14–18 (2015)
7. Eisenmann, H., et al.: RangeDB the product to meet the challenges of nowadays system database. In: SESP-ESA (2015)
8. Eickhoff, J.: Onboard Computers, Onboard Software and Satellite Operations. Springer, Berlin (2012)
9. Cazenave, C., et al.: Benefiting of digitalization for spacecraft engineering. In: SESP-ESA (2017)
10. Klepper, S., et al.: Semi-automatic generation of audience-specific release notes. In: IEEE/ACM CSED 2016 (2016)
11. Cortès-Coy, L., et al.: ChangeScribe: a tool for automatically generating commit messages. In: IEEE SCAM 2014 (2014)
12. Le, T., et al.: RCLinker: automated linking of issue reports and commits leveraging rich contextual information. In: IEEE ICPC 2015 (2015)
13. Bachmann, A., et al.: The missing links: bugs and bug-fix commits. In: ACM SIGSOFT FSE (2010)
14. Bird, C., et al.: Fair and balanced?: bias in bug-fix datasets. In: ACM SIGSOFT FSE (2009)
15. Thanh, N., et al.: A case study of bias in bug-fix datasets. In: Working Conference on Reverse Engineering (2010)

Safety Architecture Overview Framework for the Prediction, Explanation and Control of Risks of ERTMS

Katja Schuitemaker[1(✉)], G. Maarten Bonnema[1], Marco Kuijsten[2], Heidi van Spaandonk[3], and Mohammad Rajabalinejad[1]

[1] Department of Design, Production and Management, University of Twente, Enschede, The Netherlands
{k.schuitemaker,g.m.bonnema, m.rajabalinejad}@utwente.nl
[2] Safety Department, NS, Utrecht, The Netherlands
marco.kuijsten@ns.nl
[3] Safety Department, ProRail, Utrecht, The Netherlands
heidi.vanspaandonk@prorail.nl

Abstract. The proposed framework includes modelling of interfaces between risk analysis, risk evaluation and scenario's representing flows of safety information of the European Railway Traffic Management System (ERTMS). In this study, we propose a functional framework combining safety data generation, data processing and structuring, definition of interactions and finally, the creation of customized representations in order to predict, explain, and control risks. Through literature review and ERTMS applicability, we develop a safety architecture overview framework. The comprehensive overview of the safety architecture can illustrate the main interactions between government, regulations, company management, technical and operational management, physical process and activities, and environment. Explicit representation delivers insight, stimulates striving for completeness, and leads to consistency of the safety analyses.

1 Introduction

The European Railway Traffic Management System (ERTMS) is subject to an increasing number of stakeholders [1], open specifications [2], and split-responsibilities [3]. Many and varied interactions among the individual components are approached proactively and qualitatively where little time and pressure towards cost-effectiveness can inadvertently lead to generating adaptive responses [4]. In previous study [5], the effects of the safety case regime, interoperability, deregulation and dynamic specifications on the ERTMS have been researched at the Dutch national level. This study concluded that achieving an interoperable and safer railway system by implementing ERTMS appears not to be straightforward for three key reasons:

- The safety case argument involves descriptions and observations including various explanations and interpretations from stakeholders.

© Springer Nature Switzerland AG 2019
E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 145–156, 2019.
https://doi.org/10.1007/978-3-030-04209-7_12

- For the Dutch situation, the absence of a central designer [6] and overarching safety decision-making processes between railway and train transportation lowers the degree to which the parties succeed in harmonizing various processes.
- An increased number of actors has caused a lack of insight into cross-border information.

These challenges require improvements in resilience, more awareness and increased sensitivity for interrelationships between hazards and risks, but even more: joint comprehension of the safety architecture and creation of cross-discipline understanding.

In this study, we create a safety architecture overview framework representing structured scenarios including hazards, consequences, RACs, risks and decisions in various layers. We model interfaces between scenarios, risk analysis and risk evaluation so that stakeholders are able to verify data origin, argumentation route, and application. Also, we argue that the proposed framework addresses the explained challenges through combining safety data generation, data processing and structuring, definition of interactions, and finally the creation of customized representations in order to predict, explain, and control risks.

Section 2 provides an overview of ERTMS and state of the art of safety models aiming at modelling elements of the safety architecture. The methodology is discussed in Sect. 3. Section 4 explains the creation of the safety architecture overview framework and how this complies with the challenges described above. These results are discussed in Sect. 5. Section 6 summarises the results, draws conclusions, and explains future work in order to test the proposed framework.

2 Background

ERTMS is a command, control, signalling, and communication system for railway management and safe regulation. It is composed of two technical components: (1) European Train Control System (ETCS): the Automatic Train Protection (ATP) system that makes sure trains do not exceed safe speeds or run too close together and (2) Global System for Mobile Communications – Railways (GSM-R): helps to provide communication for voice and data services.

The Dutch House of Representatives took an official preferential decision in 2014 that included a phased implementation of ERTMS. The Commission Implementing Regulation (EU) 402/2013 concerns the regulation on a Common Safety Method (CSM) for risk evaluation and assessment (CSM RA). This regulation is mandatory for railway duty holders in Europe, including the Netherlands. The safety of ERTMS should comply with the European Norm (EN) 50126, 50128, 50129 and 50159. Typical safety assessment methods for safety case creation used in railway industry, but also in other industries such as offshore, nuclear plants, and air traffic control are the Preliminary Hazard Analysis (PHA) and Hazard and Operationally studies (HAZOP). For these analysis, it is important to first define causal scenarios: potential sequences of events of an initiating event that could lead to a potential dangerous scenario.

Stakeholders involved with the creation of the safety case are the Dutch Ministry of Infrastructure and the Environment (I&W), the Dutch infrastructure provider (ProRail), and train operating companies such as the Dutch railways (NS). Next, the safety case must show that the correct management for controlling safety is in place. For the Dutch ERTMS, this management system refers to the safety management systems (SMS) of both ProRail and NS.

In short, ERTMS is subject to the influence of the Dutch House of Representatives, the application of the CSM and EN5012x, the SMS of the infrastructure provider and train operating companies, multiple technical components, trains operating on tracks, and of course, the consideration that ERTMS is an important link in the ambition to ensure the passengers and shippers view the railway system as an attractive mode of transportation. This indicates active layers in the area of government, regulations, company management, technical and operational management, physical process and activities, and environment. These levels of decision making that are involved in risk management and control hazardous processes, are explained in [7].

2.1 State of the Art

Model-Based Systems Engineering (MBSE) allows systems engineers to create the system structure and behaviour using interrelated models. MBSE is mostly used for creating the system description, and safety is often considered as a dependent attribute. On the other hand, for the missing link between MBSE and safety, several models for safety analyses have been developed.

Multiple languages have been established for safety annotation, for example the Goal Structuring Notation (GSN) [8], AltaRica [9], EAST-ADL [10], and SAML [11]. GSN is a graphical notation, using hierarchical goal structures to document the safety case. In AltaRica, the expression is in the form of a collection of Node possessing hierarchical structures, focussing on computation of dysfunctional models. EAST-ADL is an architecture description language intended to support the development of automotive embedded software. One of its extensions concerns dependability and captures information related to safety. A SAML model combines discrete probability distributions and non-determinisms.

Some studies have used SysML for safety argumentation. Safety models that are system-oriented, are often using SysML, or a modified language based on SysML. MéDISIS, using SysML for PHA and FMEA, focus more on reliability [12]. Some models use SysML for safety modelling, but not on the safety analyses itself. Examples are SafeSlice focussing on requirements and inspections [13], a model focussing on requirements [14], SafetyMet focussing on compliance with standards [15], a model focusing on the certification process [16], and O&SHA focussing on requirements and on the integration between SE and safety [17], though O&SHA does create operational views and defines a safe functional architecture. Belmonte and Soubiran [18] use both DSML (which is based on SysML) and AltaRica for the creation of PHA and FMEA. MSA is based on a combination of RobotML, AltaRica and OpenPSA for the Fault Tree Analysis (FTA) [19]. HiP-HOPS uses EAST-ADL and Boolean expressions for FTA and FMEA [20]. Some of these models zoom in on scenarios or hazardous flows. However, none of these models focus on both detailed characterisation of the evidence underlying the safety case, and customisation of risk analysis and risk evaluation representations for enabling communications between safety stakeholders.

3 Method

This study and the design of the Safety Architecture Overview Framework is carried out in four successive steps, though execution of step three and four are iterative.

- Step 1. Translate need/requirements to a top-level use case diagram. The resulting use cases can be considered as top-level functionalities of the framework and related to system requirements.
- Step 2. Decompose to a set of functions. The functions explained in the use case diagram are decomposed to a set of functions. Per top-level functionality, we define input and output.
- Step 3. Finding solutions. For each functionality, literature review is combined with ERTMS application in order to find suitable solutions.
- Step 4. Evaluation on functionality and compatibility between solutions. This step is interrelated with step 2 and 3, because this evaluation can suggest a change of flow or solutions that contradict one another.

The first two steps are executed through following the Design Research Methodology (DRM) described by [21]. Step 3 and step 4 are executed through following the systematic search with the help of classification schemes described in Engineering Design by [22]. This approach is shown in Fig. 1.

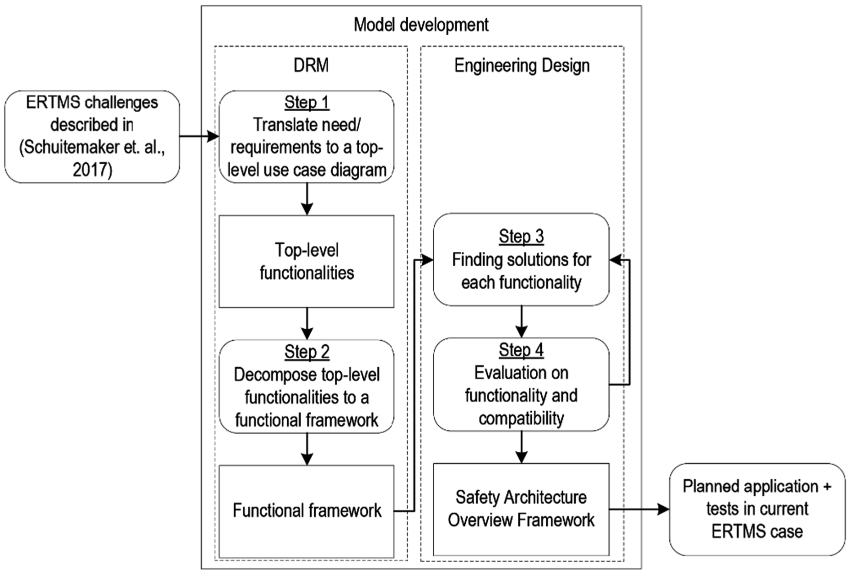

Model development
DRM
Engineering Design
ERTMS challenges described in (Schuitemaker et. al., 2017)
Step 1 Translate need/ requirements to a top-level use case diagram
Top-level functionalities
Step 2 Decompose top-level functionalities to a functional framework
Functional framework
Step 3 Finding solutions for each functionality
Step 4 Evaluation on functionality and compatibility
Safety Architecture Overview Framework
Planned application + tests in current ERTMS case

Fig. 1. Approach used for this study

4 Results

The aim of this study is to create a framework that addresses an interdisciplinary approach on both the social and technical level, and shows how parts interact and fit together. First we define interdependencies between entities and top functionalities of the framework. Next, we explain how actors are interacting with the framework. We explain each functionality in more detail and how these can be realised.

4.1 Top Functionalities

According to EN50126 (The Specification and Demonstration of Reliability, Availability, Maintainability and Safety), in order to supply relevant input, safety analysis must be performed by, at the minimum, a safety expert (key individuals or domain experts who understand the system under consideration) and a safety manager (has the responsibility over the risk assessment and ensures the traceability of safety related decision-making). For the creation of the total safety architecture, an integrator should create an integral coherence of the claims, arguments and evidence and the interdependencies between them. The task of this so-called "safety architect" is to define a complete, comprehensive and defensible argument.

For the interdependencies between entities, the use case diagram in Fig. 2 represents top functionalities of the framework, how an overview can be created, and how actors are interacting with the framework.

For the explanation of Fig. 2, the risk assessment approach requires analyses where hazards, risks, and mitigations are identified by following guidelines and logical reasoning of experts during requirements engineering and design. This data must be processed to create comprehensive information and avoid specialist terminology and linguistic ambiguity. Next, in order to consider the safety for the ERTMS as well as the safety for subsystems, it is important to clarify boundaries and relationships. Structuring the safety information evaluates and clarifies trade-offs between analyses. Technological risks must be understood within their context, where there are many active entities like actors, organisations, authorities, government, etc. Finally, stakeholders have various interests and various viewpoints, depending on the structure from which the process is viewed. To take into account these viewpoints, we need to customize the view to be analysed.

4.2 Safety Architecture Overview Framework

The proposed framework for creating the safety architecture overview combines generating and processing of safety data, structuring of information, defining interactions, and creating customized representations.

Data generation refers to the creation of data from risk assessment performed by safety experts. For identification of links between hazards and accidents, consequence analysis is often performed. For the generation of RACs of the Dutch railway system, this means that risks should be reduced to as low as reasonable practicable (ALARP). A risk matrix approach is used in conjunction with an ALARP based approach to risk reduction. Depending on the safety analysis phase, data can consist of hazards,

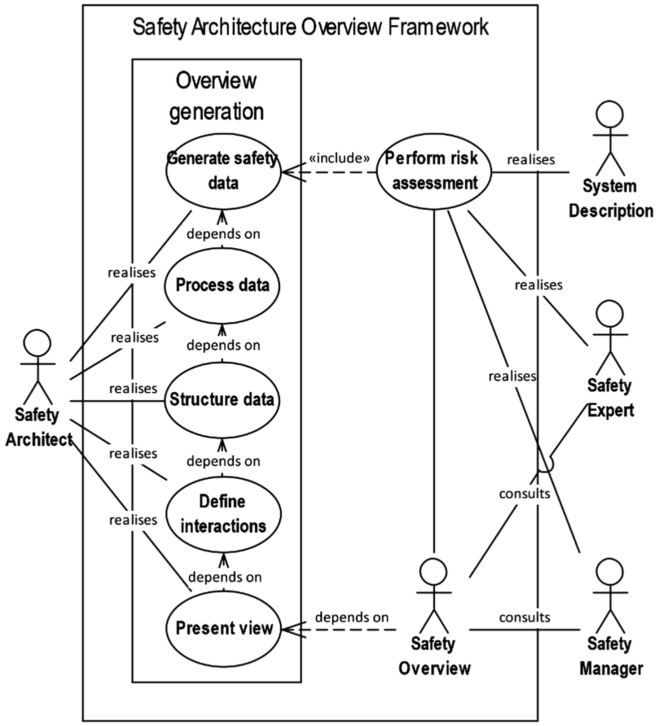


Fig. 2. Use case diagram representing top functionalities of the safety architecture overview framework.

consequences, risk matrices including tolerability limits, ALARP evaluations and decisions. This data is still in the form of raw data, obtained through oral sessions generated in real-time or documentation.

Data processing concerns the translation of raw extracted data from stakeholders, to valuable information. For the purpose of detailed characterisation of the evidence underlying the safety case, and customisation of safety analysis representations, GSN is not suitable. For the purpose of enabling communications between safety experts, safety architects, and safety managers, EAST-ADL, AltaRica and SAML are not well-known and do not focus on information presentation for these stakeholders. The Systems Modeling Language (SysML) is a more standardized and institutionalised language and has been shown to improve development communication during system design [23]. SysML also provides principles for partitioning and layering modules, which are crucial for structuring data and defining interactions. To be able to create valuable information from raw data, we need to select, abstract, and synthesize information:

- Select data. The process of collecting required and recommended data.

- Abstract data. We translate informal raw data to a formal language that creates common understanding. Next, we filter information to prevent information overload, and to deal with safety complexity.
- Synthesize information. The fitting together of parts or elements to produce new effects and to demonstrate that these effects create an all over order [22]. Grouping indicates that elements belong together based on some common characteristic. In this function, filtered information is labelled (stereotypes) according to their type.

This processing from raw data to interpretive safety information is shown in the activity diagram in Fig. 3.

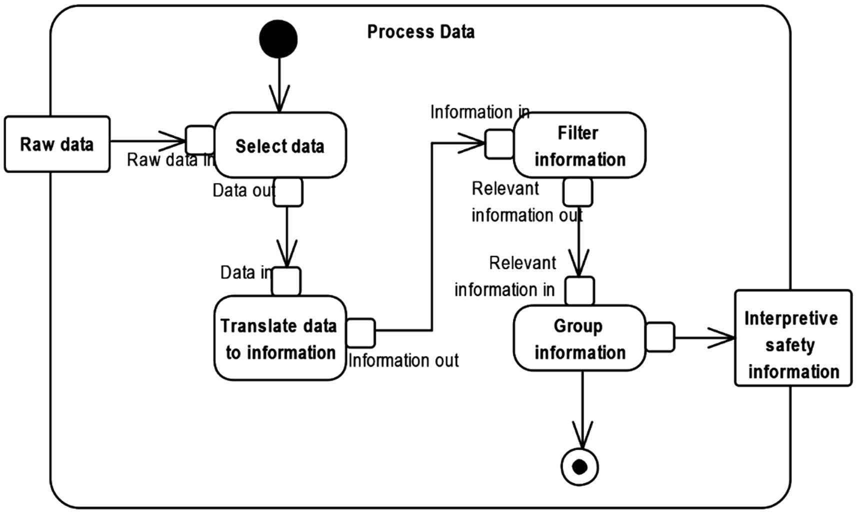


Fig. 3. Activity diagram representing the processing of safety data to interpretive safety information.

In PHA, high-level system hazards are identified inductively by asking “what if this component fails”, and hazard are also identified deductively by asking “how this could happen”. Scenario-guided hazard analysis is to be structured around the flows within a system. For example, each HAZOP contains complex chains of flow of information, and each flow can have hazardous effects. As for identification of hazards, their causes, and their effects, the focus within this framework is on the properties and behaviours of flows in the system.

A typical methodology for scenario identification is ETA; Cause-Consequence Analysis in particular may also be applied to identify scenarios. Causal analysis aims to identify the logical sequences of hazardous events that may lead to an undesirable effect (EN50126). Typical causal analysis techniques are FTA and FMECA. The use of inductive and deductive safety analyses results in downstream and upstream flows, see Fig. 4.

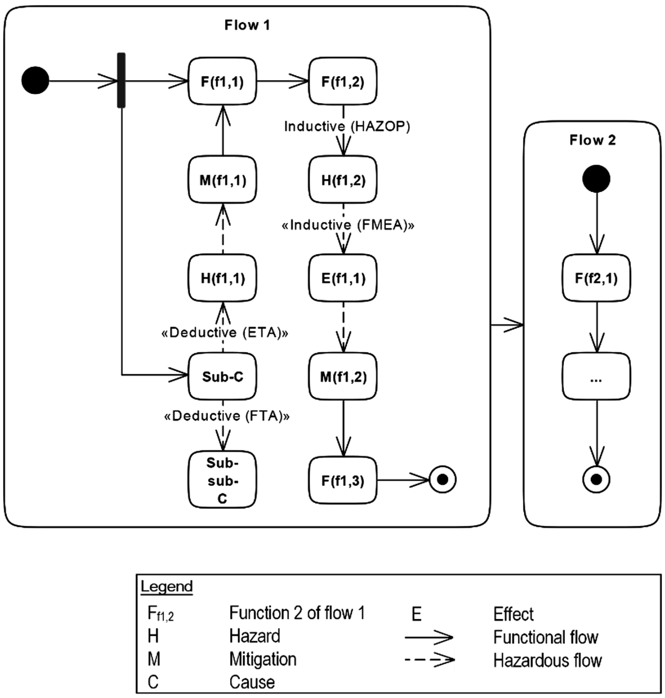


Fig. 4. Activity diagram of the flows representing safety analyses performed in the safety architecture overview framework.

As for ERTMS and moreover, the Dutch Railway industry, the safety case approach is applied to construct an argument that the system is adequately safe for a given application in a given environment. In accordance with the safety case, the structure upon which the Safety Architecture Overview Framework is built consists of:

- Claims. A conclusion or premise to be demonstrated. For example, that the system is safe to operate.
- Evidence. References that can be a result of a safety analysis. For example, FTA's or FMEA's.
- Arguments. Set of inferences between claims and evidence.

As for the example in Fig. 4, flow 1 includes some hazards (resulting for example from a HAZOP) for which mitigation M(f1, 1) and M(f1, 2) are applied in order to reduce the risk. For this reason, one can claim that execution of Flow 1 is acceptably safe.

The *definition of interactions* includes the identification of all factors that contribute to a failure. According to EN50126, the definition of the operational context is necessary to evaluate the risks specific to a hazard within its accident scenario. Identification of causal scenarios allows architects to discover interactions between various flows and layers such as human, technological, organisational and external, that might contribute to the failure at the system output. Each element of the scenario is allocated to one of the earlier explained 6 layers (government, regulations, company management, technical and operational management, physical process and activities, and environment). It is intended that each layer is considered when generating causal scenarios. We model these layers in SysML as partitions that share content. Each partition represents one of the six layers. Its content can be allocated accordingly.

For the *presentation of views*, graphic presentation exposes the interrelationships of system events and their interdependence upon each other. By visualisation, we make boundaries of safety decisions explicit, and reveal patterns such as links, inferences, and contextual relationships, that would be otherwise hard to find. In order to understand the overall safety level of ERTMS, various views of its safety architecture have to be investigated:

- Risk analysis overview. This overview includes the top-level safety architecture including risk analysis elements such as top-claim, argument and supporting evidence.
- Risk evaluation overview. This overview includes expected total risks to which the user is exposed in the form of likelihood and severity. Through consulting this view, the user is able to evaluate the integral risk analysis architecture and make judgements about the overall safety level of ERTMS.
- Scenario analysis detailed views. This view includes the scenario to be analysed. It represents combinations of flows including safety functions, hazards, consequences and mitigations and layers. This view is important for more in-depth analysis of a scenario.

5 Discussion

Some benefits of the Safety Architecture Overview Framework have been shortly explained in earlier sections. Though, there are some specific added values and challenges that require more explanation. First, the abstraction reduces complexity and emphasizes the system under consideration. This can be useful in collaborative work and should reduce ambiguity. In order to predict, explain and control risks, it is of primary importance to find a balance between concreteness and abstraction. Main challenge is to extract data without losing essential information necessary for defining the architecture. Second, incorporating structure allows better partitioning. This modelling of interfaces also allows that parts can be independently produced. Structuring the safety architecture eliminates vagueness in descriptions and clarifies tradeoffs among analyses. Third, the risk decision-maker requires an understanding of social and political issues, technical issues, management issues, and communication issues. An overview of risk analysis, risk evaluation, and detailed view of layered scenarios will improve readability and comprehension [24].

As for compatibility between top-level functionalities of the framework, hazard identification should be systematic and structured, which means taking into account factors such as system boundaries, interactions with the environment and modes of operation and environmental conditions. SysML incorporates the advantages of systematic structure of object- and process-oriented methods, which can easily describe the connection and data exchange among systems [23]. There is evidence that SysML proved its value in other safety models (see Sect. 2 about background). Information interpretation depends on information structure. Structuring information improves readability and comprehension, contributing to the creation of representations, and essential for the quality of data generation. Also, the scenarios in various layers require structure of causal relationships between the scenarios. Finally, the origin of a failure can come from decisions made earlier in the process. Complex systems come to be in the interaction of components. Baxter explains that undesirable events are simplistically seen as the result of organisational findings [25]. For these reasons, it is important to define the interactions between earlier explained layers.

6 Conclusion

The proposed framework combines safety data generation, data processing and structuring, definition of interactions and finally the creation of customized representations in order to predict, explain, and control risks by various safety experts, safety architects and safety managers.

For safety data generation, data will come from scenario-guided hazard analysis, consequences from causality analysis, risk matrices including risk acceptance criteria, and ALARP evaluations and decisions that influence the safety analyses. The focus of the Safety Architecture Overview Framework is on the properties and behaviors of functional flows and hazardous flows of the system under consideration. The structure upon which the framework is built consists of claims, evidence and arguments. The identification of causal scenarios allows safety experts, safety architects and safety managers to discover interactions between various flows and layers. Graphic representation exposes the interrelationships of events and their interdependence upon each other. By visualization, we make boundaries of safety decisions explicit, and reveal patterns such as links, inferences, and contextual relationships, that would be otherwise hard to find. The views consist of: a risk analysis overview, a risk evaluation overview, and a detailed view of scenario analyses. These views can illustrate the main interactions between the various layers and system components. Also, it is possible to illustrate the criticality of each layer and subcomponent. Explicit representation delivers insight, stimulates striving for completeness, and leads to consistency of the safety analysis.

In terms of acceptance, factors that would be of interest to the stakeholders for adoption of the framework are described in [5]. These are, among other things, more awareness and sensitivity for interrelationships between hazards and risks, but even more: comprehending the safety architecture and creating cross-discipline understanding. In response, we plan to test this framework in a real-life Dutch railway case that, at this moment, is setting up their risk analyses and evaluations.

References

1. Alexandersson, G., Hultén, S.: The Swedish deregulation path. Rev. Netw. Econ. **7**(1), 1–19 (2008)
2. European Union: Commission Decision of 25 January 2012 on the technical specification for interoperability relating to the control-command and signaling subsystems of the trans-European rail system. Off. J. Eur. Union **55**, 1–51 (2012)
3. UNIFE: UNISIG, An industrial consortium to develop ERTMS/ETCS technical specification. http://www.ertms.net. Accessed May 2018
4. Rajabalinejad, M., Martinetti, A., Dongen, L.A.M.: Operation, safety and human: critical factors for the success of railway transportation. In: Systems of Systems Engineering Conference, pp. 1–6 (2016)
5. Schuitemaker, K., Rajabalinejad, M.: ERTMS challenges for a safe and interoperable European railway system. In: Proceedings of the Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications, pp. 17–22 (2017)
6. Stoop, J.A.A.M., Dekker, S.: The ERTMS railway signaling system: deals on wheels? An inquiry into the safety architecture of high speed train safety. In: Proceedings of the Third Resilience Engineering symposium, pp. 255–262 (2008)
7. Svedung, I., Rasmussen, J.: Graphic representation of accident scenarios: mapping system structure and the causation of accidents. Saf. Sci. **40**, 397–417 (2002)
8. Kelly, T.: Arguing safety a systematic approach to managing safety cases. PhD Thesis (1998)
9. Arnold, A., Point, G., Griffault, A., Rauzy, A.: The AltaRica formalism for describing concurrent systems. Fundam. Informatica **40**(2), 109–124 (1999)
10. Cuenot, P., Chen, D.J., Gerard, S., Lönn, H., et al.: Towards improving dependability of automotive systems by using the EAST-ADL architecture description language. In: Architecting Dependable Systems IV. Lecture Notes in Computer Science, vol. 4615, pp. 39–65 (2006)
11. Güdemann, M., Ortmeier, F.: A framework for qualitative and quantitative formal model-based safety analysis. In: Proceedings of the 12th IEEE International Symposium on High-Assurance Systems Engineering (HASE), pp. 132–141 (2010)
12. Cressent, R., David, P., Idasiak, V., Kratz, F.: Designing the database for reliability aware model-based system engineering process. Reliab. Eng. Syst. Saf. **111**, 171–182 (2013)
13. Falessi, D., Nejati, S., Sabetzadeh, M., Briand, L., Messina, A.: SafeSlide: a model slicing and design safety inspection tool for SysML. In: Proceedings of SIGSOFT FSE, pp. 460–463 (2011)
14. Sabetzadeh, M., Nejati, S., Briand, L., Evensen Mills, A.: Using SysML for modeling of Safety-critical software-hardware interfaces: guidelines and industry experience. In: IEEE 13th International Symposium on High-Assurance Systems Engineering, pp. 193–201 (2011)
15. De la Vara, J.L., Panesar-Walawege, R.K.: SafetyMet: a metamodel for safety standards. In: International Conference on Model Driven Engineering Languages and Systems, pp. 69–86 (2013)
16. Biggs, G., Sakamoto, T., Kotoku, T.: A profile and tool for modelling safety information with design information in SysML. Softw. Syst. Model. **15**(1), 147–178 (2016)
17. Mauborgne, P.: Operational and system hazard analysis in a safe systems requirement engineering process – application to automotive industry. Saf. Sci. **87**, 256–268 (2016)

18. Belmonte, F., Soubiran, E.: A model based approach for safety analysis. In: International Conference on Computer Safety, Reliability, and Security, pp. 50–63 (2012)
19. Yakymets, N., Dhouib, S., Jaber, H., Lanusse, A.: Model-driven safety assessment of robotic systems. In: Intelligent Robots and Systems, pp. 1137–1142 (2013)
20. Sharvia, S., Papadopoulos, Y.: Integrating model checking with HiP-HOPS in model-based safety analysis. Reliab. Eng. Syst. Saf. **135**, 64–80 (2015)
21. Blessing, L.T.M., Chakrabarti, A.: DRM, a Design Research Methodology. Springer, London (2009)
22. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H.: Engineering Design, a Systematic Approach. Springer, Berlin, Heidelberg (2003)
23. Wang, P.: Civil Aircraft Electrical Power System Safety Assessment: Issues and Practices. Butterworth-Heinemann (2017)
24. Brussel, F.F., Bonnema, G.M.: Interactive A3 architecture overviews. Proc. Comput. Sci. **44**, 204–213 (2015)
25. Baxter, G., Sommerville, I.: Socio-technical systems: from design methods to systems engineering. Interact. Comput. **23**, 4–17 (2011)

Formalization and Reuse of Collaboration Experiences in Industrial Processes

Diana Meléndez[1](✉), Thierry Coudert[2], Laurent Geneste[2], Juan C. Romero Bejarano[1], and Aymeric De Valroger[1]

[1] Axsens bte, 20 Impasse Camille Langlade, 31400 Toulouse, France
{sofia.melendez, juan.romero, aymeric.devalroger}@axsens.com

[2] INP-ENIT, University of Toulouse, 47 Avenue d'Azereix, 65000 Tarbes, France
{thierry.coudert, laurent.geneste}@enit.fr

Abstract. Collaboration is a key factor for carrying out activities in industrial processes and an efficient collaboration is essential to accomplish an overall improvement of any process. In this article, we introduce a collaborative process-modeling framework, which allows evaluating collaboration throughout all the activities of an industrial process. The proposed framework uses experience management notions towards the creation of a repository of collaboration experiences. This experience base facilitates the reuse of past experiences to support decision making for the organization and execution of future collaborations. The article concludes by discussing the contributions and limitations of the proposed collaboration model.

1 Introduction

To confront the upcoming challenges of the market, companies must continuously evolve and improve. In order to succeed in this endeavor, an effective collaboration between companies and between people plays a central role to improve or optimize processes.

At the companies level, collaboration can be defined as the cooperative effort between two or more entities striving towards a common goal (Durugbo et al. 2011). In the last decades, the rise of outsourcing has been a strong trend for industrial firms and therefore, collaboration between companies plays a key role in the achievement of successful results in industrial processes.

At the people level, projects and industrial processes are composed by different tasks, and participants with specific characteristics contribute to these tasks. For that, participants must work together based on durable relationships and strong commitments to reach a common goal with the aim of pooling expertise and standardizing tasks (Durugbo et al. 2011).

In order to improve performance in industrial processes, the capitalization and use of past experiences is a key aspect (Skyrme 2007). More specifically, experience and knowledge management applied to collaboration processes can create value in inter-organizational activities (Lambert et al. 1999).

The overall aim of this paper is to propose a conceptual collaboration model that allows capitalizing how individuals collaborate in a process in order to reuse these collaboration experiences in the future.

This article is organized as follows: Sect. 2 describes the related works on collaboration characterization and Knowledge Management Systems applied to collaboration in processes. In Sect. 3, the collaboration model and capitalization methodology are presented. Finally, Sect. 4 presents the conclusions and discusses some limitations of the proposed model and perspectives for future research.

2 Literature Review

Collaboration has been analyzed in several studies due to its impact on the enterprise success. This section presents the current research of two key domains in our model: collaboration characterization in industrial processes and Knowledge Management Systems applied to collaboration in processes.

2.1 Collaboration Characterization in Industrial Processes

Collaborative Engineering (CE) emerged in the 1990s as an approach to structure the collective aspects of product and system design (Segonds et al. 2014). CE is defined as a technological approach that supports distributed, multi-disciplinary, and multi-organizational teams during the product development and manufacturing processes (Ma et al. 2008).

The main characteristic of CE is that the different project stakeholders are requested to work together and interact with each other in order to reach an agreement and make shared decisions (Segonds et al. 2014). To breakdown the wall between functional design and industrial design and to perform the design process with a unique team, (Mas et al. 2013) emphasize the importance of creating a new methodology that needs new procedures and new PLM tools. CE works if the team members' abilities are combined to perform complex tasks in a short time, which individual members will not be able to achieve on their own (Gogan et al. 2014).

On the other hand, Collaborative Business Processes Management - cBPM - intervenes across organizational boundaries involving actors from inside or from outside an organization (Hermann et al. 2017). In addition, (Roa et al. 2015) complement the definition with the inclusion of inter-organizational systems interactions.

Collaboration in organizations can be analyzed as complex networks as shown in Fig. 1 (Durugbo et al. 2011). They propose a mathematical model that enables to study how individuals in organizations work together to solve a problem or achieve a common goal. The two main objectives of this model are: (i) to define topologies for the information structure and (ii) to propose quantitative indicators for the information behavior that can be used to characterize collaboration in organizations. This model focuses on information flow but it does not consider other elements of the collaboration context such as contracts, commitments and indicators of quality among others.

From Durugbo's mathematical model three indicators of collaboration have been proposed: Team work scale; Decision making scale; Coordination scale. The team

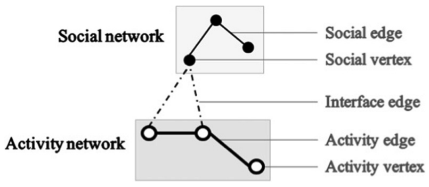


Fig. 1. Collaboration as a graph (Durugbo et al. 2011).

work scale measures the ease with which social vertices can pool resources, it is calculated by aggregating two mathematical measures: the clustering coefficient and the centrality degree of the collaboration graph. This indicator allows for assessing the activity of an actor and interconnectedness within a cluster for teamwork. The decision making scale measures the ease with which social vertices can make choices, it is calculated by aggregating the clustering coefficient and the closeness degree. This indicator assesses the ease with which an actor within the intra-organizational network can make decisions based on the interconnectedness and connections for relationships. Finally, coordination scale measures the ease with which social vertices can harmonize interactions; it is calculated by aggregating the closeness and the centrality degree. These indicators permit to characterize the performance of collaboration between actors to perform activities.

2.2 Knowledge Management Applied to Collaboration in Processes

Principles of Knowledge Engineering (KE) have been introduced in cBPM towards a collaboration model based on ontologies and deduction rules in order to build a collaboration information system (Rajsiri et al. 2010). This model is a collaborative process model that describes interactions and information exchanges between business partners. This work proposes a higher abstraction level of a given collaboration. It allows characterizing collaboration from existing knowledge. Therefore, the precision of collaboration characterization strongly relies on the quantity and quality of the knowledge provided by business partners (i.e. the experts). High-level knowledge such as general deduction rules are difficult to implement in specific contexts. Thus, it is necessary to have a detailed level of knowledge modelling consistent with an actual context in order to be able to deduce general knowledge based on actual experiences.

The systematic reuse of experience in industry allows making better use of experiences during an industrial process. Experience Management (EM) supports the capture, storage, search, and retrieval of past experiences (De Mendonça Neto et al. 2001) and its ultimate goal is experience reuse (Bergmann 2002).

Accompanying this logic of experience reuse, Case Based Reasoning (CBR) is an approach that facilitates the resolution of problems by recovering, adapting and reusing previous experiences. This approach requires the characterization of the context in which the experiment took place and the lesson learned in this context for solving a given problem (Kolodner 1993).

In summary, the main purpose of this paper is the use of Experience Management principles in order to establish a model of collaboration experience, capitalize the contributions of each actor throughout the activities of a collaborative process and reuse experiences to improve the future execution of the process or the definition and execution of similar processes.

In the next section we will describe the experience feedback process, the elements of the collaboration model and an illustrative application based on a real process execution.

3 Experience Feedback Process for Collaboration

3.1 Experience Feedback Process

The main goal of this study is to propose an experience feedback process in order to capitalize experiences of collaboration in industrial processes and to reuse them to define future collaborations. The capitalization of an experience is done for all activities of a process as shown in Fig. 2, it is formalized through the elements of the knowledge base in order to standardize the capitalization and facilitate the future reuse of past experiences.

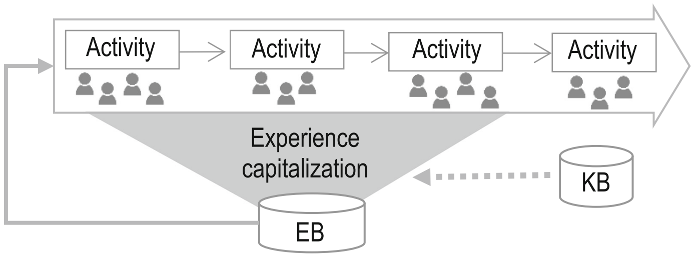


Fig. 2. Overall experience feedback process

In order to facilitate the reuse of experiences, it is necessary to define a collaboration model corresponding to a generic experience frame. This collaboration model is stored in a knowledge base. Every collaboration experience will be an instance of this model. In order to define the characteristics of a collaboration experience without ambiguity, it is necessary to standardize the main concepts and to store them. Therefore, the knowledge base also contains a taxonomy of concepts that are used to characterize the different elements of the experience. Once an experience has been properly defined from the available knowledge, it is stored into the Experience Base (EB). It is important to be able to capitalize the planned collaboration and the actual one within an experience, since this will allow to compute some performance indicators corresponding to the experience. When a process has to be planned for a new execution, the prior experiences stored into the Experience Base, and corresponding to the activities of the process, can be reused.

The collaboration model and the taxonomy are described in the next section.

3.2 Collaboration Model

This section describes the collaboration model that allows standardizing the experience capitalization and which is used by the experience feedback process. The concepts organized in the taxonomy are also presented. An experience is modeled by an oriented graph which is based on the collaboration model as shown in Fig. 3.

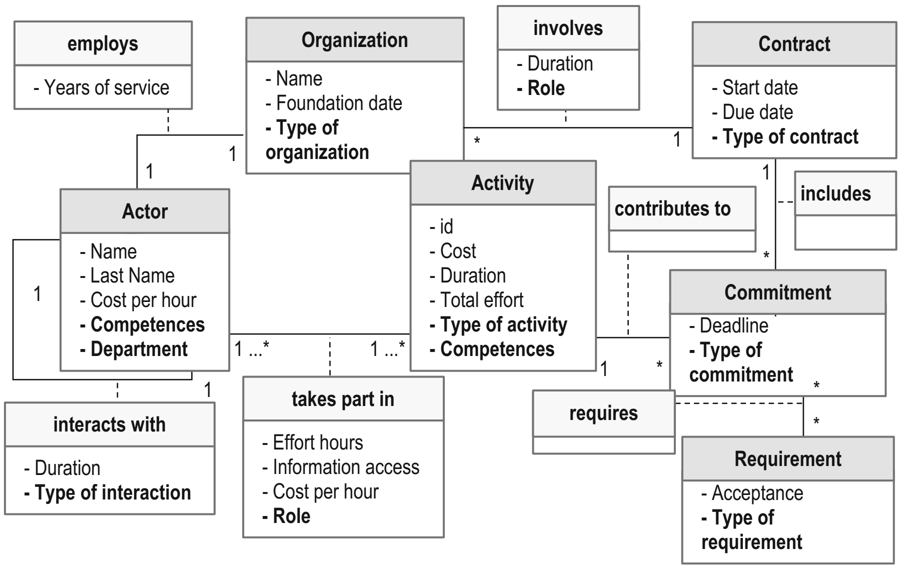


Fig. 3. Collaboration model frame

The proposed collaboration model is based on the execution of an industrial process. Different **organizations** can contribute to the execution of several activities of an industrial process in order to reach for defined goals. These goals are represented in our model by **commitments** and they must accomplish one or several **requirements**. In order to formalize the different participation of the organizations, they are governed by **contracts**. The interaction between the organizations to fulfill the commitments of a contract engenders an industrial process. It is a structured, managed and controlled set of **activities** with the purpose of transforming inputs into specified outputs. During the execution of the process activities, **actors** collaborate in order to reach the process commitments. Every actor exerts different roles and contribute to one or several activities throughout the process.

Two or more organizations are **involved** by a contract in order to achieve one or several commitments, and the contract **includes** all the agreed commitments. Furthermore, these commitments must contribute to fulfill one or several requirements. In the model this relation is named **requires**. The commitments are the result of one or several activities of the process, this means that one activity **contributes to** one or several commitments. At this level, there are the interactions between people to execute an activity. Thus, an actor **takes part in** one or several activities and he **interacts with** one or several actors during the execution of the activity. Figure 3 shows the set of elements of the proposed collaboration model.

An organization is a group of people, structured in a specific way to achieve shared commitments. For this element, we must identify the name, the foundation date and the type of organization. A **contract** represents one or several agreements where an organization provides goods or services to another organization; these agreements can also be verbal, what allows starting collaborative activities without a written contract. For this element, we must identify the start date, end date and the type of contract. A **commitment** in the proposed collaboration model represents the output of a process activity. It is characterized by a type of commitment classified in: product, report, service and systems for example. A **requirement** is a specific need that the commitments have to meet. For this element, we must identify the type of requirement. An **activity** of an industrial process describes the work, which transform one or several inputs in intermediate or final outputs of the process. The following information must be identified for each activity: cost, duration, total effort, and type of activity. The cost attribute includes the cost of all actors who participate in the activity and others costs such as material cost, transportation cost, etc. The duration attribute is the difference between the start date and the due date. The total effort is the sum of all workloads in person-hours needed to carry out the activity. An **actor** a person who participate in one or more activities of an industrial process. They are characterized by: name, cost per hour, department and one or more competences.

For the relations between vertices, the main relations are: **Takes part in, Interacts, Includes, Contributes, Involves, Requires** and **Employs**.

The relation **"Takes part in"** is the relation between an actor and an activity. It is the contribution of the actor for a given activity and it is characterized by the total number of hours required by the actor to execute his/her contribution otherwise the actor's effort. Another characteristic is the information access. We propose to measure this indicator with a number between the 0 and 1. The value 1 indicates that the information necessary to carry out an activity is easily obtainable. The value 0 means that it is impossible to access to the information. The relation **"Interacts"** is the relation between an actor i and actor j. The relation **"Contributes"** is the relation between an activity and a commitment, it indicates which activity contribute to a commitment. The relation **"Requires"** is the relation between a commitment and a requirement, it represents the requirements that must be met. The relation **"Involves"** represents the relation between an organization and a contract. It is characterized by the duration and the organization's role for a specific contract. The relation **"Includes"** is the link between a contract and a commitment. A contract may have one or several commitments. The relation **"Employs"** represents the link of work between an actor and an organization. An actor cannot have a direct link to two or more organizations.

The attributes of vertices and edges must be standardizing in order to facilitate the future reuse. Then, a taxonomy of concepts allows this standardization and it ensures an accurate capitalization.

3.3 Taxonomy of Concepts

Each vertex and some edges must be characterized from a taxonomy of concepts. An example of taxonomy, which can be used for the characterization of collaboration experiences, is represented in Fig. 4. A taxonomy is a hierarchical structure described through relations between concepts included in the hierarchy (Van Rees 2003). Taxonomies create a consistent representation of concepts through their structuration into a tree according to their similarity (Jabrouni et al. 2011).

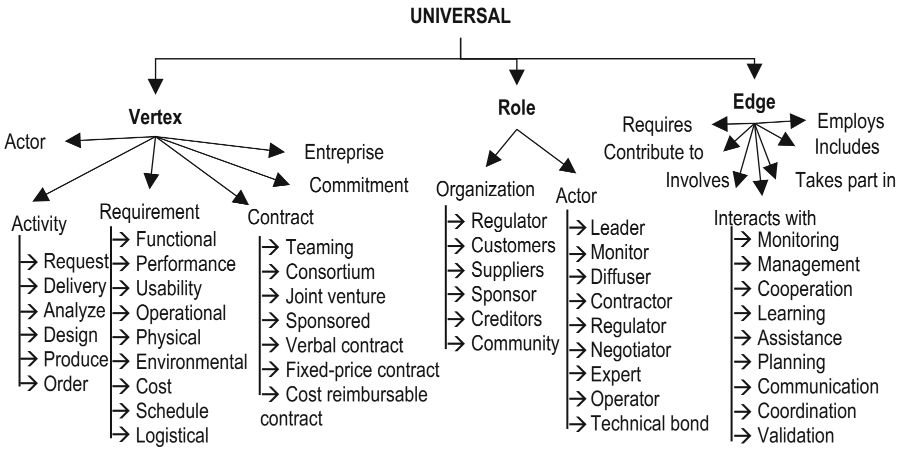


Fig. 4. Extract from taxonomy of concepts

In our work, taxonomies are defined for some attributes in order to characterize the collaboration experiences and facilitate their retrieval into the experience base where all experiences will be stored. This will be develop in Sect. 3.4. An example of taxonomy for collaboration experiences is represented in Fig. 4. This taxonomy of concepts is based on existing taxonomies proposed by (System Requirements - SEBoK, 2015) for requirements, (Boucher et al. 2007) for actor's roles and (Mayer et al. 2012) for contracts.

3.4 Collaboration Experience Building

The knowledge base contains the collaboration model frame to structure a collaboration experience, and the taxonomy to characterize, with validated and standard concepts, all the elements of a collaboration experience. The KB is essential in our model because it facilitates the experience formalization and reuse. In addition to the elements and their interactions previously described, it is necessary to distinguish two stages of the collaboration experience: the planned collaboration and the actual collaboration.

The first stage is the planned collaboration where all the necessary actors, activities, commitments, requirements, contracts and organizations of the process to execute are included. They are planned a priori. Figure 5 shows an example of an instantiation for one vertex actor and one vertex activity. For each element, there are certain attributes for which their values will be found in the proposed taxonomy. In Fig. 5, for the vertex *activity 1*, the given value for the attribute competence is "*Quality control*" and the given value for the attribute type of activity is "*Production*". Both values are coming from the taxonomy of concepts.

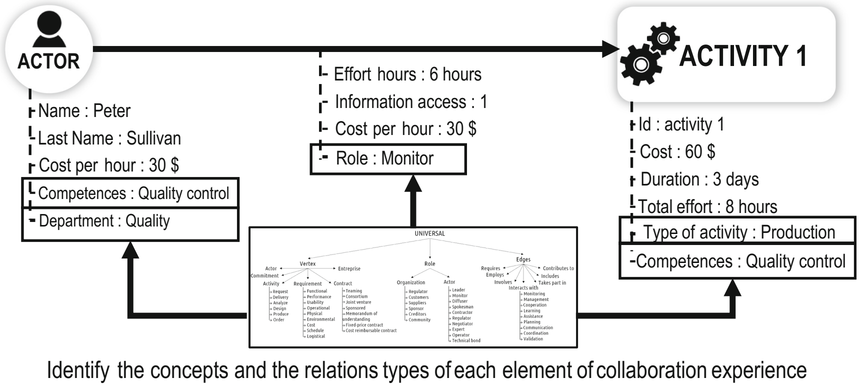


Fig. 5. Example of instantiation and link with taxonomy for two elements

Figure 6 represents an example of a planned collaboration instance. It is the instance of a real case of a process in the consulting sector.

The second level is the actual collaboration. It means the actual information of the process execution. The Fig. 7 represents the changes of the execution of planned collaboration experience, which mainly concerns the activity 2; the other elements have not been represented because they are identical to the planned experience. This allows the calculation of performance indicators in order to identify the gap between the planned collaboration experience and the actual collaboration experience. These performance indicators are commitment acceptance, process delay and respect of the budget among others.

The second step of the proposed approach is the storage in the Experience Base (EB) (after validation) of the planned collaborative experience, the actual collaborative experience and the indicators. When the EB has a significant number of experiences, the information can be reused to facilitate the decision making process of future collaborations. The reuse of experiences is described in the next section.

3.5 Collaboration Experience Reuse

The main objective of the Experience Feedback Process is the reuse of past experiences to improve current situations. In order to fall within this reuse logic, we have proposed a characterization of the context of a process by using a labelled graph. In the Sect. 3.2, we defined a collaboration model frame of a process that forms the basis of the search mechanism for similar experiments. This frame allows creating an experience base and

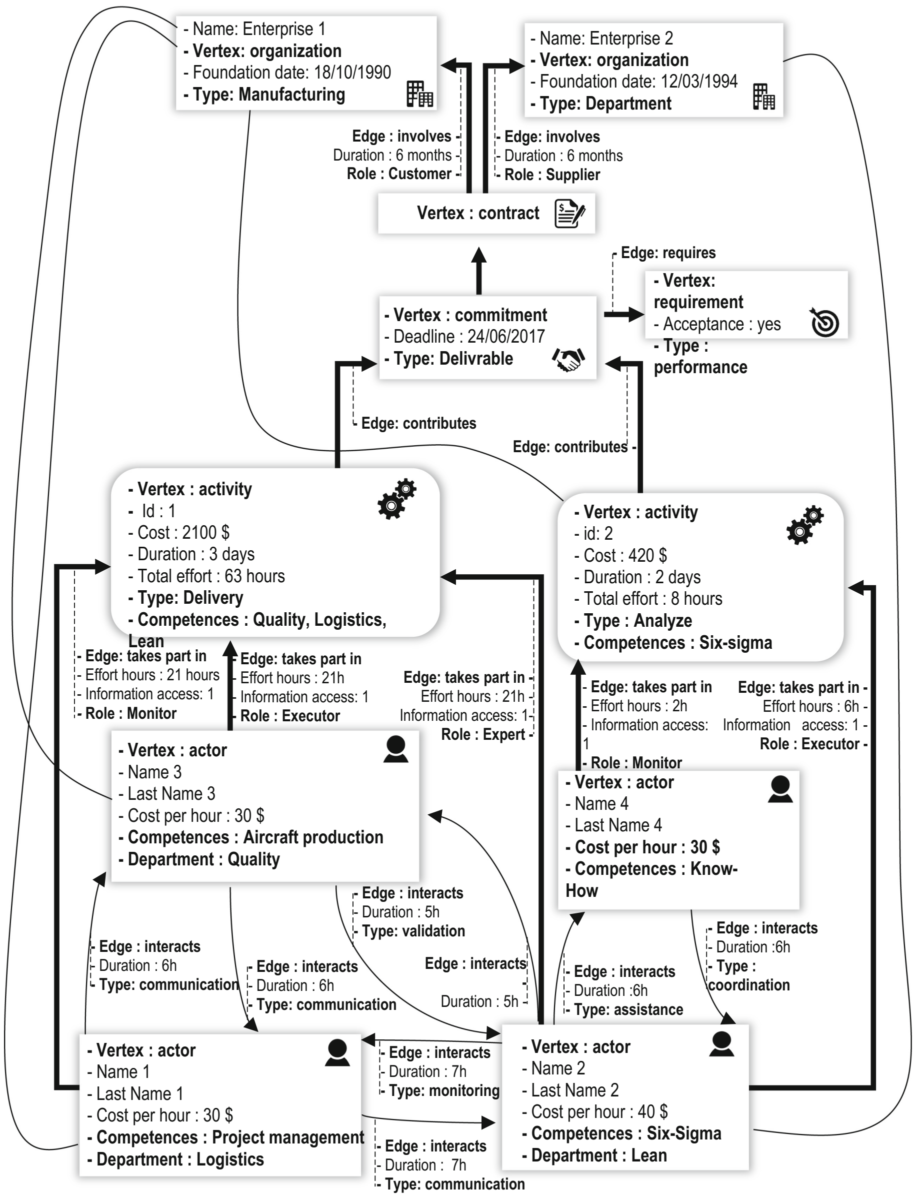

- Name: Enterprise 1
- Vertex: organization
- Foundation date: 18/10/1990
- Type: Manufacturing
- Name: Enterprise 2
- Vertex: organization
- Foundation date: 12/03/1994
- Type: Department
Edge : involves - Duration : 6 months - Role : Customer
- Edge: involves - Duration : 6 months - Role : Supplier
Vertex : contract
- Edge: requires
- Vertex: requirement - Acceptance : yes - Type : performance
- Vertex : commitment - Deadline : 24/06/2017 - Type: Delivrable
Edge: contributes
Edge: contributes
- Vertex : activity - Id : 1 - Cost : 2100 $ - Duration : 3 days - Total effort : 63 hours - Type: Delivery - Competences : Quality, Logistics, Lean
- Vertex : activity - id: 2 - Cost : 420 $ - Duration : 2 days - Total effort : 8 hours - Type : Analyze - Competences : Six-sigma
- Edge: takes part in - Effort hours : 21 hours - Information access: 1 - Role : Monitor
Edge: takes part in - Effort hours : 21h - Information access: 1 - Role : Executor
Edge: takes part in - Effort hours : 21h - Information access: 1 - Role : Expert
- Edge: takes part in - Effort hours : 2h - Information access: 1 - Role : Monitor
Edge: takes part in - Effort hours : 6h - Information access: 1 - Role : Executor
- Vertex : actor - Name 3 - Last Name 3 - Cost per hour : 30 $ - Competences : Aircraft production - Department : Quality
- Vertex : actor - Name 4 - Last Name 4 - Cost per hour : 30 $ - Competences : Know-How
- Edge : interacts - Duration : 5h - Type: validation
- Edge : interacts - Duration : 6h - Type: communication
- Edge : interacts - Duration : 6h - Type: communication
Edge : interacts Duration : 5h
- Edge : interacts - Duration :6h - Type: assistance
- Edge : interacts - Duration :6h - Type : coordination
- Vertex : actor - Name 1 - Last Name 1 - Cost per hour : 30 $ - Competences : Project management - Department : Logistics
- Edge : interacts - Duration : 7h - Type: monitoring
- Edge : interacts - Duration : 7h - Type: communication
- Vertex : actor - Name 2 - Last Name 2 - Cost per hour : 40 $ - Competences : Six-Sigma - Department : Lean

Fig. 6. Example of planned collaboration experience

to develop a mechanism of research to build on previous experiences. These experiences can then be used to improve the selection of key actors for similar processes. This choice could be done by one of several criteria of the collaboration frame and the characteristics of the actor. For example, given a non-quality situation in an industrial process, the process of problem solving can be improved thanks to the experience base

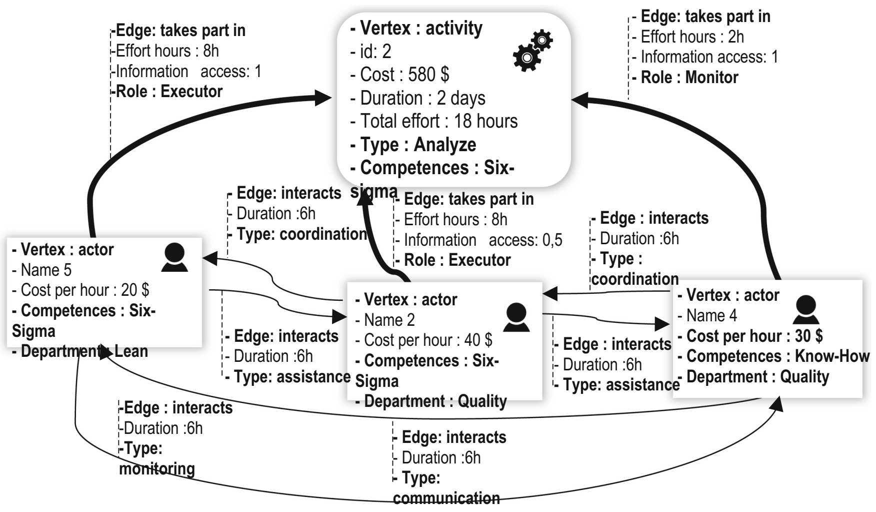


Fig. 7. Example of an actual collaboration experience (activity 2)

where the actors of the process who have previously participated in the problem solving can be identified and filtered by more specific characteristics such as product type, years of experience or competences.

4 Conclusion

Due to the increasing complexity of industrial processes with outsourcing activities, collaboration has become one of the relevant areas of performance measurement. The analysis of the collected literature indicates that there is a lack of methodologies for collaboration characterization between companies based on the characterization and performance of team collaboration, as well as an absence of a formal inclusion of experience feedback process.

In this article, a collaboration model for experience collaborations characterization has been defined. This model allows characterizing the collaboration experience in two stages: (i) team stage and (ii) company stage. The collaboration model proposed within the formalization of elements such as contracts, commitments and requirements is novel. Also, this article has shown the importance of experiences capitalization and reuse, in order to improve and to facilitate future collaborations in industrial processes.

Despite the model described in this article allows the calculation of performance indicators focused on requirements, activities and actors, it is important to notice that the quality of the collaboration process cannot be evaluated. Therefore, the perspectives of this research are to propose some new indicators which will reflect how good is the collaboration within an experience. This will enable to characterize how two or more organizations are collaborating within the experiences.

From these quality indicators, it will be possible to help to define efficient associations of organizations following the past experiences with regard to collaboration.

Finally, the experience feedback process is still at a preliminary stage and requires further development. The first axis of development is the definition of (i) a method to reuse experiences and (ii) a mechanism to generalize several experiences in knowledge.

References

Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Springer, Heidelberg (2002)

Boucher, X., Bonjour, E., Grabot, B.: Formalisation and use of competencies for industrial performance optimisation: a survey. Comput. Ind. **58**(2), 98–117 (2007)

Durugbo, C., Hutabarat, W., Tiwari, A., Alcock, J.: Modelling collaboration using complex networks. Inf. Sci. **181**(1), 3143–3161 (2011)

Gogan, L., Popescu, A., Duran, V.: Misunderstandings between cross-cultural members within collaborative engineering teams. Procedia Soc. Behav. Sci. **109**, 370–374 (2014)

Hermann, A., Scholta, H., Bräuer, S., Becker, J.: Collaborative business process management-a literature-based analysis of methods for supporting model understandability. In: Proceedings Internationale Tagung Wirtschaftsinformatik, vol. 13, pp. 286–300 (2017)

Jabrouni, H., Kamsu-Foguem, B., Geneste, L., Vaysse, C.: Continuous improvement through knowledge-guided analysis in experience feedback. Eng. Appl. Artif. Intell. **24**(8), 1419–1431 (2011)

Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann (1993)

Lambert, D., Emmelhainz, M., Gardner, J.: Building successful logistics partnerships. J. Bus. Logistics **20**(1), 165 (1999)

Ma, Y.-S., Chen, G., Thimm, G.: Paradigm shift: unified and associative feature-based concurrent and collaborative engineering. J. Intell. Manuf. **19**(6), 625–641 (2008)

Mas, F., Menéndez, J., Oliva, M., Ríos, J.: Collaborative engineering: an airbus case study. Procedia Eng. **63**, 336–345 (2013)

Mayer, D., Warner, D., Siedel, G., Lieberman, J.: The Law, Sales, and Marketing (2012)

De Mendonça Neto, M.G., Seaman, C., Basili, V.R., Kim, Y.M.: A prototype experience management system for a software consulting organization. In: SEKE, pp. 29–36 (2001)

Rajsiri, V., Lorré, J.P., Benaben, F., Pingaud, H.: Knowledge-based system for collaborative process specification. Comput. Ind. **61**(2), 161–175 (2010)

Roa, J., Chiotti, O., Villarreal, P.: Detection of anti-patterns in the control flow of collaborative business processes. In: Simposio Argentino de Ingeniería de Software - ASSE 44 (2015)

Segonds, F., Mantelet, F., Maranzana, N., Gaillard, S.: Early stages of apparel design: how to define collaborative needs for PLM and fashion? Int. J. Fash. Des. Technol. Educ. **7**(2), 105–114 (2014)

Skyrme, D.: Knowledge Networking: Creating the Collaborative Enterprise. Routledge (2007)

Pyster, A., et al.: Guide to the systems engineering body of knowledge (SEBoK) v. 1.0. 1. Guide to the Systems Engineering Body of Knowledge (SEBoK) (2012)

Van Rees, R.: Clarity in the Usage of the Terms Ontology, Taxonomy and Classification. CIB REPORT 284. 432, pp. 1–8 (2003)

An MBSE Framework to Support Agile Functional Definition of an Avionics System

Jian Tang[1], Shaofan Zhu[1], Raphaël Faudou[2], and Jean-Marie Gauthier[2](✉)

[1] Beijing Aeronautical Science and Technology Research Institute of COMAC, Future Science Park, Changping District, Beijing, China
{tangjianl, zhushaofan}@comac.cc

[2] Ethics Biotope, Samares Engineering, 2, Av. Escadrille Normandie Niemen, 31700 Blagnac, France
{raphael.faudou, jm.gauthier}@samares-engineering.com

Abstract. In avionics domain, there have been many efforts in recent years to build a MBSE methodology with tooling support. The main purpose is often to improve quality and efficiency of system definition, architecture and integration. Sometimes there is also an additional objective to ease system verification and validation. This paper introduces an additional challenge with the support of an agile development cycle to ease impact analysis and incorporation of late and changing requirements at different times. It presents key principles and requirements of an agile MBSE approach and presents associated modeling activities with illustration on an avionics case study.

1 Introduction

Model-Based Systems Engineering (MBSE) is a recognized good practice to improve detection of errors and ambiguities in requirements early in the system development life cycle and its adoption is rising in industry [1]. Concurrently, there is a growing demand for support of an agile Systems Engineering approach and traditional SE methods have started to adapt mainly by supporting MBSE as means to be more agile [2]. But an agile MBSE approach requires a modeling framework to be responsive for changes from requirements and currently there is not much support for that. This is quite an interesting challenge, especially in avionics domain where there are certification rules and guidelines that require all engineering artifacts to be traced to requirements.

Agile approaches rely on iterative and incremental development cycle with the objective to frequently deliver values that satisfy the customer. In 2001, the agile manifesto[1] was published, listing the common principles to all agile method. Those principles, mainly applied on software development, are now spreading in the systems engineering community.

In addition to the agile principles, continuous integration and test driven development are common practices in the software engineering domain. We believe that this capability can be applied in an MBSE context, meaning that a system model should be

[1] http://agilemanifesto.org/principles.html.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 168–178, 2019.
https://doi.org/10.1007/978-3-030-04209-7_14

continuously validated regarding the requirements. Finally, the International Council on Systems Engineering (INCOSE) has identified needs and potential outcomes to the practice of agile methods in a Systems Engineering context [3, 4]. For instance, Quick Reaction Capability (QRC) is an identified need for an agile acquisition process.

The main contribution of this paper is the presentation of an agile MBSE framework that supports system definition compliant with ARP4754A [5]. The objective of the proposed agile MBSE approach is to improve responsiveness of modelling activities when requirements are modified or when new requirements are given by customers or by system domain experts (safety, physical simulation, control, etc.). To achieve this objective, the proposed approach relies on several steps that permits to capture requirements and to perform impact analysis on actual architectures using the Systems Modeling Language (SysML) [6].

The rest of this paper is organized as follows: Sect. 2 presents the key principles of the proposed MBSE approach. Section 3 presents introduces requirements to achieve agile MBSE. Section 4 details the case study and discusses experiment results with regards to agile approach. Finally, after reviewing the related work in Sect. 5, we conclude and outline further work in Sect. 6.

2 Key Principles of the Proposed Agile MBSE Approach

At the beginning, all functional needs or requirements are collected from the different stakeholders identified for the System of Interest (SoI). They are imported in the model to be easily manipulated and related for traceability. Then, the proposed MBSE framework is made of organized sets of model elements and 4 modeling activities that populate and update those elements with traceability.

2.1 Capture Intended Behaviors from Input Functional Needs

The first principle of the approach consists in structuring the functional needs around system functionalities using *Use Cases* (UC) and scenario-based approach. A functionality (from ISO15288 [7]) represents a set of interactions of system with its operating environment to achieve a functional goal. We use UML use case recognized modeling technic to identify functionalities of the system [8, 9]. Identification of use cases provides a set of independent high-level functionalities of the system, what is a very interesting property for teamwork in an agile context: each person of the team can work on a different use case with some autonomy.

A use case has a simple documentation that defines the starting event(s), the goal, the end success conditions and the identified errors. Use cases are traced to input functional needs/requirements with SysML *Refine* link to monitor progress and ease review of their analysis.

After structuring functional needs within use cases, we propose to capture the expected behaviors using black-box operational scenarios. The black-box operational scenarios are defined as SysML *Interactions* owned by a use case (owned behaviors). Those interactions are represented graphically by *Sequence Diagrams* (SD). For each use case, the focus is put on functional messages between the SoI and use case's associated actors. The first message shows the triggering event of the scenario while the expected responses of system is described with reflexive messages attached to the SoI lifeline. The capture of those reflexive messages is useful to provide a first list of top level functions, but it is not main priority at that stage because that list is likely to change when other requirements are considered. Focus is rather put on functional interfaces, i.e. the incoming and outgoing functional messages, and on the causal dependencies between those functional messages. The goal is to deduce consistent functional scenarios that cross the SoI, i.e. interactions between the SoI and its functional operating context represented by actors.

2.2 Define System Functional Interfaces and Top-Level Functions

The second principle concerns definition of functional interfaces and top-level functions. Concerning functional interfaces, an *Interaction message* cannot fit because it only addresses one usage, not a definition. Hence, a concept is needed to define functional interfaces exchanged between the SoI and its operating context, but also one to define the associated event matching reception of the message by the system. For this purpose, UML *Signal* and *Signal Event* are used to support those definition concepts.

Concerning the definition of top-level functions, ARP4754A recommends using function-based approach from aircraft level down to software and hardware items. We want to keep that recommendation and have the same concept to formalize functions at different engineering levels, in an iterative and incremental approach. Furthermore, this concept shall support breakdown relation to go from one level to another, as well as definition of inputs and outputs. For this purpose, SysML *Activity* is used to define functions and *CallBehavior* is used for functions call. Using SysML activity to define function is debatable as we get two options:

1. either formalize functions as *blocks* and complete each *block* with an *activity* containing *actions* to define behavior and allow simulation,
2. or formalize functions as *activities* and complete those *activities* with *actions* to define behavior.

Both options would work but the first one requires synchronization of block interface (ports) and activity interface (parameters) for consistency. This is additional effort compared to option 2.

The proposed MBSE framework enables the automation of functional interfaces and top-level functions synthesis from all the different validated use cases and scenarios. As use cases are independent, it means that it is possible to identify by construction a full set of top-level functions without overlap. It is a good property to ensure consistency between functions. Furthermore, traceability between sequence diagrams and generated top-level functions is ensured. In the same time, the proposed framework can generate executable actors' behavior to prepare functional simulation and continuous validation of the system in its operating context.

2.3 Define Functional Architecture

A functional architecture is a structural arrangement of functions that specifies their relations. It includes the hierarchical refinement between levels, the flows between functional interfaces of a parent functions, functional interfaces of children functions, and flows between functions of same level. The third principle concerns the design and refinement of top-level functions. The goal is to refine each top-level function with lower-level technical functions identified from domain knowledge while considering non-functional requirements. The Fig. 1 gives an example of top-level function refined by interface functions and internal functions. We use SysML *Activity* as behavioral container, *Action* nodes to express behavioral elements, and *Control Flow* to specify their execution logic. In addition, functional interfaces of the SoI are specified using S*end Signal Action* (to send a signal), A*ccept Event Action* (wait for a given event), and "*call behavior actions*" represent function calls. Each functional interface is embedded in interface function for further functional refinement.

We reuse same functional definition framework to manage all functions and their functional interfaces. UML *stereotypes* are used to tag functions according to their source (required function, technical functions, product functions) to better identify owners of the functions. This refinement is performed recursively down to a level where either function specification is clear enough for allocation or realization without further decomposition, or there are identified products that realize it.

During this step, functions interfaces (inputs and outputs) are identified. This identification is mainly driven by data/energy continuity principle. This principle ensures that any data/energy needed as input by a function is produced by another function or comes directly from SoI context. By following this principle, we can identify required inputs decide if those inputs are provided by the input signal that triggered top-level function, by another function or are already available in the system. In that latter case it means that we need a function to retrieve/read that information.

Concurrently to the refinement of top-level functions, it is also important to define their validity over time, i.e. conditions of the mission when activation of those functions is valid. The proposed approach suggests using SysML *state machine* to represent the different states and transitions that formalize the mission profile. It is possible to set input signal of the function as *trigger* of a state *transition*. Next, the functions are executed either as *effect* of a transition or as a *doActivity* of a target state, as illustrated in Fig. 2. Progressively, all functions are connected to the state machine that is the top-level system behavior.

2.4 Simulation of Scenarios

The fourth principle of the proposed MBSE approach concerns support of functional simulation using *fUML* [10]. For instance, the Fig. 3 depicts an *fUML* simulation. The left and right sides of the Figure show the behaviors of the actors that stimulate the functional architecture at the center of the Figure. The goal here is to verify that the SoI behaves as expected in the different validated scenarios once they have been combined in a centralized definition. This simulation can be automated but can also be interactive so that the customer may use it as a mean for early functional validation. At any time,

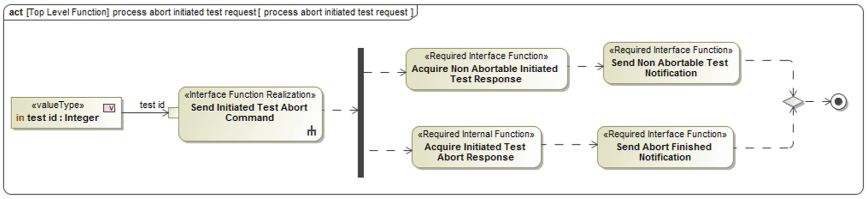

act [Top Level Function] process abort initiated test request [process abort initiated test request]
«valueType»
in test id : Integer
test id
«Interface Function Realization»
Send Initiated Test Abort Command
«Required Interface Function»
Acquire Non Abortable Initiated Test Response
«Required Interface Function»
Send Non Abortable Test Notification
«Required Internal Function»
Acquire Initiated Test Abort Response
«Required Interface Function»
Send Abort Finished Notification

Fig. 1. Example of top-level function formalized through a SysML activity

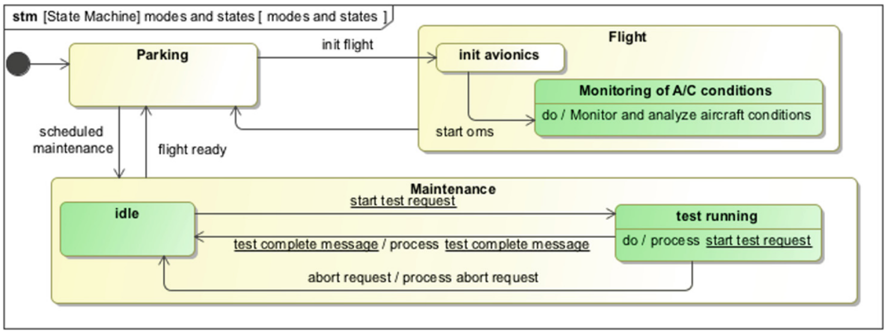

stm [State Machine] modes and states [modes and states]
Parking
init flight
Flight
init avionics
Monitoring of A/C conditions
do / Monitor and analyze aircraft conditions
start oms
scheduled maintenance
flight ready
Maintenance
start test request
idle
test running
do / process start test request
test complete message / process test complete message
abort request / process abort request

Fig. 2. Simplified example of state-machine containing top-level functions

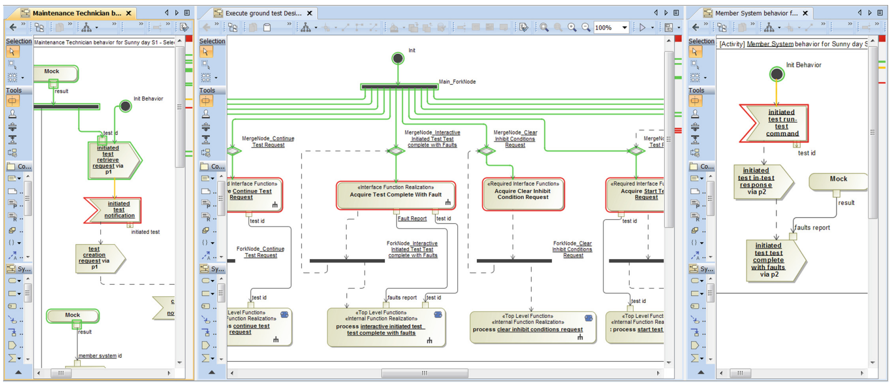

Maintenance Technician b...
Execute ground test Desi...
Member System behavior f...
Main_ForkNode
Init Behavior
Mock
result
test id
faults report
Acquire Test Complete With Fault
Acquire Clear Inhibit Condition Request

Fig. 3. Simulation of functional architecture for validated scenario

one can select a validated UC scenario and use it to simulate functional architecture model in its operational context.

When simulation passes for all validated scenarios, then the model contains a consistent set of system interfaces and top-level functions that address all source functional needs capturing by scenarios validated by customer. Hence, using model simulation helps in preparing V&V and continuous integration of systems models in an agile context. Furthermore, ensuring consistency in the model is a key factor to derive a set of system requirements with good quality (correct, necessary, unambiguous, complete, consistent with each other), and traced to stakeholder functional requirements.

3 Requirements for Agile System Definition and V&V

The proposed MBSE approach aims at supporting late arrival or late analysis of functional requirements, with immediate integration of new functions and updated functions into current definition. As we apply a scenario-based approach for functional requirements, customers must be able to analyze and validate changes in use case and scenarios at any time. Indeed, in an agile context, the customer is involved in the development process, especially to write and review user stories. Therefore, logics (if/else and loops) are prohibited as this behavior is likely to change during functional refinement. This leads to a first requirement:

REQ 1: *The framework shall enable customers to validate use cases and scenarios. The scenarios shall remain conceptual and functional to be easily understood by customer and validated at some point in time.*

Changes in use cases and scenarios shall be immediately reflected into the current functional definition. Such immediacy implies a second requirement:

REQ 2: *The framework shall be able to identify new and updated functions from scenarios and inject those changes into functional definition model.*

In addition, the signals identified from input and output messages shall also be updated when new scenarios are created or modified. We call those signals "*system external functional flows*" as they represent functional exchanges between the SoI and its environment.

REQ 3: *The framework shall be able to identify new and updated system external functional flows from scenarios.*

Analyzing changes in an agile context is a strong asset to perform risk analysis. Hence, the definition model elements (*signals* an *activities*) shall be traced to source requirements, but through the scenarios to get easy support of change management thanks to the traceability links. In addition, traceability is mandatory when developing safety critical systems.

REQ 4: *The framework shall be able to provide complete traceability between requirements, scenarios, and functional architecture.*

To maintain the capability of validating functional definition at any time, the framework shall support the creation of simulated functional behavior for system operating environment (represented by *actors* interacting with SoI).

REQ 5: *The framework shall be able to generate and update behavioral model to stress the system with inputs coming from actor's behavior involved in the context of a given UC scenario.*

Teamwork is also an important part of agile method. Each team member should be able to deliver value on the model without overlaps:

REQ 6: *The framework shall be able to assess the independence of use cases. In case a function is shared between several use cases, then the granularity of uses cases should be modified to avoid overlap.*

4 Experiments and Discussion

To assess the relevance of those activities and illustrate them, we used the Onboard Maintenance System (OMS) as an industrial case study. The OMS is a software intensive avionics system that monitors the health of the aircraft during flight and supports run of tests while on ground. The main functions of the OMS are as follows:

- Monitor continuously aircraft avionics during flight
- Analyze faults, diagnosis of the root cause, and alert crew
- Inform maintenance crew of needed repairs
- Perform on-ground testing on aircraft avionics

In this experiment we focused on the operational use of the OMS without considering the physical aspects (electrical, thermal, …) of the system. Then, since the OMS is a software intensive system, we mainly specified the discrete aspect of the system. Experiments were conducted using *Cameo suite (Systems Modeler, Simulation Toolkit and DataHub).* Requirements were imported from *DOORS* Database.

The approach was applied in an engineering team comprising 7 systems engineers. As depicted by the Fig. 4, high-level functional needs were captured from input functional requirements and structured with use cases. Then, each use case was assigned to one systems engineer whose goal was to provide black-box operational scenarios. For each black-box scenario, the proposed approach was performed iteratively: scenario writing, scenarios combining (automated) and scenarios simulation.

4.1 Results

During functional needs and requirement capture, 8 main use cases and 4 actors were identified for OMS. Among those use cases, 3 were considered as representative enough to be detailed through the MBSE approach:

- Execute ground tests: this use case has a lot of interactions with aircraft equipment and with maintenance technician.
- Aircraft Condition Monitoring Function (ACMF): it provides important functionality with regards to failure messages and good health of aircraft equipment.
- Upload data to target member system: it provides important functionality with regards to aircraft equipment configuration.

From those 3 use cases, and with help of textual functional needs and requirements, 17 operational black-box scenarios were created, containing 108 functional messages between actors and the OMS. The scenarios permitted to generate 71 signal definitions all traced to the initial messages for impact analysis. Then, scenarios were combined to create 18 executable top-level functions, 15 internal sub-functions, and 51 interface functions that define the first level of the functional architecture model.

Concerning the "agile performance" of the approach here are our findings from the experiments. First, it appears that the independence of use cases is a strong asset for parallel teamwork. Indeed, we did not find any shared function between the use cases. This is interesting as each team member can go from scenario down to functional design without overlapping other team members work, at least at the functional level.

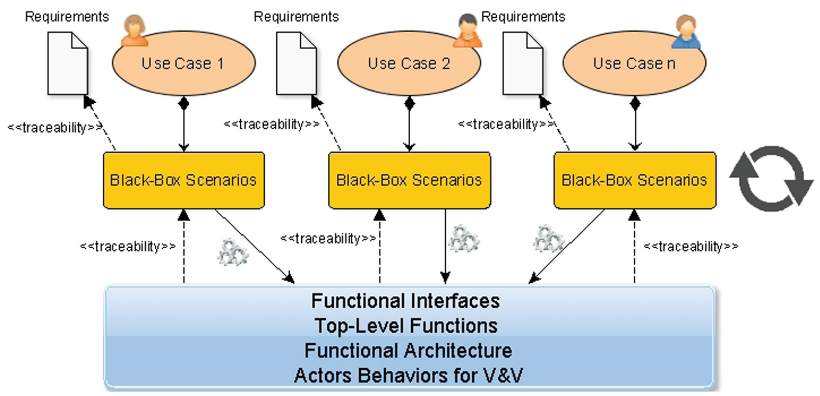


Fig. 4. MBSE approach for agile functional definition

Therefore, everyone was able to contribute to the same model at the same time by delivering frequently added values to the system models.

The second finding concern traceability. In the proposed approach, interaction messages are traced input requirements (when direct capture), and each extracted signal, interface function and internal function are traced to the messages that generate them. Therefore, when modifying a scenario or when adding a new one, team members were able to identify impacts on the functional architecture under design.

Finally, the third finding concern continuous validation. Indeed, the simulation context representing actors' behaviors permitted to validate the functional architecture during the design phase.

4.2 Discussion

Let us now discuss the use case modelling approach. For use case identification we may have used the functions allocated from upper engineering level (aircraft and avionics system in our case). However, there is often overlap between those functions and it is not easy to detect overlaps because of limited function description. Hence, we suggest performing the use case technics without relying on assumption that upper level functions are the use cases. The purpose is to reach good use cases with "complete" property, i.e. a use case shall be executed up to its end to fully address the functionality it describes and reach a new stable state or error state of the system.

The second point of discussion concerns the use of actors instead of real external systems in the black-box operational scenarios. This choice is motivated by two main reasons. First, an actor is a role that can be played by any kind of real external systems but irrespectively of the real interfaces: it let the problem space open, reusable, and abstract, especially in avionics domain when operating context is often complex (ARINC and AFDX network with redundancy for instance). However, this does not mean that no operating context is needed. This is the point for the second reason: actors can be allocated to real external systems present in the operating context. In that way, the generated actors' behaviors (UML Activity) could be automatically allocated to one or several different real external systems for simulation.

The third point concerns the synthesis of activity diagram from sequence diagrams. While several MBSE approaches [9, 11–13] suggest creating definition diagrams like SysML A*ctivity diagram* or *EFFBD* (Enhanced Function Flow Block Diagram) to combine the different UC behaviors. We consider that the sequence diagrams define user stories that are also acceptance test cases to validate the system. Hence, sequence diagrams and synthetized activity diagrams have not the same value. The first one is validated by customers, who are non-model experts, and is a usage diagram, while the second one is a definition diagram that enables refinement and design.

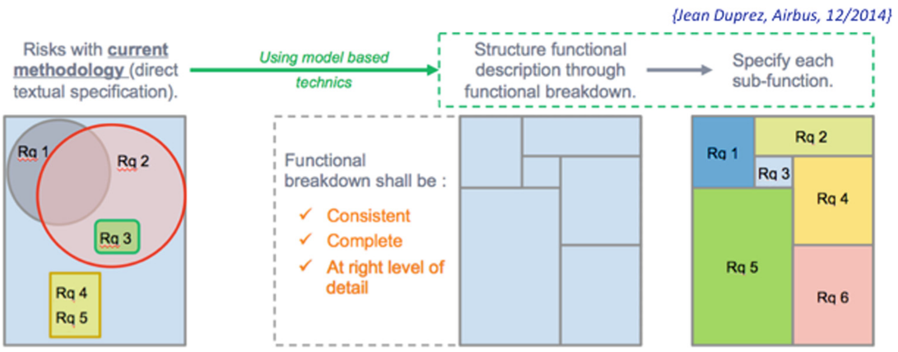

{Jean Duprez, Airbus, 12/2014}
Risks with current methodology (direct textual specification).
Using model based technics
Structure functional description through functional breakdown.
Specify each sub-function.
Rq 1
Rq 2
Rq 3
Rq 4
Rq 5
Functional breakdown shall be :
✓ Consistent
✓ Complete
✓ At right level of detail
Rq 1
Rq 2
Rq 3
Rq 4
Rq 5
Rq 6

Fig. 5. Good quality requirements from structured functional definition

Finally, the end of Sect. 2 just touches upon requirements derivation process. The derivation process creates requirements from top-level functions definitions (inputs and outputs, trigger event, conditions of activation) all along functional breakdown hierarchy. As depicted by the Fig. 5, one can derive a set of good-quality system requirements from the validated functional definition, what is not guaranteed by document-centric approaches. This could be done using generation patterns and template. The functional hierarchy leads to requirements decomposition, while the different kind of functions (interface, internal, top-level) lead to different requirements. We have done a proof of concept, but this is mainly part of further work.

5 Related Work

Use case driven development and use of Sequence Diagrams are common practices. Yue et al. [14–16] propose a set of rules and use case templates to reduce ambiguities and generate Activity Diagrams and State-Machine. The rules are applied on the use of Natural Language and enforce the use of specific keywords to document a use case, i.e. its precondition, dependency, basic flow steps, alternative flow. Song et al. [17] provide a method to create Sequence Diagrams and check their consistency with regards to the corresponding use case and class diagram. While those papers deal with scenario consistency, they do not address the concept of function and generation of functional chains in a Systems Engineering context.

Function modelling within Sequence Diagrams has been addressed in several published researches. The SysCARS approach [18] proposed to refine system scenarios by adding functions modeled as SysML operations attached to the System of Interest lifeline. Felician et al. introduced Enhanced Sequence Diagram (ESD) [19] for functions modelling. It proposed to augment lifelines with multiple “flowlines” representing Energy, Material, and Information. In that way the ESD offers a mapping between different flows and message nature to functions.

While agile method apply to Systems Engineering has been addressed in [2, 20] from the conceptual point view, our paper deals with real industrial application and experiments with an integrated team project. We provide a framework and tooling support to validate and to deliverer frequently value in a model-based approach. In [21], the authors proposed to use Object Process Methodology (OPM) in an incremental system development. However, the approach is demonstrated on a hypothetical system, and does not make any link with continuous delivery, customer in the loop or continuous validation.

6 Conclusion

We have presented motivations, principles, recommended activities, and requirements for an MBSE framework that supports agile functional definition with consistency by construction and validation through simulation. In addition, conducted experiments on an avionics subsystem has demonstrated that the proposed MBSE framework can be used in an incremental and iterative manner. The proposed activities help to detect, and fix uncomplete or wrong functional definition (missing function, functional flow, missing or wrong behavior…) and they can be conducted at any time.

In a very near future, we plan to assess the agile property of the approach during functional allocation on technical components. Teamwork and continuous validation should be assessed during technical architecture tradeoff. We also plan to conduct experiments on products identification from model. Those products can be reused in the physical architecture. Finally, we have started requirements generation from models. This ongoing work would provide consistent by construction system requirements for suppliers or domain experts.

References

1. Cloutier, R., Bone, M.: MBSE survey: initial report results, INCOSE IW, Los Angeles, USA, January 2015
2. Douglass, B.P.: Agile Systems Engineering, San Francisco, CA, USA (2015)
3. Rosser, L., Marbach, P., Lempia, D., Osvalds, G.: Systems engineering for software intensive projects using agile methods. In: International Council on Systems Engineering IS2014, Las Vegas, USA (2014)
4. Schindel, B., Dove, R.: Introduction to the agile systems engineering life cycle MBSE pattern. In: INCOSE International Symposium, 2016
5. Society of Automotive Engineers (SAE): Guidelines for Development of Civil Aircraft and Systems - ARP4754A (2010)

6. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann (2014). ISBN 978-0-12-800202-5
7. ISO/IEC 15288: Systems Engineering-System Life Cycle Processes. ISO 2015
8. Adolph, S., Cockburn, A., Bramble, P.: Patterns for Effective Use Cases. Addison-Wesley Longman Publishing Co., Inc., Boston (2002)
9. Weilkiens, T.: SYSMOD-The Systems Modeling Toolbox (20160
10. OMG: Object Management Group Foundational Subset for Executable UML Models Specification (2017). https://www.omg.org/spec/FUML/1.3/
11. Vitech: STRATA Methodology – One Model, Many Interest, Many Views. http://www.vitechcorp.com/resources/white_papers/onemodel.pdf
12. Hoffmann, H.: Harmony SE A SysML Based Systems Engineering Process (2008)
13. Pearce, P., Hause, M.: OOSEM and model-based submarine design. In: SETE/APCOSE (2012)
14. Yue, T., Briand, L., Labiche, Y.: A use case modeling approach to facilitate the transition towards analysis models: concepts and empirical evaluation. In: Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems. (MODELS), Denver USA (2009)
15. Yue, T., Briand, L., Labiche, Y.: An automated approach to transform use cases into activity; diagram. In: Proceedings of the European Conference on Modelling Foundations and Application, Paris, France (2010)
16. Yue, T., Ali, S., Briand, L., Automated transition from use cases to UML state machines to support state-based testing. In: Proceedings of the 7th European Conference on Modelling Foundations and Application, Birmingham, UK (2011)
17. Song, I.Y., Khare, R., An, Y., Hilbos, M.: A multi-level methodology for developing UML sequence diagram. In: Proceedings of the 27th International Conference on Conceptual Modeling, Barcelona, Spain (2008)
18. Piques, J.D.: SysML for embedded automotive systems: SysCARS methodology. In: Proceedings of the Embedded Real Time Software and Systems Conference, Toulouse, France (2011)
19. Campean, F., Yildirim, U., Enhanced sequence diagram for function modelling of complex systems. In: Proceedings of the 27th Complex Systems Engineering and Development, Cranfield, UK (2017)
20. Douglass, B.P.: Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development. Pearson Education (2009)
21. Mordecai, Y., Dori, D.: Agile modeling of an evolving ballistic missile defense system with object-process methodology. In: IEEE Systems Conference Proceedings. Vancouver, BC (2015)

Analyzing Awareness, Decision, and Outcome Sequences of Project Design Groups: A Platform for Instrumentation of Workshop-Based Experiments

Carl Fruehling[1] and Bryan R. Moser[2](✉)

[1] Chair of Product Development, Technical University of Munich, Munich, Germany
cfruehling@mytum.de

[2] System Design and Management, Massachusetts Institute of Technology, Cambridge, USA
bry@mit.edu

Abstract. Activity dependencies gain importance as engineering programs become more complex and global. We treat the planning of engineering programs and projects as a collaborative exercise by teams to design their shared project. An Awareness-Decision model and sensors for measurement were developed to correlate attentions and actions to outcomes during project design. A cloud-based platform allows teams to model projects and to simulate a project's cost and duration. This approach enables efficient deployment of experiments with global project design groups. During experimental sessions, we captured attention allocation, project change and performance data across 38 groups at three global sites to explore their decision-making process and exploration of the cost-duration-tradespace. In this set of experiments, the groups that were stimulated to pay more attention to dependencies did not show a correlation with improved outcomes. Employing several sequence analyses, including return time distribution, proximity walk, element focus, and vision distribution, we attempted to draw insights on awareness and attention to dependence and overall outcomes. Based on our analysis results, we give recommendations for research to more completely expose the role of attention to dependence during project design.

Keywords: Project Design · Complexity management · Dependencies
Situational awareness · Attention allocation · Decision-making
Project management, team performance · Workshop-based experiments

1 Introduction

Large-scale engineering programs and projects face increasing complexity in coordinating activities as their teams are globally dispersed. The dependencies between activities cause coordination efforts which are not taken into account in classical project planning tools. *Project Design* is a method which allows to model the architecture of complex projects [1]. The agent-based simulation software *TeamPort* forecasts the project's cost and duration including the dynamics driven by coordination efforts [2].

A project's team leaders come together in a workshop at the project's front-end to design the project architecture and model its activity dependencies. They rely on known theories and their experience from previous projects. When modelling dependencies, they face the Rumsfeld-Dilemma of known unknowns and unknown knowns [3]. While known unknowns can be simulated with the help of probability functions, unknown knowns need to be discovered by social interactions and experiments. These so-called *blind spots* are often not visible in the first place but occur to be obvious when reflecting the project after it has been completed [4]. Common examples are large public projects like building a shopping mall or an airport where cost and duration suddenly explode due to unseen conditions. Afterwards, the reasons for failure appear to be simple and one does wonder why they have not been discovered earlier.

To unveil these blind spots, a Project Design group needs to *become aware* of the causes and effects of critical activity dependencies in order to establish interactions between those teams who are assigned to the dependent activities [5]. When modelling the dependencies in TeamPort correctly (with respect to environmental system boundaries), the group can decrease both the cost and duration of a given project model. This ability is defined as the subject's Project Design *performance* (outcome). In our research, we relate the three levels of awareness, decision, and outcome with each other. The detailed research framework and methodology is explained in the following.

2 Research Framework

To research the group behavior and decision-making during a predefined Project Design exercise, we formulated the Awareness-Decision-Model (Fig. 1). The model consists of a three-step cycle process. First, a subject becomes aware of a certain project element. This *situational awareness* implies a sufficient amount of concentration and cannot be intuitive. As described by Endsley [6], the subject first perceives the element, then comprehends the element's conditions and relations within the model. Finally, the subject projects its comprehension and makes a decision. The decision on making a change in the project model leads to an outcome, after the subject simulates the new status of the model. The achieved performance is fed back to the subject. Now the subject perceives the new situation and initiates a new iteration in its awareness-decision process.

In order to reframe the situation the subject needs to activate its "System 2" [7]. From our perspective, this is achieved by applying a structuring method in the Project Design process. This process is simulated in a *workshop-based experiment* where the subjects work in teams of multiple persons on a project model in the simulation software TeamPort. A subject gets a "scrambled" project model. The objective is to minimize the forecasted project time and cost by making changes in the project model (e.g. team size, resources per activity, dependencies etc.). The lower the forecasted time and cost, the higher the subject performance is ranked. Before making changes, the subject needs to lay out the project architecture by dragging and dropping elements

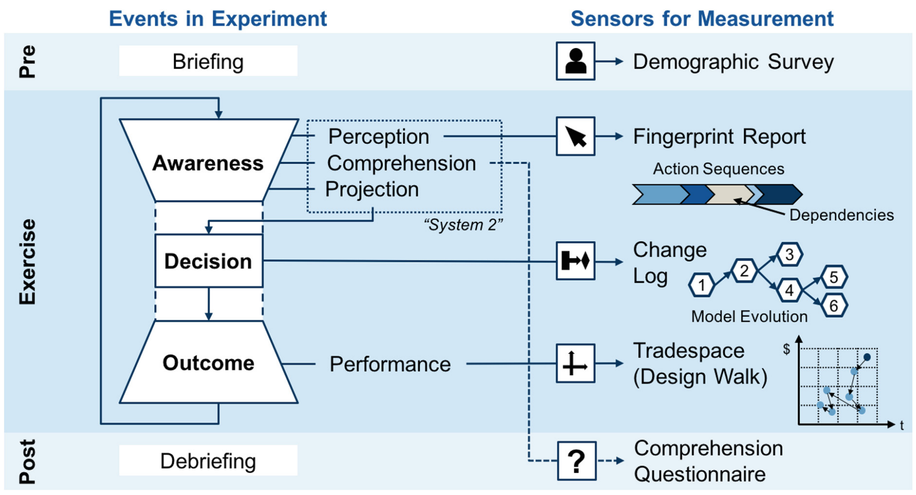


Fig. 1. Awareness-decision-model and experiment sensors

apart and clearing up the structure. This structuring method is understood as *treatment* in the following. A control group would not conduct the layout exercise but receive a ready structured project model from the beginning on. **Two research questions emerge:** (1) Does a *structuring method* increase a group's *awareness* for activity dependencies? (2) Does the *awareness* for activity dependencies increase a group's *performance*?

To answer these questions, we designed an experiment in a workshop setup for Project Design groups. We used TeamPort as platform and orientated the design of experiment on an approach which was previously defined by Moser et al. [1]. Later we elaborated the approach by implementing the learnings of Chucholowski et al. [8].

In our workshop-based experiment, the participants were assigned to either a control group or a treatment group. Each Project Design group consisted of four participants. Social interactions between the group members fostered the participants' learning. On the one hand, they could discuss and reflect on their modeling approach. On the other hand, they could support each other on technical questions. After a briefing and tutorial phase which explained the software controls, all groups worked on the same predefined project model and tried to reduce its cost and duration. The treatment groups began with the layout structuring exercise, while the control group started with an already prepared clear structure. The exercise lasted 1.5 h. After completing the exercise, the participants were debriefed. The groups compared their modeling results. Each participant filled out a survey regarding their Project Design approach. The feedback from the participants given in the surveys on the workshop-based experiment indicates a high benefit from the interactive way of solving a problem with the simulation software using an intuitive graphical interface as well as the social experience of working in group and developing solution strategies together. The feedback was also used to collect improvement suggestions for further research.

Based on this design of experiment, we formulated the following **five hypotheses**: (1) High performing Project Design groups allocate their attention differently than low performing Project Design groups. (2) High performing Project Design groups allocate their attention more to activities and dependencies than do low performing groups. (3) High performing Project Design groups focus on the project architecture before making changes on the project model. (4) Project Design groups increase awareness of activity dependencies through laying out the project architecture themselves. (5) High performing Project Design groups follow similar action patterns which low performing Project Design groups do not follow.

To test these hypotheses, we implemented several sensors to the software and collected data about the groups' attention allocation, their decisions, and their performance in real-time during the experiment (Fig. 1). First, the demographic data of the participants was collected. Second, the *Fingerprint Report* protocoled all mouse clicks on the graphical user interface of the software. As a result, the action sequences show on which elements and for how long the groups focused on. Third, the *Change Log* protocoled all changes a group made between to simulation runs. This allows a comparison between the attention allocation of a group and the actual changes that the group performed on the project model. We can thus capture behavioral patterns in project design. Fourth, *Design Walk* collected the performance of each group in reducing the cost and duration of the project. Each time a simulation is ran, a data point is added to the *Tradespace* (a plot with two axes for cost and duration). Finally, the *Comprehension Questionnaire* asked the participants how they did approach the exercise and what they would do differently next time. It also allowed them to reflect their group work and give feedback on the design of experiment. The collected data was aggregated and analyzed to identify patterns of success-promising Project Design strategies.

The analysis methods are explained in Fig. 2.

3 Clustering Analysis

When analyzing the attention allocation data of a subject, not only the absolute number of perceptions (mouse clicks) must be taken into account, but the relative order of occurrences as well. We used Bioinformatic algorithms to perform a sequence analysis. In Bioinformatics, sequences of genomes are aligned with each other. After counting the number of matches between two sequences, their similarity is accessed. Comparing all sequences from one data set allows to draw a phylogenetic tree which shows the relationship between the genomes. Looking at two genomes, the less branches lie between them, the closer they are related. In our sociotechnical research, a sequence alignment would not work as it would imply that similar behaving groups would click on the same elements at the same time. Instead, we use an alignment-free approach and define features which are frequency-based. For each feature, we first generate a first distance matrix for all groups (G) and a second distance matrix for each included sequence (S). Next, we cluster the matrices hierarchically and compare the clusterings of predictive and outcome features. After accessing the degree of similarity, we

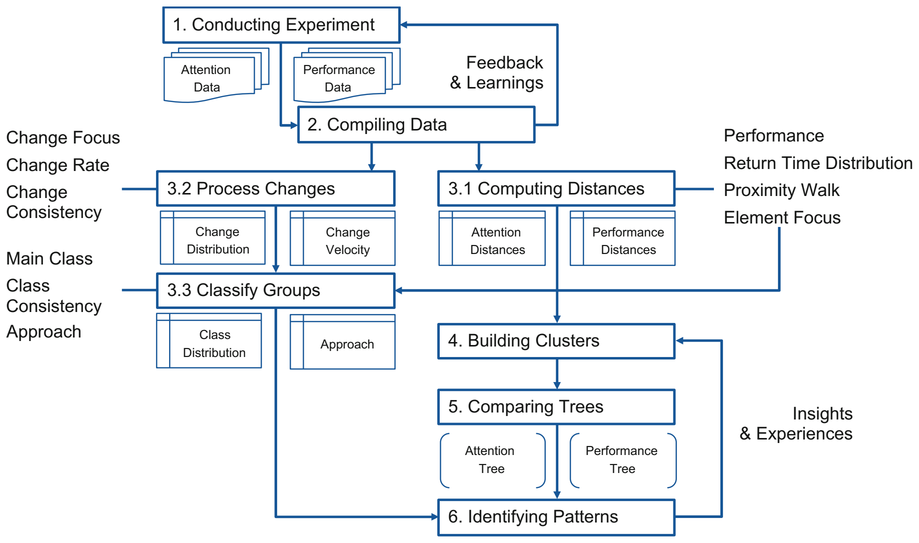


Fig. 2. Steps in the analysis method

determine p-values with respect to a null hypothesis. This allows us to identify performance-related features, validate our hypotheses and draw conclusions.

We defined four features for the clustering analysis. The Performance Impact represents the outcome variable, while the Return Time Distribution, the Proximity Walk and the Element Focus represent the predictive variables. The following firstly describes the defined features. Secondly, it interprets the achieved results.

The **Performance Impact** (PI) of a sequence is defined as achieved performance compared to a baseline. The PI_{tot} refers to the project's initial status and reflects the total performance. The PI_{inc} refers to the incremental change compared to the last simulation run. We combine the measures PI_{tot} and PI_{inc} of the *ith* and *jth* sequence in distance matrix $D^{PI} = \left[d_{ij}^{PI}\right]$. Each pairwise distance is calculated with

$$d_{ij}^{PI} = \sqrt{\left(PI_{itot} - PI_{jtot}\right)^2 + \left(PI_{iinc} - PI_{jinc}\right)^2} \tag{1}$$

Next, we calculate the **Return Time Distribution** (RTD). This feature is determined by the frequency of elements which appear in sequence until the same element appears again [9]. In our case, we take the number of clicks on other element types, until an element of the same type is clicked again. The smaller this number (the Return Time) is, the higher is a subject's attention allocation towards that element. As a result, we get the frequency of each Return Time for each element. The means and standard deviations of these frequencies give the RTD. The pairwise distances of this feature ($D^{RTD} = \left[d_{ij}^{RTD}\right]$) are calculated with

$$d_{ij}^{RTD} = \sqrt{\sum\nolimits_{r=A}^{X}\left(\mu_{ir}^{RTD} - \mu_{jr}^{RTD}\right)^2 + \sum\nolimits_{r=A}^{X}\left(\sigma_{ir}^{RTD} - \sigma_{jr}^{RTD}\right)^2} \tag{2}$$

where

$$\mu_r^{RTD} = \frac{\sum(Return\,Time \times Frequency)}{\sum Frequency} \tag{3}$$

$$\sigma_r^{RTD} = \sqrt{\frac{\Sigma(Frequency - \mu_r^{RTD})^2}{\Sigma Frequency}} \tag{4}$$

$$\forall r \in \{A, B, C, D, E, F, G, H, X\} \tag{5}$$

The index r represents the element type. In TeamPort, we measure eight different element clicks ($A - H$) plus the click for a simulation run (X). The element types are separated into objects (products, teams, activities, and phases), relations between objects (dependencies and contracts), as well as locations and projects.

Further, we examine the groups' **Proximity Walk** (PW). The PW is defined by the structural distances a subject covers between two consecutive clicks. The structural distance is the shortest path on connections between to objects. Each click is given a Proximity distance with respect to the previous click. The frequency distribution of all Proximity distances gives the PW. Two different frequency profiles exist. Either, A subject clicks incrementally from element to element what results in a short PW. Or, it jumps forth and back between different segments of the project what results in a long PW. We use the means (μ^{PW}) and standard deviations (σ^{PW}) to compute the distance matrix $D^{Prox} = \left[d_{ij}^{Prox}\right]$. The pairwise distances are calculated with

$$d_{ij}^{PW} = \sqrt{\left(\mu_i^{PW} - \mu_j^{PW}\right)^2 + \left(\sigma_i^{PW} - \sigma_j^{PW}\right)^2} \tag{6}$$

Next, we determine the **Element Focus** (EF) of each subject. This feature is defined by the ratio (R_{ET}) of number of clicks on one element types (ET) to number of clicks in total.

$$R_{ET} = \frac{clicks_{ET}}{clicks_{total}} \tag{7}$$

We calculate the ratios for eight element types ($A - H$). The pairwise distances ($D^{EF} = \left[d_{ij}^{EF}\right]$) are calculated with

$$d_{ij}^{EF} = \sqrt{\sum_{r=A}^{H}\left(R_{ir} - R_{jr}\right)^2} \tag{8}$$

$$\forall r \in \{A, B, C, D, E, F, G, H\} \tag{9}$$

Finally, we analyzed the **View Distribution** (VD). This feature is analogically calculated to the EF. Table 1 gives an overview of all defined features for the clustering analysis. It shows which statistical sub-features have been applied for calculating the pairwise distances.

Table 1. Clustering features overview

Var.	Features	G	S	Mean	SD	Sum	Max.	Ratio
out.	Performance Impact	✓	✓	✓	✓	–	✓	✓
pre.	Return Time Distribution	✓	✓	✓	✓	–	–	–
pre.	Proximity Walk	✓	✓	✓	✓	✓	–	–
pre.	Element Focus	✓	✓	✗	✗	✓	✗	✓
pre.	View Distribution	✓	✓	✗	✗	✓	✗	✓

G = group-based, S = sequence-based,
out. = outcome variable, pre. = predictive variable
✓ Applied – Not applied
✗ Not applicable

Each distance matrix was normalized to its highest value. This allows to directly compare them with each other. Next, all distance matrices were clustered hierarchically with the Neighbor Joining algorithm. The Neighbor Joining algorithm minimizes the sum of branch lengths for each node and results in an unrooted tree. The method is computational more efficient than other clustering algorithms [10].

In total, we examined the features of 38 groups (two had to be excluded as their data was insufficiently recorded). We gained 303 sequences for our analysis. The PIs are plotted in Fig. 3. The four quadrants of the scatter plot represent the four possible outcomes of a sequence. If both PI_{tot} and PI_{inc} are positive, the group is performing well. If PI_{tot} is positive but PI_{inc} is negative, the group's performance is still better than the initial status but is moving in the wrong direction. It should consider stepping back to its last project model version. If PI_{tot} is negative but PI_{inc} is positive, the group moves in the right direction, but should consider loading a different project model version, as its total performance is still negative. If both PI_{tot} and PI_{inc} are negative, the group is completely on the wrong track as it is declining project performance in both ways.

In our data, we can see that most groups could make a positive incremental as well as total impact on the project performance. However, the treatment groups (those who used the structuring method) underperformed the control groups on average. Therefore, the treatment was not effective.

Nevertheless, we applied the clustering algorithm and produced a PI clustering (not shown here). The PI clustering is very distinct as it has multiple large and small clusters. To draw a conclusion, we need to analyze the predictive features. For that purpose, we show a cell plot of the attention allocation over the exercise time initiated

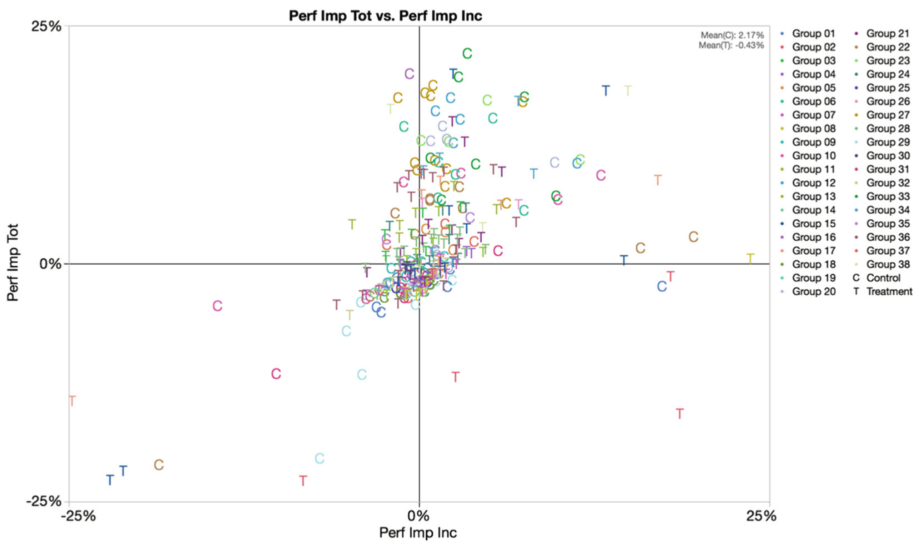


Fig. 3. Total versus incremental performance impact per simulation run

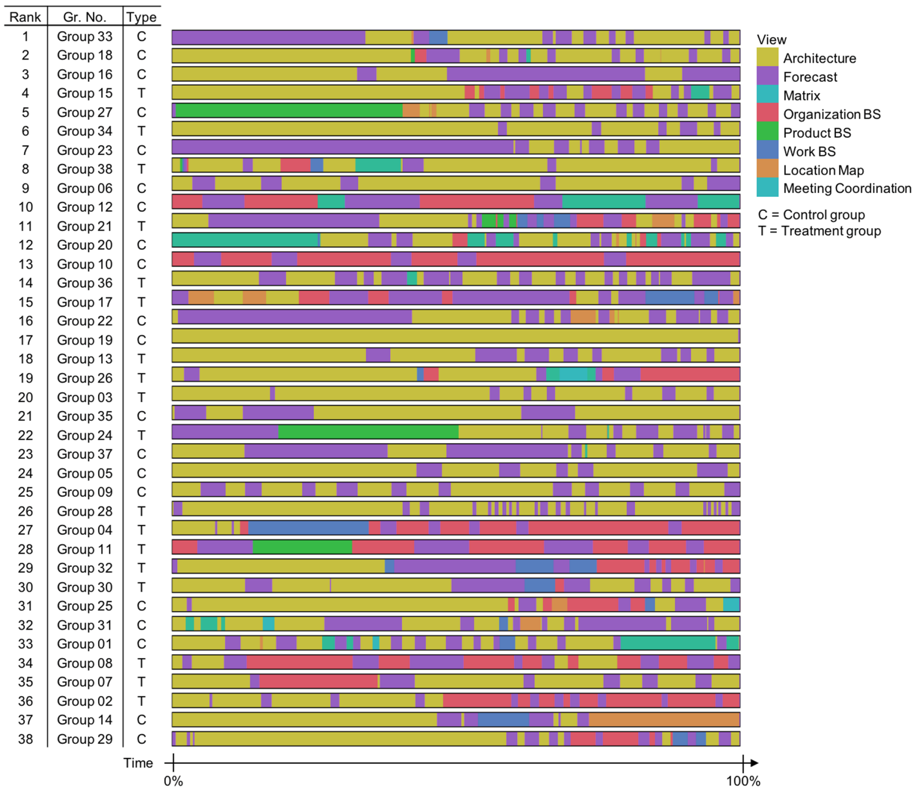
by mouse clicks (Fig. 4). The length of each bar in the cell plot indicates the relative time spent in one view mode. This data is gained from the Fingerprint Report as explained above. It shows the participants' mouse clicks and the mode in which they view the project model.

The *Architecture* view allows to sketch the project structure. The *Forecast* window is used to trace back the reasons for cost and duration effects. The *Matrix* view shows relations between objects in a sorted table. Additionally, the project model can be viewed in three different *Breakdown Structures* (*BS*). The *Location Map* gives an overview about the project teams' locations and the environmental boundaries. The *Meeting Coordination* tab allows to schedule meetings for the project teams. By rowing the view modes of each group together, their action sequences are gained.

The groups' action sequences have been ranked for their maximum performance. A qualitative analysis of the cell plot gives only little information. It seems that high performing groups first plan their approach before starting to work on the model. Low performing groups seem to use a trial-and-error approach instead. Additional analyses have shown the effectiveness of the treatment. If many clicks occur in a short time during the first part of an action sequence, the group clicked on many elements while it was laying out the project model.

4 Results Interpretation

In the next step, we derive the predictive clusterings. Figure 5 shows the RTD clustering which consists of a few large clusters and many small clusters. The other clusterings are not shown here. The distance between two elements is calculated as RTD (fingerprint based). Nodes bundle elements with short distance to each other.



Rank	Gr. No.	Type
1	Group 33	C
2	Group 18	C
3	Group 16	C
4	Group 15	T
5	Group 27	C
6	Group 34	T
7	Group 23	C
8	Group 38	T
9	Group 06	C
10	Group 12	C
11	Group 21	T
12	Group 20	C
13	Group 10	C
14	Group 36	T
15	Group 17	T
16	Group 22	C
17	Group 19	C
18	Group 13	T
19	Group 26	T
20	Group 03	T
21	Group 35	C
22	Group 24	T
23	Group 37	C
24	Group 05	C
25	Group 09	C
26	Group 28	T
27	Group 04	T
28	Group 11	T
29	Group 32	T
30	Group 30	T
31	Group 25	C
32	Group 31	C
33	Group 01	C
34	Group 08	T
35	Group 07	T
36	Group 02	T
37	Group 14	C
38	Group 29	C

Fig. 4. The attention allocation cell plot over exercise time

Clusters can be defined by “cutting” the tree at a certain diameter. In other words, one node can represent a cluster. The node itself has no meaning. There are large and small clusters. The farther a knot is away from the middle, the smaller a cluster is. Large clusters are a sign of typical behavior shared with many subjects. Small clusters are “exotic” behaviors only few subjects showed.

The idea of clustering visualization is to identify correlations between features by layering their clusterings over each other. However, this qualitative comparison is only practicable for a small sample. On the other hand, a large sample size gives a predictability. A quantitative comparison method is needed to analyze the correlation of different clusterings. Therefore, we used the Fowlkes-Mallows-Index (FMI) to compare the clusterings and determine their degree of similarity. The FMI takes values between 0 and 1. At 1, the examined clusterings are identical. At 0, they are not related at all. Both clusterings must consist of the same number of groups (n). The number of clusters (k) takes all integer number from 2 to $n - 1$. Monte Carlo samplings show that the FMI converts against 0 for higher number of clusters what differentiates it from other similarity indices [11].

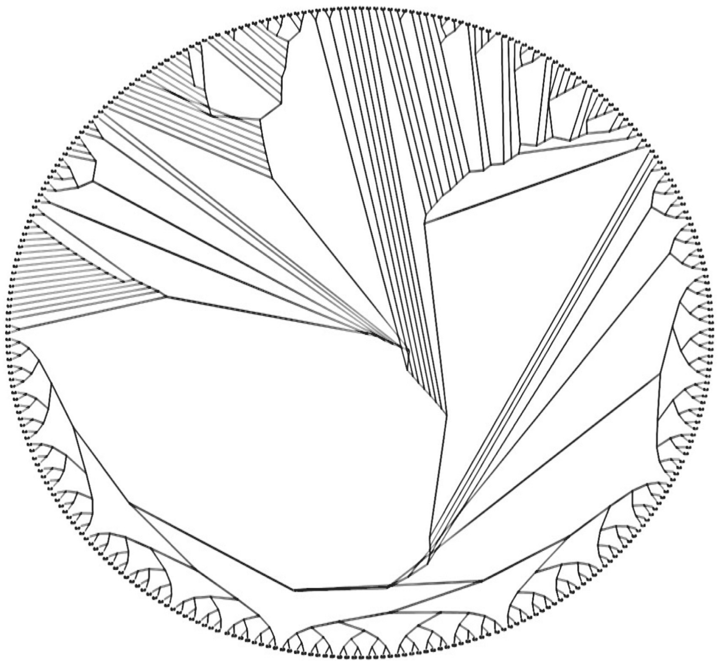


Fig. 5. Clustering example for the return time distribution

The FMI needs to be contrasted by its null hypothesis to determine the p-Value of the resulting values. In our case, we used a bivariate normal sample of 20 pairs with the null hypothesis values

$$n = 303, \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1}$$

The sample size equals the size of data set. Figure 6 shows the FMI Distribution of three clustering comparisons (PI with RTD, PI with PW, and PI with EF) as well as the null case.

If the line lies above the dashed null case line, the degree of similarity is higher than random. The analysis shows that the FMI of PI with RTD is lower than null for cluster numbers higher than 39. The same applies for the FMI of PI with PW for cluster numbers higher than 26. Only the FMI of PI with EF stays over null for the entire interval. Nevertheless, we need to add a confidence level to null case (Table 2).

The calculated p-values do not satisfy a statistical threshold of 0.05. So far, we could not find a feature which would be a good predictor in our clustering analysis.

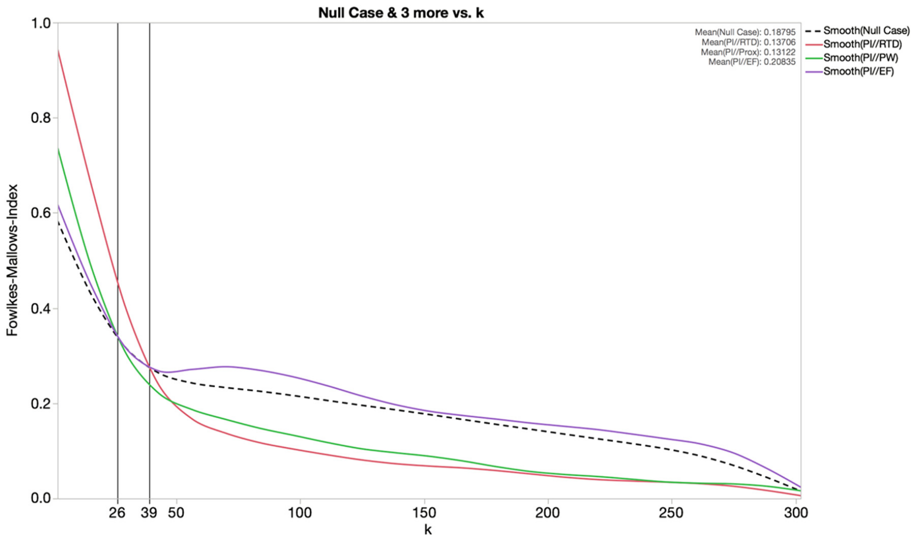


Fig. 6. The fowlkes-mallows-index distribution of performance impact and return time distribution

Table 2. The p-values for selected FMI distributions

FMI distribution	PI with RTD	PI with PW	PI with EF
Mean (p)	0.8900	0.9439	0.1142
StdDev (p)	0.3129	0.2192	0.3070

5 Conclusion

We conducted four workshop-based experiments with in total 38 Project Design groups. We measured the groups' attention allocation, decisions, and performance to identify patterns and success-promising Project Design strategies. From the hypotheses formulated in the beginning of the paper, only two could be proven as valid. The first hypothesis (high performing groups allocate their attention differently) and the second (high performing groups allocate their attention more to dependencies) indicated validity. Hypothesis 3 (High performing groups focus on the project architecture) and 4 (groups become aware of activity dependencies through laying out the project architecture themselves) could not be proven as valid. The last hypothesis (High performing groups follow similar action patterns) is only been proven to a certain degree. Preliminary results indicate that high iteration speed combined with a change-observe approach (rather than trial and error) can lead to higher success.

In summary, project design groups can benefit from using planning software that visualizes the project architecture and highlights dependencies between project activities. To gain performance improvements a project design group should first plan their approach on how the handle the different activity dependencies before starting to revise the project architecture. After increasing their awareness for activity dependencies, project design groups should use a respective approach for making decision in the project design phase. Decision-making data was collected throughout the experiments. This data has not been analyzed, yet. Next, we would like to add different clustering techniques to our analysis methods to test the recorded data for more features.

The method proposed in this paper is a first step in various directions of our research strategy. First, it proposes activity dependencies more richly characterized than precedence as a relevant research topic. Complex dependencies are a crucial aspect in complex programs and can have a big impact on the performance. By modelling dependencies explicitly and being able to capture how teams are aware and interact with them, we bring the topic closer to the practitioners and give them a way to evaluate and steer the behavior of the team. Second, the research platform and approach we propose opens possibilities of fast, global experiments in real time.

The instrumentation of the decision-making process, together with the algorithms and analysis techniques proposed, enable teams to gain more insights on their behavior. Teams can recognize best practices and adapt their collaboration accordingly. With a globally deployable approach for the proposed workshop-based experiments, large amounts of data would be gathered. Analyses techniques like machine learning could be used to process the resulting data and build a prediction model for project design practices. Further research should leverage the gained insights on how to steer projects in the right direction for the design of new projects.

Acknowledgements. The authors would like the thank the contributions of the Global Teamwork Lab (GTL) at MIT and University of Tokyo, as well as Dr. Eric Rebentisch of CEPE at MIT and Nepomuk Chucholowski of Technical University of Munich.

References

1. Moser, R., Grossmann, W., Starke, P.: Mechanism of dependence in engineering projects as sociotechnical systems. Presented at the 22nd ISPE Concurrent Engineering Conference (CE 2015) (2015)
2. Moser, B.R., Wood, R.T.: Design of complex programs as sociotechnical systems. In: Stjepandić, J. (ed.) Concurrent Engineering in the 21st Century, pp. 197–220. Springer International Publishing, Switzerland (2015)
3. Rumsfeld, D.H.: DoD News Briefing. Federal News Service Inc., Washington, D.C. (2002)
4. Luft, J., Ingham, H.: The Johari Window: a graphic model of awareness in interpersonal relations, Human relations training news, vol. 5, pp. 6–7 (1961)
5. Marle, F., Vidal, L.-A.: Managing Complex, High Risk Projects - A Guide to Basic and Advanced Project Management. Springer-Verlag, London (2016)
6. Endsley, M.R.: Toward a theory of situational awareness in dynamic systems. Hum. Factors **37**, 32–64 (1995)
7. Kahneman, D.: Thinking, Fast and Slow, 1st edn. Farrar, Straus and Giroux, New York (2011)

8. Chucholowski, N., Starke, P., Moser, B.R., Rebentisch, E., Lindemann, U.: Characterizing and measuring activity dependence in engineering projects. Presented at the Portland international conference on management of engineering and technology 2016, Honolulu, Hawaii, USA (2016)
9. Kolekar, P., Kale, M., Kulkarni-Kale, U.: Alignment-free distance measure based on return time distribution for sequence analysis: applications to clustering, molecular phylogeny and subtyping. Mol. Phylogenet. Evol. **65**, 510–522 (2012)
10. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
11. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. **78**, 553–569 (1983)

Systemic Design Engineering

Curriculum and Instructional Results

Jon Wade[1(✉)], Steven Hoffenson[1], and Hortense Gerardo[2]

[1] Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030, USA
{jon.wade, steven.hoffenson}@stevens.edu
[2] Lasell College, 1844 Commonwealth Avenue, Auburndale, MA 02466, USA
hgerardo@lasell.edu

Abstract. This paper describes a methodology, Systemic Design Engineering, that integrates systems thinking for sustainability, design thinking for human centricity, and systems and software engineering for implementation efficiency. Based on these operating principles, an integrated set of learning objectives for this approach are described. A supporting syllabus is presented which is suitable to a semester-long undergraduate or graduate design course. A pilot of this curriculum in a graduate level systems and software engineering design course has been conducted. The learning results from this pilot are presented. These results are compared and contrasted with the results from a similar, traditional systems engineering design course and a dedicated systems thinking course. Finally, conclusions and areas for further refinement and developments are described.

1 Background

The Concept Phase of Systems Engineering, which is often referred to as "preliminary design" in other engineering disciplines, is most critical for it determines most of the cost, complexity and value of a system. Figure 1 illustrates how the committed costs and the costs to remediate defects increase as the design process progresses over time [1].

The percentages along the timeline represent the actual life cycle cost (LCC) accrued over time based on a statistical analysis performed on projects in the US Department of Defense (DoD), as reported by the Defense Acquisition University (1993). However, perhaps due to the traditional role of systems engineering in the contract-driven defense/aerospace industry, systems engineers are often thought of as starting with a set of high-level customer requirements and developing the architecture, design and implementation through integration, often handing off the program to manufacturing and logistics. In addition, as systems continue to become more human-centric, it is critical that these aspects are also considered throughout the lifecycle in such a manner that the systems are sustainable.

Given the interdisciplinary nature of conceptual and detailed design, it is no wonder that a wide variety of methods, processes and approaches have been developed to aid in

The original version of this chapter was revised: Third author name has been corrected. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-04209-7_29

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 192–202, 2019.
https://doi.org/10.1007/978-3-030-04209-7_16

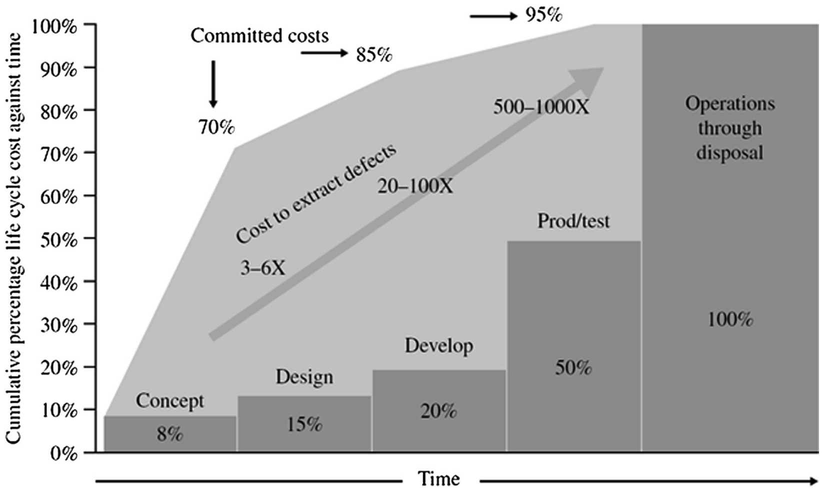


Fig. 1. Committed Life Cycle Cost against Time. 3.1. Fort Belvoir, VA: Defense Acquisition University. (Source: DAU, 1993)

addressing design problems [2–10]. Even within the engineering community, designers from different disciplines working on different types of problems have devised their own frameworks to demystify the design process. As academia and industry are moving towards becoming transdisciplinary, it is important to find ways to train students with knowledge from various disciplines. Part of the challenge is to identify where these disciplines overlap and find a common language to communicate among practitioners with different training. It no longer makes sense to have separate design processes for different disciplines that seek to solve technical problems, and there is a need to create a codified standard to optimally equip students who will be designing the products, services, and systems of the future [11].

The objectives for this interdisciplinary approach are [11]:

- To create a methodology that is logical, easy to follow, and can be taught as part of any curriculum that teaches approaches to problem-solving, complex systems, and product development.
- To create a process that promotes exploration of innovative ideas and a systematic way to select the most promising option (engineering design).
- To promote a process that results in a low-risk, realizable, and profitable solution (systems engineering).
- To create resulting products that satisfy a real user need (design thinking).
- To demonstrate that resulting products are sustainable in the changing and increasingly connected marketplace (systems thinking).
- To verify that the approach can be used for systems which are humancentric, including complex social systems (systemic design).
- To establish a framework that can be tailored to be used on systems and products of all types (agile systems/software engineering).

2 Systemic Design Engineering

Systemic Design Engineering (SDE) [11] was developed to satisfy the aforementioned objectives. This approach combines the most advantageous elements of: (1) systems thinking to ensure that the resulting design is appropriate to its context and is sustainable, (2) design thinking to ensure that the design properly considers human centricity, and (3) systems and software engineering to ensure that the design can be implemented, deployed and supported at the required performance, cost and quality.

One commercial example of success in each of these areas is the iPhone. In this case, Apple has enabled an ecosystem of application developers such that they are not direct employees, but rather are the emergent result of a system that was established by Apple. This ecosystem is such a formidable barrier to entry that only two have been successfully established, those for the IOS and Android. Other phone makers, such as the pioneer Nokia, are no longer in the market as their businesses were not sustainable without the existence of such an ecosystem. Hence, the critical importance of systems thinking. Design thinking is a critical partner to systems thinking, as it ensures that the product properly considers the human-centric nature of systems and their interactions with people. In the smart phone example, the iPhone utilized a successful human interface that has resulted in many consumers staying with the iPhone despite the sometimes significant advantages of competitors' hardware platforms. Learning a new user interface is simply too great of a hurdle for many, even if the competing interfaces apply many of the same interface principles (e.g., swipes and other gestural motions). In addition, understanding the ethnographic and cultural contexts of the users can be critical to the success of the products. Finally, systems and software engineering are critical to enable successful engineering of the product such that it can be rapidly ramped up to tens of millions of units per month all while achieving the necessary levels of cost and quality.

The following is a brief description from [11] of how SDE is used which provides context for the development of the course curricula that was developed for the instruction of conceptual design. As noted earlier, it is in the front-end conceptualization and design of the system where design and systems thinking can make the greatest contributions. Using this approach, the Cynefin system taxonomy [12], shown in Fig. 2, is used to first identify and classify different types of systems.

As noted in this diagram, simple systems can be addressed by best practices using an approach where one senses the system, categorizes it, and then responds with the best practice that is applicable for that situation. Those skilled in bureaucratic are most comfortable with this approach. Complicated systems are ones in which cause and effect are well-understood, and the most appropriate approach is to sense the system to collect data, perform analysis, and then make decisions based on this analysis. This is the system type with which engineers are most comfortable with the result that all systems are seen to be complicated. Chaotic systems require immediate action, and as a result, there is no time for data collection and analysis. Thus, the appropriate response is one of act, sense the results of the action, and then base further responses on these results. Authoritarian are most comfortable with these types of systems with the result

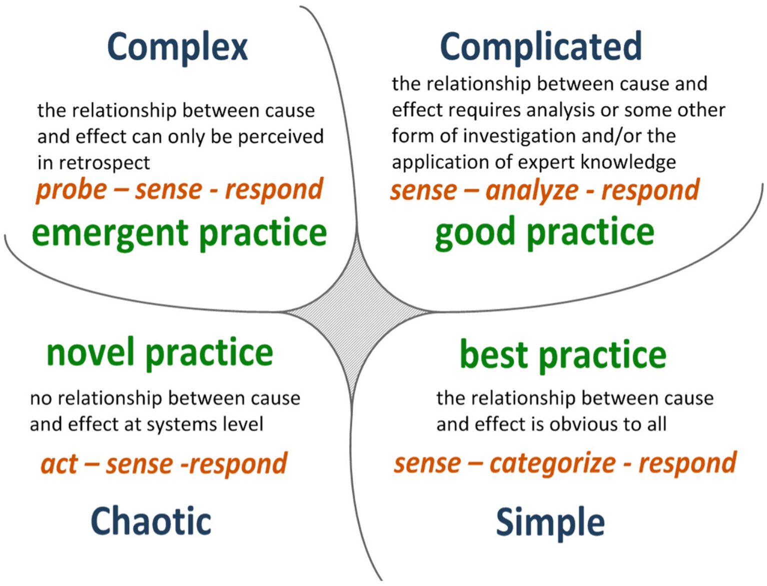

Complex
the relationship between cause and effect can only be perceived in retrospect
probe – sense - respond
emergent practice
Complicated
the relationship between cause and effect requires analysis or some other form of investigation and/or the application of expert knowledge
sense – analyze - respond
good practice
novel practice
no relationship between cause and effect at systems level
act – sense -respond
Chaotic
best practice
the relationship between cause and effect is obvious to all
sense – categorize - respond
Simple

Fig. 2. Cynefin Framework (source: Kurtz, 2003 [12])

that the firefighters might be the ones who start the fires through the lack of proactive measures.

Simple systems are rare and generally do not require the attention of engineering. Chaotic systems require rapid response and thus are not appropriate for engineering responses. While complicated systems are already addressed well by existing analytic engineering practices, this is not the case for complex systems. In addition, the importance of complex systems is growing rapidly due to the increasing predominance of humans and social elements in our socio-technical systems. In fact, with the inclusion of humans in systems, anthropological approaches resulting in an understanding of the cultural norms and "interwoven meanings and practices" of the stakeholders and system users becomes critical [11]. In complex systems, engineers need to play the role of scientists, probing the systems through experimentation probing the system and then sensing the results to begin to understand the relationships between cause and effect, which then enables an educated response. The design of experiments and use of rapid iterative prototyping are essential skills in this area, for the goal is not to 'quickly fail', but rather to quickly learn.

Thus, the combination of systems and design thinking is critical in addressing the challenges of complex systems. Systems thinking approaches are necessary to ensure that the overall context of the system is well understood so that resulting systems can evolve in an acceptable fashion and be sustainable. Many different systems thinking methods and tools can be deployed in this effort, but they will need to define the extent of the systems, its critical interconnections and interactions, the overall dynamics of the system, and the potential leverage points that might need to be used to influence the system to have the desired behavior [13]. Through system and design thinking, critical assumptions must identified, and anthropological approaches should be used to establish ground truth [14]. These can be short studies, or they can involve weeks or even months of fieldwork observation, which might be very difficult for action-oriented engineers to conduct, particularly where observation skills are critical. In these situations, knowledge in the development of design of experiments is critical, and scientific approaches of discovery are highly valued. Ideally, this is where engineers might enlist the assistance of those who are experts in the field, particularly in the social sciences.

Once the context, including the social and human factors are understood, these results can be used in models that enable the generation and selection of design concepts which can then support the more traditional activities of systems and software engineering. This will enable the development of the traditional systems and software engineering artifacts such as a concept of operations, top-level system requirements and derived requirements, both functional and non-functional requirement (e.g., performance, reliability, cost, etc.), competitive analysis, profitability and so on.

This approach, entitled Systemic Design Engineering, is a tailorable framework, extending existing engineering frameworks to incorporating the most critical elements of systems and design thinking, depending on the system type and need. Design thinking is present through the emphasis on the users and stakeholders throughout the process, particularly in the early stages, and elements of systems thinking ensure that the designers are thinking about how the product fits into the world, related markets, sustainability concerns, and synergistic opportunities. By combining the elements of the existing predominant design paradigms, the success criteria enumerated above for a unified approach to design should be satisfied by Systemic Design Engineering.

3 Course Curricula

Once this Systemic Design Engineering concept was created, it was tested in the context of a course called the Conception of Cyber-Physical Systems, which is part of a four course graduate series covering the lifecycle of conception, design, implementation and sustainment of cyber-physical systems. The core modules for the course are based on the critical aspects of systems thinking, design thinking, and systems and software engineering that are described below, along with the structure and learning objectives for each module.

3.1 Context – Systems Thinking

The learning objective for this major section is that the students understand systems concepts and are able to properly classify systems so that they can apply appropriate analysis and design approaches to them. In addition, they learn how to identify the interrelationships and structures of systems, analyze and predict system dynamics, and determine where and how to best influence and modify systems to achieve the desired outcomes. The objectives for the section modules are noted below.

3.1.1 Systems Perspectives

Objectives:

- Identify different system types: simple, complicated, complex and chaotic
- Understand the basic components of systems thinking
- Understand the basics of the soft systems methodology
- Be able to construct a root definition for a system of interest
- Be able to view a system from multiple perspectives using CATWOE.

3.1.2 Relationships

Objectives:

- Understand how systemigrams are constructed and used
- Be able to transform a root definition into a systemigram mainstay
- Be able to construct a systemigram for a system of interest
- Be able to tell a story using a systemigram.

3.1.3 Dynamics

Objectives:

- Understand the elements and operation of causal loops
- Understand the difference between reinforcing and balancing loops
- Be familiar with some causal loop archetypes
- Be able to construct a causal loop
- Be able to tell a story with causal loops.

3.1.4 Leverage Points

Objectives:

- Understand the concept of system leverage points
- Understand the various types of leverage points and how they differ
- Be able to identify leverage points in your system of interest and determine which are most appropriate to achieve your objectives.

3.2 Human Centricity – Design Thinking

The objective for this major section is for the students to understand design thinking concepts informed by anthropocentric analysis to enable informed understanding of the needs of customers and stakeholders, thereby creating a system in which a design concept can be created that meets preliminary requirements. The objectives for the section modules are noted below.

3.2.1 Design Thinking Essentials

Objectives:

- Become familiar with the elements of design thinking
- Hear the views of prominent design thinkers regarding process and objectives

- See examples of design thinking success
- Experience a one hour Ideation Process
- Learn the importance of storytelling
- Create artifacts that can be used throughout this course.

3.2.2 Identifying Opportunities

Objectives:

- Describe a framework (Real-Win-Worth it) by which to evaluate opportunities
- Describe situations that enable opportunity
- Discuss the importance and characteristics:
  - Appropriate Problem Statement
  - Market Segmentation
  - Stakeholders
  - Business Model and Scale
  - Product and Market type.

3.2.3 Identifying Customer Needs

Objectives:

- Understand the Voice of the Customer (VOC) process that can be used to elicit valid customer needs, including the means to:
  - Create a design of experiments
  - Gather raw data from customers
  - Interpret the raw data in terms of customer needs
  - Organize the need into a hierarchy of primary, secondary, and (if necessary) tertiary needs
  - Establish the relative importance of the needs
  - Reflect on the results and the process.

3.2.4 Preliminary Product Specifications

Objectives:

- stakeholder requirements categorized as "characteristics"
- Introduce one potential method to facilitate this translation: Quality Function Deployment
- Introduce the concept of acceptance criteria and emphasize their importance for maintaining project focus.

3.3 Realization – Systems and Software Engineering

The objective for this major section is that the students translate the results determined from systems and design thinking into a concept of design, concept of operations, use cases and systems requirements that can be used for subsequent engineering activities, all while satisfying the economic requirements for the systems. The objectives for the section modules are noted below.

3.3.1 Concept Design

Objectives:

- Discuss the development of system implementation concepts
- Describe how to use a Pugh Concept Selection Matrix
- Learn when rapid prototyping is appropriate
- Understand the various types of rapid prototypes
- Know how to create and execute a design of experiments
- Learn how and when to use rapid prototyping.

3.3.2 Concept of Operations

Objectives:

- Introduce the context diagram for clarifying the boundary of the system being envisioned
- Define the active stakeholders and relate them to the context diagram
- Describe and develop a Concept of Operations.

3.3.3 Use Case Scenarios

Objectives:

- Have the capability to describe the required capabilities of the system using Use Cases and Sequence Diagrams.

3.3.4 System Requirements

Objectives:

- Understand how to translate the use case and sequence diagram analysis into system requirements; and translate system objectives into system requirements
- Introduce the concept of managing system requirements, developing a readable requirements organizing scheme (whether for a database model or a document)
- Understand the characteristics of "good" requirements.

3.3.5 Economics and Financial Analysis

Objectives:

- Estimate costs associated with production and operations
- Understand the concepts of cash flow, net present value (NPV), break even analysis, return on investment (ROI)
- Be able to create a financial analysis of a new product development.

3.4 Learning Integration

In this major section, the learning and results from the prior sections are assembled into an integrated whole. This is accomplished through a project presentation that serves as a funding proposal and conceptual design review, and a final report which serves as the high-level business and conceptual plan for a program. The project presentations are made after the modules have been presented, but before the students have completed the detailed work which is contained in the final report. A rubric is presented at the beginning of the course such that it can be reinforced in the subsequent instruction. The outputs from this course have been used as the preliminary design concept for a complete system development, which includes architecture and design, implementation, and sustainment in a four-course series that forms the core of a systems engineering of cyber-physical systems program.

4 Pilot Results

While much of the lecture and course materials for this course existed in a series of prior separate courses, they were first integrated into a single Systemic Design Engineering graduate course targeted for introduction in the summer of 2017. The pilot class was composed of eight professional masters students in an aerospace corporation. The greatest concern when assembling this course was not whether the materials were useful, but rather if it would be possible to successfully cover this amount of material in a single course. The prior version of the course had successfully combined elements of design thinking with systems and software engineering. In addition, it was believed that the course already was full of content even before adding the new major section on systems thinking and its four modules. The problem was addressed by comparing the learning objectives and modules from this course with those of a separate Systems Thinking course and determining the relative importance of the learning objectives in each as well as their overlap. The four essential modules from Systems Thinking were identified, and through the removal of some overlapping material and less critical material in the existing course, there was time to accommodate all of them.

Despite these challenges, the instruction of the material in the pilot course went smoothly with a small number of changes noted for subsequent courses. Most critical, however, was the assessment of the student's results to determine how well they had learned the concepts taught in the course. This was assessed subjectively by instructor in two different ways. The first assessment was in systems thinking concepts in which the comparison was between the results obtained in an existing course devoted entirely to systems thinking, and to this new course which included the core of systems thinking. The results were surprising. The systems thinking modeling and analysis in the Systemic Design Engineering course was judged to be the equal and, in some ways, superior to what is produced by similar professional master's students in a course dedicated to systems thinking. While the analysis may not have been as deep as would be seen in the dedicated systems thinking course, the quality of the analysis was at least on par. The lessened depth of the analysis is easily explained by the fact that final reports are limited in size and thus the systems thinking analysis is also limited.

The second area of comparison is in the design thinking, and systems and software engineering portions of the course. These areas also were judged to be on par or superior to the results seen in the prior conceptualization courses which did not contain the systems thinking material. The inclusion of the systems thinking material provides the basis for a more sustainable proposed system concept.

While the sample size is very small (the pilot contained eight students grouped into two teams), the results are promising in that the learning objectives of all the material in the Systemic Design Engineering course was mastered by most, if not all the students, without any notice of loss from that of the individual courses containing this material. The only noted downside was that some of the students noted that this course required more time than they had anticipated. However, this same comment was just as prevalent on prior courses that did not have this additional material.

5 Conclusions and Further Work

The results of creating a Systemic Design Engineering course for systems and software engineering masters engineering students were quite positive. The students in the pilot successfully mastered the systems thinking concepts that were added to the existing course, with results that were judged to be at least equivalent to what students were learning in a course dedicated to that subject. The additional material also provided an increased richness in the resulting designs. The course will continue to be refined based on feedback from the students and instructors, and a thorough analysis of the learning results. The future plan is to update the Systemic Design Engineering course such that it can be used as the design course for systems, software, cyber-physical and socio-technical systems masters' degree program, as the three elements of systems thinking, design thinking, and systems and software engineering makes it relevant to each of these four systems types in the Cynefin framework. It will be interesting to see how the course will be optimized to support design efforts for each of these different system types. It is believed that this understanding will enable the continued evolution of the course and the concept of Systemic Design Engineering as a multi-disciplinary design approach resulting in improvements that will provide benefits for all four system types.

References

1. Walden, D., Roedler, G., Forsberg, K., Hamelin, R., Shortell, T. (eds.): INCOSE Systems Engineering Handbook – A Guide for System Life Cycle Processes and Activities, 4th edn. Wiley Publishing (2015)
2. Boehm, B.W.: A spiral model of software development and enhancement. Computer **21**(5), 61–72 (1988)
3. Brown, T.: Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation. Harper Collins, New York (2009)
4. Buede, D.M.: The Engineering Design of Systems: Models and Methods. Wiley, New York (2000)
5. Chen, W., Lewis, K.E., Schmidt, L.: Decision-Based Design: An Emerging Design Perspective, Engineering Valuation & Cost Analysis, special edition on "Decision-Based Design: Status & Promise" 3(2/3), pp. 57–66 (2000)
6. Dym, C.L., Agogino, A.M., Eris, O., Frey, D.D., Leifer, L.J.: Engineering design thinking, teaching, and learning. J. Eng. Educ. **94**(1) (2005)

7. Giambalvo, J., Vance, J., Hoffenson, S.: Toward a decision support tool for selecting engineering design methodologies. In: ASEE Annual Conference and Exposition, Columbus, Ohio, 25–28 June 2017
8. Hildenbrand, T., Wiele, C.: The road to innovation: design thinking and lean development at SAP (2013)
9. Jones, P.H.: Systemic design principles for complex social systems. In: Metcalf, G. (ed.) Social Systems and Design, volume 1 of the Translational Systems Science Series, pp. 91–128. Springer, Japan (2014)
10. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H.: Engineering Design: A Systematic Approach, 3rd edn. Springer, London (2007)
11. Wade, J., Hoffenson, S., Gerardo, H.: Systemic design engineering. In: 27th Annual INCOSE International Symposium. Adelaide, Australia, 15–20 July 2017
12. Kurtz, C.F., Snowden, D.J.: The new dynamics of strategy: sense-making in a complex and complicated world. IBM Syst. J. **42**(3), 462–483 (2003). https://doi.org/10.1147/sj.423.0462. ISSN 0018-8670
13. Arnold, R.D., Wade, J.P.: A definition of systems thinking: a systems approach. Procedia Comput. Sci. **44**, 669–678 (2015)
14. Geertz, C.: The Interpretation of Cultures (1973)

Field Guide for Interpreting Engineering Team Behavior with Sensor Data

Lorena Pelegrin[1], Bryan Moser[1](✉), Shinnosuke Wanaka[2], Marc-Andre Chavy-Macdonald[2], and Ira Winder[1]

[1] Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
{pelegrin, bry, jiw}@mit.edu

[2] The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan
swanaka@s.h.k.u-tokyo.ac.jp, marc@m.sys.t.u-tokyo.ac.jp

Abstract. The design of complex systems is a challenge with many capabilities enabled by a recent generation of digital tools for modeling, simulation, and interaction. Relevant studies on teamwork from coordination science, learning, design, HCI, and serious games are briefly summarized. However, validation of resulting behavior and emergent outcomes given these much-needed new tools has been difficult. The recent availability of pervasive sensors may allow the creation of experiment platforms to increase empirical data from experiments, their scalability, and analyses towards reproducibility. This paper's approach treats engineering teamwork as a sociotechnical system and proposes instrumentation of teamwork across problem, solution, and social spaces. A quasi-experiment was conducted, with experts in the maritime industry exploring options for transition to natural gas infrastructure and shipping. The experiment derives a narrative of the engineering teamwork both from ethnography and digital sensors to uncover teamwork behavior. Thus, this work integrates disparate data to create mapping rules from sensors to story. We find the approach promising for the generation of sensor-derived stories and the potential for deeper and scalable studies on engineering teamwork.

1 Introduction

The study of engineering, like many fields, will be changed by pervasive sensors, data, and analytics. In particular, manual observation of engineering teams conducted by researchers may be supplemented or even replaced by "digital fingerprints" generated by pervasive and non-intrusive instrumentation. There is an opportunity for the research community to utilize sensors to increase the reproducibility, scalability and efficiency of teamwork experiments. Before researchers embrace sensors, however, a thoughtful understanding of the sensitivity and accuracy of these new data relative to and in combination with existing experimental methods, including ethnographic and survey based techniques, should be considered. This field guide introduces our recent experience in interpreting a narrative derived from a team's digital fingerprints.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 203–218, 2019.
https://doi.org/10.1007/978-3-030-04209-7_17

Also, we explore how team narratives generated from sensor data can differ from direct human observation.

The unit of analysis in this exploration is the team narrative, a record of an engineering team's collective behavior that includes moment-by-moment attention, rationale, decision making, solution generation, and solution performance. While an engineering team may consist of multiple individuals, this research departs from prior investigation of design teams of diverse individuals sharing common scope and objectives, to a broader view of the "team of teams" with different (yet coupled) scope and various agenda. A team narrative tells the story of a semi-structured design experience, during which teams utilize their attentions, tacit knowledge, dialogue, and tools to explore solution(s) for a given challenge.

A narrative acts as frame to gain a deeper understanding of the challenge, the teams, and how these teams came together to interact, design, and implement solutions. A useful narrative explains and predicts workshop performance outcomes. A researcher may derive a teamwork narrative through primary observation of an engineering team (i.e. watching a team in action) and by secondary observation of a team's digital fingerprints. We are interested in how narratives derived from digital fingerprints differ or complement those generated by primary observation. Differences to consider are specificity, accuracy, and teams' affirmation of a narrative's content.

In this paper, a framework to transform data from engineering design teamwork into team narratives is proposed, and it is applied to some experiments. On the basis of our experience in these experiments, we discuss what steps a researcher can take to improve observations derived from pervasive and non-intrusive instrumentation in complement to direct observation by humans.

The research contributions of this paper are:

1. a formalized concept of a team design walk narrative and a taxonomy of a design experiment for understanding team dynamics;
2. a proposed experimental setup generating team narratives and a set of mapping rules for transforming sensor data into a team narrative;
3. a demonstration of the aforementioned setup and rules, by showing some experimental data and actual narratives of design.

2 Related Research

The complexity of a SoS (Systems-of-Systems) is two-fold – the technical system and the social dynamics through which humans design, develop and use said system. Researchers have studied how to manage system complexity and to design the SoS successfully. For example, NASA provides general guidance and information on the systems design process as a handbook [1]. They propose 3 processes: system design, technical management, and product realization, and build one overarching framework including interaction amongst these processes. Even with clear standards, as target systems grow more complex, the human aspects, i.e. behaviors, motivations, deeply held assumptions, thinking and decisions, may become equally significant to outcomes. Complex problem-solving requires collaboration across many teams. In this context, how should teams be organized and behave for the engineering of systems of systems? To begin, related research in collective intelligence, learning, design studies, human-computer interaction, and serious games are reviewed.

Collective intelligence (CI) is a group's emergent capability, defined by Malone et al. as "groups of individuals acting collectively in ways that seem intelligent." Several papers [2–4] report a psychometric methodology for quantifying CI, reflecting how well groups perform on a diverse set of group problem-solving tasks. These studies suggest that CI is not strongly correlated with the average or maximum individual intelligence of group members, but is instead correlated with: the average social sensitivity of group members; the equality in distribution of conversational turn-taking; and the proportion of females in the group. Malone [2] also notes that some scholars equate intelligence with learning.

Learning is associated with change in knowledge which may be gauged from change in performance, but also change in group processes or repertoires. Sensing knowledge changes in groups is difficult, as it may either explicit (easily codified and observable), but other times tacit (unarticulated and difficult to communicate). Moser [5] reviews and categorizes models of learning: models based on experiential learning [6, 7], reflective practice [8, 9], learning as knowledge conversion [10], and expansive learning [11]. Valkenburg [9] discusses the so-called Element of Surprise introduced by Schoen as initiator of the reflective practice process that ultimately underpins the results of the design process. Valkenburg argues that, for design teams, surprises become social events. Further, Stompff et al. [12] revise Valkenburg's model of the mechanism of reflective practice to include surprises, things that "fall outside of the current frame", and require reflection. They argue that surprises are a source for team learning and innovation.

Studies of designers in practice—"design observatories"—are a strongly empirical approach. For example, Carrizosa et al. [13] developed a special room for the design observatory, installing four cameras to capture designers' behavior and work. Milne et al. [14] installed an interactive whiteboard and attempted to capture work processes. Törlind et al. [15] showed two uses of the design observatory, to observe social aspects of synchronous team-based design and to understand design activity through iterative prototyping. Similarly researchers in *design thinking* have conducted series of experiments, often in small teams and centered on iterative prototypes using various tools and rules of interaction [16]. A recent application extended this approach to consideration of teamwork in healthcare [17].

Human-Computer Interaction (HCI) research began grounded in the premise that computing systems had become inherently characterized by human factors [18]. Contemporary HCI researchers have studied the interaction of teams through digital means as computer supported cooperative work, or CSCW [19, 20]. A taxonomy called the Groupware Matrix characterizes team interaction by two dimensions: location and time [21]. Location indicates a team's co-location (or not), while time refers synchronous (or asynchronous) interactions by individuals. A CSCW system's position within the Groupware Matrix has guided the classification of teamwork observations according to this technological context.

Typically, HCI experiments had relied on small sample sizes (10–100 users), constrained by the need to gather participants and record data [18]. Remote collaboration tools have recently been able to achieve much larger sample sizes (10,000–100,000 users), yet synchronous-collocated environments, where users must be in the same place at the same time, are still difficult for researchers to scale. Additionally, human cognitive phenomena previously studied in individuals have been extended to the collective level, such as memory and learning for groups [22].

Simulations and Serious Games are *s*imilar to the complex systems architectural decision models treated in this paper. Grogan and de Weck [23] developed an interoperable simulation game and applied it to infrastructure systems. Infrastructure design involves cross-sector interactions amongst stakeholders - that is, highly collaborative design. They conducted experiments to identify what features of a design tool lead to more effective collaborative decisions, and they demonstrated technical data exchange supported by integrated and synchronous tools is correlated with effective designs. Their analysis is focused on technical aspects of the design process, and these insights are limited in that the supporting data comes only from their own design tools.

In summary, related studies across several disciplines are relevant to the motivation of this paper. Our examination of related research was limited given these broad multiple sources. CI research provides us with a measure of group intelligence and insight about what may be correlated with intelligence. Frame reflection and learning guide us to develop sensors that can capture mental models and surprises during teamwork. Design studies capture processes and the generation of ideas and solutions. In contract, HCI research teaches us to measure the means for interaction across interfaces, described as channels of technology with timing, location, and purpose.

Our approach is similar to design observatories with three contributions: problems framed by well-articulated systems models, increased interactive visualization for real-time exploration of complex SoS, and new sensors for data capture. This experimental framework characterizes activity across the problem space, the solution space and the social space for engineering teamwork. The framework takes experiment repeatability into consideration, in order to ease the implementation of quasi-experiments and, through improvement, increase robustness of experimental design. Moreover, interactive visualization based data capture, shown as a "design walk", is applied to observe interaction between problem and solution spaces. These sensors allow the researchers to see all events as the solution space is explored. Further, the design walk can also visualize the relationship between the solution space and social space. With the addition of audio, video, and ethnographic notes, it is possible to observe meaningful interactions amongst the three spaces.

Our research seeks to detect how team attention and activities map to the problem space (manifested as need and, values of stakeholders), solution space (as requirements, function and form of the technical system) and the social space (through roles, capability, motivation and power of the agents). We assert that a sociotechnical event (e.g. awareness of a need, exposure of an assumption, a decision about the technical solution) will have effects across all 3 spaces. This assertion follows a broad view of design as a social activity, of teamwork during complex problem solving having stakeholder, organization, process, and product considerations [24, 25].

Decades of research in the social sciences provide important input to our research approach, including many case studies, meta-analysis and so on. However, a frustration is that these studies are often difficult or infeasible to reproduce, nor scalable to industrial teams of teams. For example, while it is widely accepted that overall efficiency in organizations is achievable with high performing teams and learning organizations [26], questions remain of what defines a high performing team and what metrics measure organization learning. Google's Project Aristotle, which reviewed academic studies on how teams work, did not establish strong patterns to define the "ideal" makeup of the team that achieves best team effectiveness [27].

Ideally, a scientific body of knowledge should be scalable to industrial relevance and produce reproducible insights, in order to increase collective cognitive capability by teams of teams for complex systems problems. Moreover, a final objective should be to reveal the mechanisms inside teams working in complex problems, a sociotechnical physics. Previous studies, including inspired ethnography, great thinkers and insightful writers are relevant guides, yet we cannot be sure without uncovering the underlying phenomena with reproducible experiments. Indeed, insightful case studies might be only shadows of the underlying phenomena. This work is an early attempt to seek the underlying science of teamwork for complexity, and the first principles of sociotechnical systems.

3 Methods

In this section, the following methods are explained; a simple taxonomy of the design process, an overview of the quasi experiment, and then a method to instrument the experiment, collect data, and generate design walk narratives.

3.1 Taxonomy of Design Process and Experiment

The design challenge in this paper has goals, and design work is an activity of a teams-of-teams (ToT) seeking feasible solutions which best achieve those goals. *Goals* are defined as the desirability of what is to be accomplished, and can be expressed as targets and metrics, often called *-ilities*, *KPIs* and so on. For complex systems problems, a design process most often includes coupled goals across multiple dimensions, thus driving the teams to analyze options, conduct trades, and make a decision about their preferred solutions. To generate and pick preferred solution(s) from a tradespace, the teams may have in mind a *strategy*, including the prioritization of goals, a ranking of the metrics, ways of organizing their teams' capabilities, and a differentiated process (compared to others').

We consider the teams during a design process as acting and interacting across *solution*, *problem*, and *social spaces*. Figure 1 shows the overview of this framework. The solution space is explored through component design choices, in combination generating the set of design configurations which are available. The problem space is driven by needs, the desirability of fulfilling these needs, and how these needs interrelate, value traded and shared as manifested by stakeholders. The social space includes the rules of organizational power, hierarchy, knowledge, capabilities and so on.

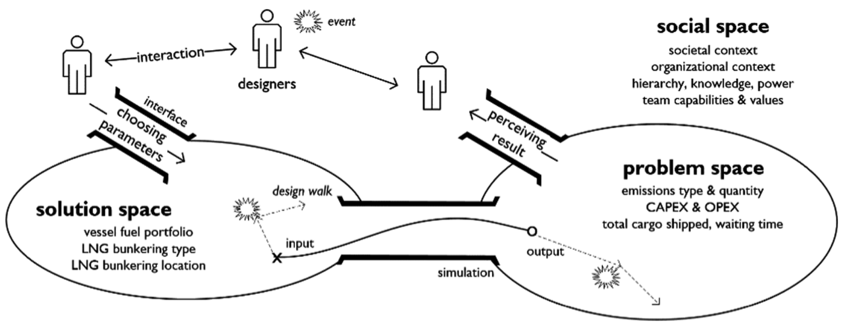


Fig. 1. A nomenclature for the design process, which consists of a design walk and events occurring simultaneously in the problem, solution, and social spaces - which forms the context for problem, solution and teamwork.

A team's behavior in the social space affects their awareness, exploration, and selection of solutions, which is their behavior as seen in the solution space. The outcome of their selection in the solution space is expressed in the problem space, and a decision on whether they are satisfied with the result or not is also affected by the social space.

Thus, the underlying events that make up a design process interact simultaneously across solution, problem and social spaces. By tracking these events and interactions, *instrumented teamwork experiments* attempt to reveal teams' dynamics moment by moment. A *design walk* is the path taken by the team over time to consider, generate, evaluate, and iterate design alternatives. While along the design walk, a tradespace is explored and exposed. We characterize the happenings moment by moment during the design walk as *sociotechnical events* existing simultaneously in all three spaces. Solution and problem spaces are connected via an assessment or analysis method - here simulation - while they are connected to the social space by some interface, here an interactive visualization software. Teamwork of interest takes place in the context of this design activity.

3.2 Experimental Framework and Setup

Figure 2 shows an overview of the experimental framework & setup. There are two types of data sources: sensors, consisting of Design Support System (DSS) logs and microphones, and direct feedback from human participants or observers. DSS logs include attention (to output KPIs), input & output logs, while direct feedback includes a "surprise detector" (manual by the participant, on a spreadsheet), scratch pad for participant use during the experiment, post-surveys, and in some cases observation and interviews by experimenters. The microphone records the participants' audio, though in evolutions of this setup, we intend to also have the capability to simply detect presence or absence of conversation; likewise it is hoped a simple "surprise detector" might be made automatic, pervasive and non-intrusive.

In the post-survey, several questionnaires are given individually to the participants which collect data about e.g. their strategy, agreement with the group, result etc. The scratch pad is used to collect data about the team's rationale in their selection in the

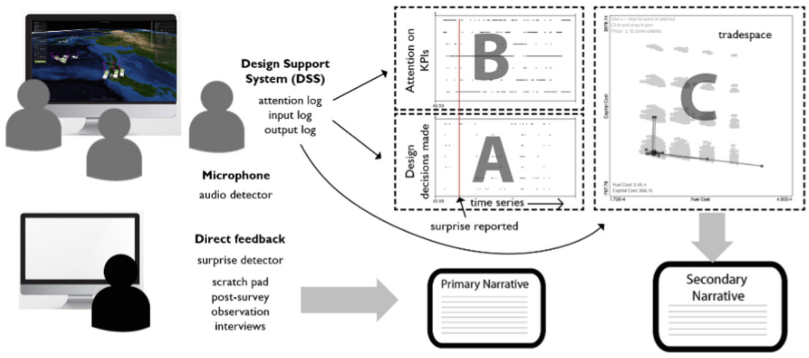


Fig. 2. A conceptual diagram of the experiment setup and research flow. During design experiments, sensors consist of DSS logs and microphones and "direct feedback" by human participants or observers. Sensor data is displayed, and both are interpreted into narratives.

solution space, and surprises. The "direct feedback" in particular may be used to generate a holistic (primary) narrative of the design walk, for analysis, supported by sensor data. However, it is preferable to create a narrative *mainly* based on inexpensive, non-intrusive sensor data - a *secondary narrative*. Such sensors record: (A) design decisions made (or attention allocation to inputs), (B) attention allocation to KPIs, and (C) design walk in the problem space. By comparing the primary and secondary narratives, we may assess the validity of the sensor interpretation. We seek to develop an inexpensive, pervasive, non-intrusive sensor package that can make a secondary narrative similar in quality to a primary.

Observations are made over the course of a design experiment, sensing how attention is allocated to different parts of the system at hand. A graphical user interface (GUI) enables users to deliberately choose to render or activate mutually exclusive parameters of information. Attention is "allocated" as long as the user is viewing or activating a certain parameter. For DSS inputs (*A* in Fig. 2), a unit of attention is allocated if an input parameter is changed between discrete simulation events. A simulation event occurs when a team evaluates a selected architecture using the simulator. In the case of inputs, attention is modeled as an instantaneous blip when the "Simulate" button is pressed. For DSS output calculation (*B*), attention to a particular output parameter (e.g. *Fuel Cost*) is allocated continuously as long as the output is selected by the user for viewing on the tradespace plot (*C*). Since the tradespace plot only has two axes, the GUI only allows a team to view two of the seven KPIs at a time. One view of a team's design walk can be visualized on such a plot by how two selected output parameters' values change from one architecture to the next. Outputs of consecutive architectures are connected by a line, and the solution chosen by a team, represented by a larger circle (see Fig. 4).

3.3 Experimental Case and Procedure

This subsection provides brief explanation of our design experiment procedure; a detailed explanation is outside the scope of this work. The procedure consists of 6 steps: (1) pre-survey, (2) design case review, (3) tutorial about the software, (4) design challenge, (5) collective discussion, and (6) post-survey. The target case is the fuel upgrade of a maritime logistic system, including fleet and port. The background is that responding to the new IMO (International Maritime Organization) MARPOL emissions reduction regulation, the maritime industry needs to think about how to change the current transportation system. Participants are mid-career practicing engineers, divided in teams of 3, and must decide the portfolio of ship fuel type, and the number, type and location of possible LNG bunkering facilities. The design challenge lasts 90 min. Much more information is in Pelegrin [28].

3.4 Method for Generating Team Design Walk Narratives

Figure 3 provides an overview of the process of this research, focused on generating narratives and rules to interpret sensors. It is in the context of design experiments, as explained earlier. An instrumented DSS and design experiment generate both direct (human) feedback and sensor data. These can be analyzed to discover new phenomena. The focus of this work is to improve the experimental setup, by shifting towards more sensor data. To do this, we evaluate the ability of sensors to capture as much (or more!) phenomena as direct feedback. Sensor data is used to generate holistic narratives of the experiment, and compared to purely "human-based" narratives. Stepwise, some narratives are constructed using all available data and feedback; these are then shown to participants, who anonymously rate their fidelity on a 5-point scale and comment on

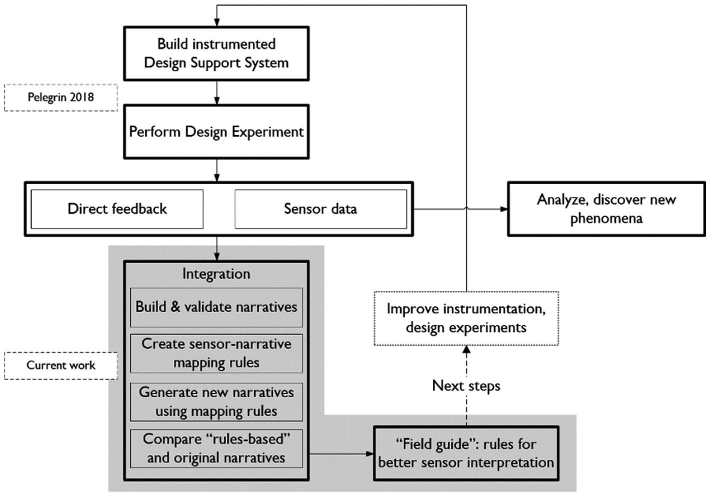

Build instrumented Design Support System
Pelegrin 2018
Perform Design Experiment
Direct feedback
Sensor data
Analyze, discover new phenomena
Integration
Build & validate narratives
Create sensor-narrative mapping rules
Generate new narratives using mapping rules
Compare "rules-based" and original narratives
Current work
Improve instrumentation, design experiments
Next steps
"Field guide": rules for better sensor interpretation

Fig. 3. Overview of the experimental framework and procedure. The current work is in grey, concerned with integrating data to generate holistic design walk narratives.

any errors or omissions. These validated narratives are used to create tentative "mapping rules" from a sensor to a fragment of the narrative that it could inform. These rules are then used with other design experiments' sensor data only, to generate new sensor-based design walk narratives. These are similarly shown to participants, and we seek any difference in score or feedback between the primary and secondary narratives.

4 Results

The following design walk narrative was generated for a team from an experiment in March 2018 with an emphasis on "direct feedback": written rationale, post-surveys and interviews - supported by observation and DSS data. Figure 4 is the sensor data corresponding to the narrative.

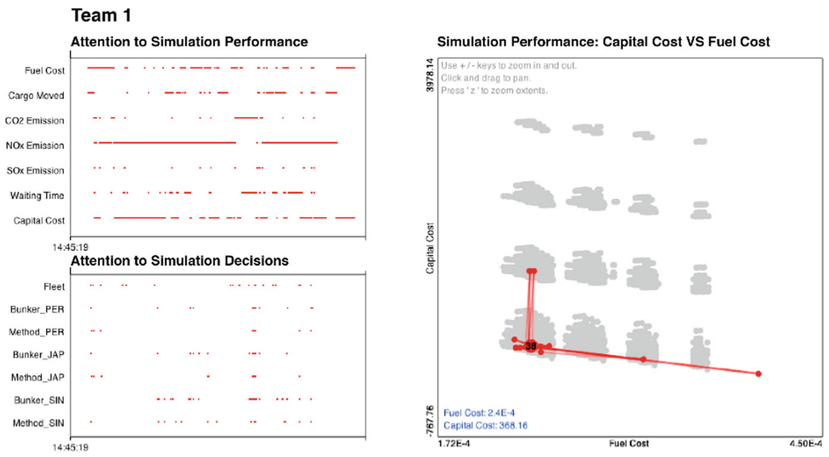


Fig. 4. A "digital fingerprint" of a team's activity while performing a design walk.

Primary Narrative: *This team interprets the design task literally: to reduce emissions at low cost, though are concerned at the lack of comparison of emissions to regulation. They also realize that waiting time should be considered, but decide to neglect the amount of cargo moved, because of unclear interpretation of this KPI. Thus they consider mostly NOx, CAPEX and OPEX, checking other KPIs also. Their design walk has 3 phases. Firstly, there is some initial exploration with multiple parameters. Secondly, they focus on LNG options and investigate bunkering facilities, finding the best combination of location and methods. A key surprise for them is that inexpensive truck-to-ship bunkering seems sufficient, and located in Singapore might be best. Thirdly, they try to change fleet composition. The 2 team members from the maritime industry seem to be like-minded, possibly because of their pre-existing relationship and shared (hierarchical) culture. On the other hand, the third designer may be less influential because of a cultural barrier. When recording, they usually signal "surprise" - options better or worse than expected - yet discussions suggest they don't* *have strong expectations, instead taking a "wait-and-see" approach when simulating. This is consistent with higher model confidence (or less engagement), and their goal acceptance, and shows no signs of struggling with a pre-existing mental model. Finally, they select LNG-fueled ships, and 1 truck-to-ship bunkering facility in each of Singapore and the Persian Gulf. The design has low emissions from LNG, but might reduce overall cargo moved by 5%.*

The narrative above was created in a holistic manner. The observer was very familiar with the design exercise, the problem itself, and many of the participants and their style of working. After showing the narratives anonymously to the participants, on the whole, good fidelity scores were given (average 4 out of 5), with relatively minor suggestions. Thus, such narratives at least seem to be accepted by the original team.

We attempt to parse the design walk into key elements, to ascertain their provenance, and to identify sensors and rules that could facilitate a digitally supported generation of narrative. The result for the above narrative is shown in Table 1. Parsed narrative "fragments" are listed and examined for their origin - typically from human observations or written design rationale (see the 1st and 2nd columns). The 3rd column indicates sensors, or proposed sensors, to furnish this information in part or in whole. Finally, the 4th column explains how the sensor might be used to provide that narrative fragment. For instance, "low model confidence" is relatively difficult to determine, and perhaps reported in a team's postsurvey, yet two types of sensors may provide strong hints: the *surprise detector* and *input change log*. Frequent or early surprises in a design walk are plausibly correlated with lower model confidence; an input log showing the team engaging in systematic One-At-A-Time (OAT) model-testing behavior likewise plausibly indicates they may be evaluating the model, rather than accepting it directly. Seven example narrative fragments are shown, and proposed rules range from trivial to very sophisticated or perhaps impractical. However, it is noteworthy that every identified fragment has some proposed sensor and mapping, although several will only allow partial information (e.g. identifying "fuzziness" of requirements from sensor data) (Fig. 5).

Three path dependent sequences are identified in Team 1's outcomes and design changes time series, considering the surprises identified. In the first sequence, after an initial phase of exploration, the team fixes the Ship Portfolio choice to 20x LNG Ships and further explores bunkering options, this phase is triggered by Surprise 1 on better Emission Reduction than expected (i.e. an LNG fleet works well). The second path dependent sequence is triggered by Surprise 4 on better Initial Cost and Fuel Cost Efficiency than expected (i.e. Truck-to-Ship bunkering could be the best option with lower costs). In this sequence the team fixes no bunkering at Persian Gulf and Japan, along with a one Truck-To-Ship bunker in Singapore. The team then experiments with different Ship Portfolio options. The third path dependent sequence is also triggered by Surprise 4. In this case the team fixes one Truck-to-Ship bunker in both Persian Gulf and Japan, and experiments with variations of Ship Portfolio options and bunkering infrastructure in Singapore. The selected architecture by Team 1 has features of these three path dependent sequences. Therefore, this data suggests that local Surprises 1 and 4 (Key Surprises) had global or systemic significance driving the team's architecture selection decision (Fig. 6).

Table 1. Selected mapping rules, from sensor data to design walk narrative

Narrative fragment	Source of narrative	Sensor/ *proposed*	Mapping rule/*proposed*
Model confidence	Comments from scratch sheet, post survey, observation	*Surprise detector, input logy*	*Fequent or early surprises may indicate conflicting mental model; an input log showing OAT model factor testing is low-confidence*
Prioritized preferences for decision	Human - design rationale	*Output log, attention log*	*Avoided or preferred area of problem space: suspect key. Expected attention on key variable*
Phases of design walk	Time series of aggregated input	Input log, output log	Look for pattern in input action "macro" time series, results
Preferences - KPIs "satisfied"(e.g. "fuel type- LNG is good")	Human - scratch sheet comments	*Input log, output log, attention log*	*Sequence: change input levers, achieve "good" output, then leave these levers unchanged & explore other levers; maybe change attention*
Key surprises (learning) (e.g. "1 truck is enough!")	Human - scratch sheet comments (on surprises), post survey	Surprise detector, input log, output log, attention log	After surprise, "path dependent sequence": marked change in behavior - use different input levers, attention, maybe output trends
Accidental result	Combined output and attention logs	Attention log, output log	Good result, but no attention paid to this KPI in the log

Utilizing 16 rules (of which 6 are in Table 1), a secondary narrative was generated for another sensor footprint of the design walk, shown below:

Secondary Narrative: *From their attention to KPIs, the team's goal appears to be interpreted literally: reduce emissions at low cost, but keeping cargo moved nominal. However, emissions attention & outcomes are slightly less disciplined than for cargo moved and OPEX & CAPEX - the team may have made some minor change in goal emphasis (indeed, they often return to check NOx later). But we see no clear sign of perceiving a goal or requirement to be ambiguous or unclear. We do not suspect low model confidence, as no One-At-a-Time (OAT) model testing behavior was observed. We may segment the design walk into two broad phases: early on, larger KPI fluctuations occurred with fleet composition, and then later they focused on options around LNG bunkering facilities. Approximately halfway through, there was also an intensive period of considering Singapore options, particularly for OPEX, CAPEX and cargo moved. A "satisficing" event occurred, for a half hybrid, half LNG fleet with non-shore bunkers - a solution subspace which was never left. It followed two key surprises: a positive one which preceded keeping the hybrid fleet composition, and a negative one*

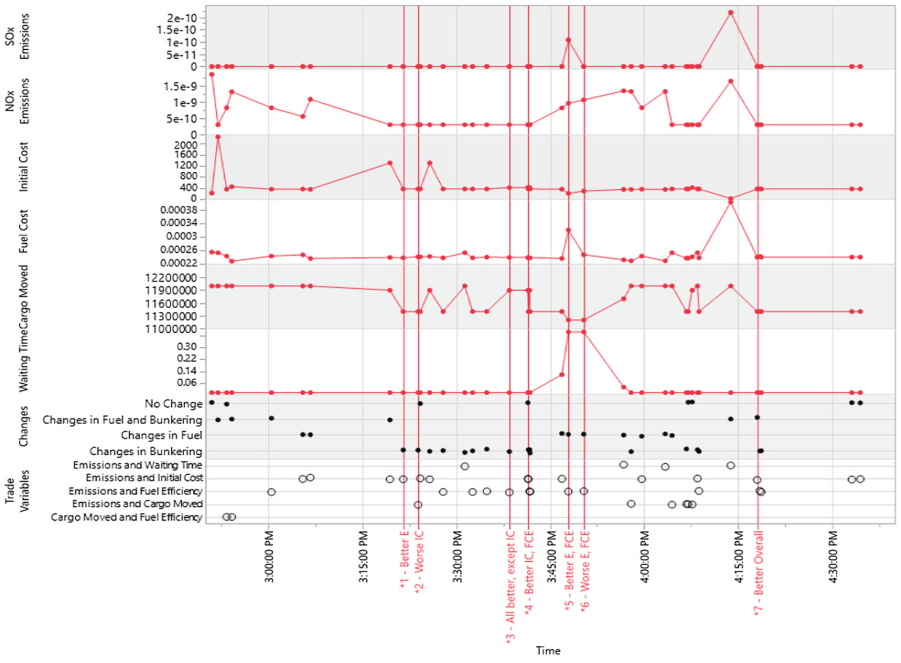


Fig. 5. Example Outputs over time (horizontal lines), Input Aggregated Changes over time, Trade Variable Pairs consulted over time (empty dots), Surprises over time (vertical lines).

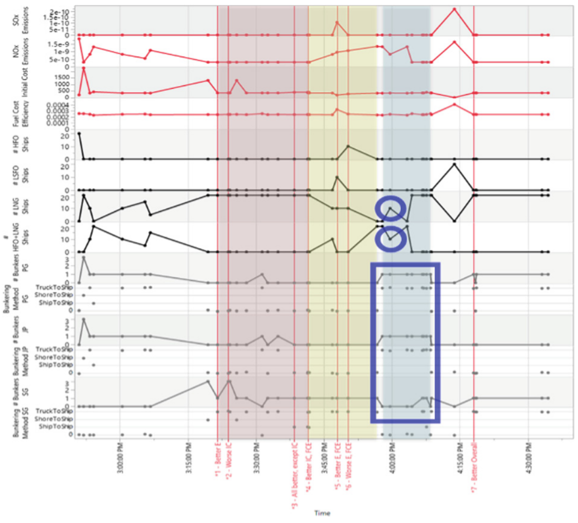


Fig. 6. Design Walk as interpreted. Blue circles and squares indicate key events.

when trying 3 shore bunkers, leading to the largest cost peak of the walk, after which shore bunkers were always avoided. As this coincided with the largest cost peak. By contrast, 9 "surprises" were detected in groups later in the walk, but they were associated to small KPI changes, so we suspect they are considered minor. As for degree of team consensus, a proposed audio analysis was unsuccessful, yet the walk appears steady & unperturbed.

5 Discussion

The first, "primary" narrative exposes information from the path of exploration, which we refer to as the *design walk*. Some of the path events are relevant to goals and their attributes - interpretation and clarity, requirements and preferences; KPIs and their prioritization are neglected. Other signals reveal phases of the design walk, characterized by different primary activities, subspaces, or approaches. In turn, these phases are marked at the boundary by surprises or learning which trigger a new phase; relative influence, hierarchy & agreement in the social space; general approach to tool usage; trust in the model, mental models, and their pre-existence or conflict. These terms begin to populate a taxonomy of design walk characteristics this experimental setup can capture, refining the simple taxonomy illustrated in Fig. 1. Comparing the terms to the framework of Fig. 1, the social space seems underrepresented - perhaps unsurprising since few sensors were intended to capture it so far. However, it is somewhat encouraging that in our early feedback responses, none of the participants identified any large gaps in the narrative, and rated high fidelity. This suggests our initial approach can capture meaningful characteristics of a design walk.

Some of these aspects are more easily captured by the sensor footprint than others. At this early stage, across the limited datasets surveyed (10+ in this experiment), more easily capturable characteristics include the KPI prioritization, phases of design walk, and the approach to tool usage, while information about goals (details or interpretation) and social influence is among the most difficult. Model trust, mental models, and surprises are somewhat intermediate - the sensor footprint provides some strong hints, but not a full story. In fact, early evaluation suggests "secondary" narratives capture many of the phenomena from the direct feedback sources, and we anticipate modest differences in fidelity scoring between primary and secondary narratives (though data analysis continues). However as can be seen in the examples above, secondary narratives may only *hint* at many characteristics, while direct feedback data is more amenable to a stronger assertion. Between the various sensors, so far, no obvious data inconsistencies have been found, nor any with the direct feedback data. Comparing the sensors' merits, we find that unsurprisingly the DSS logs are most useful so far, perhaps particularly the attention log - however this suffers from the drawback of forcing a 2-dimensional tradespace view. Used together with a time series of input and output data, a rough picture of the design walk can be quickly created (see Fig. 4, though output time series not shown). Desired improvements include an "automatic" surprise detector, and "conversation detector" with voice recognition. This should provide more insight into the social space.

The mapping rules and field guide are to serve two overall purposes (Fig. 3): aid data interpretation, and improve experimental setup. Both should reduce dependence on difficult-to-obtain human-sourced direct feedback data, succeeding primary narratives with high-fidelity secondary narratives (see Fig. 2). How well does our initial stab work towards this goal? Table 1 shows some sample rules to holistically characterize a design walk using sensor data; they are among the more promising of 16 rules (and growing). We note that diverse and fundamental aspects of the walk can be discussed using sensors - the most promising rules so far may yield insight on prioritization, model trust, phases/modes of activity, and depth of surprise/learning. Many of these rules are low confidence, but more data, particularly validation from participant ratings, should improve it. Interestingly, some sensor data may also be *better* than direct feedback - akin to "revealed preferences". For now, human feedback is key to intent & the social space, and to validate sensor-based narratives.

5.1 Lessons Learned

In many ways the quasi-experiments described in this paper were important for our research team to build capability to experiment by experimenting. We are at the beginning of a journey to adopt recently available instrumentation and analytics techniques into our research strategy. As such, we developed lessons learned and awareness of limitations to carry forward into a next series of workshop experiments.

One choice for the design of experiments is to constrain less or more tightly the teamwork goals, workshop format, and learning events. A real-world environment will often proceed fluidly, without the sorts of experimental control that may benefit the science of teamwork yet in turn enforce an unnatural condition. By not further specifying (constraining) both (i) the goals of the design exercise with more explicit design targets, but also (ii) the experiment format and UI, it could be observed that teams approached the problem in a wider variety of ways. This might provide the researchers with a broader view of the ways that designer teams approach problems.

Though we have been considering surprises as the triggers of learning and path dependency, it might be useful to expand the categories of learnings to include events related to other sorts of mental model development and validation.

Another learning mechanism to be included in categories of learning is a latency phenomenon – obtaining a new insight through surprise but not changing an architectural decision accordingly (this could be thought as "considering a change, but not executing it"), and later on in the design walk confirming the insight with another surprise. In other words, a surprise confirms an "initial surprise" in the same design walk which likely drafts a mental model.

Participants wore individual voice recorders hanging as badges, with the experimental intent was to obtain a clear recording of their voices and so transcript their discussions. It has been difficult to obtain faithful indexing of the audio files with state-of-the-art NLP software tested. Although NLP software providers claim to be able to transcript audio speech-to-text, perform speaker indexing, separate background noise and voice activity, as well as perform keywords extraction, and text sentiment analysis, the conditions of the meeting (background noise of other teams, different accents, microphones distant from voices, and participants talking over one another) did not allow for sufficient speech clarity.

6 Conclusion

In this paper, a framework to transform data from engineering teamwork into team narratives is proposed, and it is applied to a quasi-experiment. On the basis of our experience in these experiments, we discuss what steps a researcher can take to improve observations derived from pervasive and non-intrusive instrumentation in complement to direct observation by humans.

The contributions of this paper are:

1. a formalized concept of a team design walk narrative and a taxonomy of a design experiment for understanding team dynamics;
2. a proposed experimental setup generating team narratives and a set of mapping rules for transforming sensor data into a team narrative;
3. a demonstration of the aforementioned setup and rules, by showing some experimental data and actual narratives of design.

The design of complex systems is a challenge with many promises from the recent generation of digital tools for modeling, simulation, and interaction. However, validation of resulting behavior and emergent outcomes given these much-needed new tools has been difficult. The recent availability of pervasive sensors may allow the creation of experiment platforms to increase empirical data from experiments, their scalability, and analyses towards reproducibility. We find the approach promising for the generation of sensor-derived stories and the potential for deeper and scalable studies on engineering teamwork.

References

1. Hirshorn, S.R., Voss, L.D., Bromley, L.K.: NASA Systems Engineering Handbook (2017)
2. Malone, T.W., Bernstein, M.S.: Handbook of Collective Intelligence. MIT Press (2015)
3. Woolley, A.W., Chabris, C.F., Pentland, A., Hashmi, N., Malone, T.W.: Evidence for a collective intelligence factor in the performance of human groups. Science, 686–688 (2010)
4. Malone, T.W., Laubacher, R., Dellarocas, C.: The collective intelligence genome. MIT Sloan Manag. Rev. **51**(3), 21 (2010)
5. Moser, H.A.: Systems Engineering, Systems Thinking, and Learning: A Case Study in Space Industry. Springer (2013)
6. Kolb, D.: Experience as the Source of Learning and Development. Prentice-Hall, Englewood Cliffs (1984)
7. Ross, A.N.: Knowledge creation and learning through conversation: a longitudinal case study of a design project (2003)
8. Schon, D.A.: The Reflective Practitioner: How Professionals Think in Action, vol. 1. Basic Books, New York (1983)
9. Valkenburg, R.: The Reflective Practice in Product Design Teams (Ph.D. thesis). Delft University of Technology, Netherlands (2000)

10. Nonaka, I.: A dynamic theory of organizational knowledge creation. Organ. Sci. **1**, 14 (1994)
11. Engeström, Y.: Learning by Expanding. Cambridge University Press (2014)
12. Stompff, G., Smulders, F., Henze, L.: Surprises are the benefits: reframing in multidisciplinary design teams. Des. Stud. **47**, 187–214 (2016)
13. Carrizosa, K., Eris, Ö., Milne, A., Mabogunje, A.: Building the design observatory: a core instrument for design research. In DS 30: Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik, pp. 37–42
14. Milne, A., Winograd, T.: The iLoft project: a technologically advanced collaborative design workspace as research instrument. In: DS 31: Proceedings of ICED 2003, the 14th International Conference on Engineering Design, Stockholm, pp. 315–316 (2003)
15. Törlind, P., Sonalkar, N., Bergström, M., Blanco, E., Hicks, B., McAlpine, H.: Lessons learned and future challenges for design observatory research. In DS 58-2: Proceedings of ICED 09, the 17th International Conference on Engineering Design, vol. 2, Design Theory and Research Methodology, Palo Alto, CA, USA, 24–27 August 2009
16. Yang, M.C.: Observations on concept generation and sketching in engineering design. Res. Eng. Des. **20**(1), 1–11 (2009)
17. Rosen, M.A., Dietz, A.S., Yang, T., Priebe, C.E., Pronovost, P.J.: An integrative framework for sensor-based measurement of teamwork in healthcare. J. Am. Med. Inform. Assoc. **22**(1), 11–18 (2015)
18. Lazar, J., Feng, J.H., Hochheiser, H.: Research Methods in Human-Computer Interaction. Morgan Kaufmann (2017)
19. Baecker, R., Grudin, J., Buxton, W., Greenberg, S., Chui, M.: Readings in human-computer interaction, towards year 2000. Libr. Inf. Sci. Res. **18**(2), 187–188 (1996)
20. Wilson, P.: Computer Supported Cooperative Work: An Introduction. Springer Science & Business Media (1991)
21. Johansen, R.: Groupware: Computer Support for Business Teams. The Free Press (1988)
22. Kirschner, P.A., Sweller, J., Kirschner, F., Zambrano, J.: From cognitive load theory to collaborative cognitive load theory. Int. J. Comput. Supp. Collab. Learn. 1–21 (2018)
23. Grogan, P.T., de Weck, O.L.: Collaborative design in the sustainable infrastructure planning game. In: Society for Computer Simulation International, p. 4 (2016)
24. Moser, B., Mori, K., Suzuki, H., Kimura, F.: Global product development based on activity models with coordination distance features. In: Proceedings of the 29th International Seminar on Manufacturing Systems, pp. 161–166 (1997)
25. Moser, B.R., Wood, R.T.: Design of complex programs as sociotechnical systems. In: Concurrent Engineering in the 21st Century, pp. 197–220. Springer (2015)
26. Argyris, C.: Double loop learning in organizations. Harvard Bus. Rev. **5**, 115–125 (1977)
27. Duhigg, C.: What Google learned from its quest to build the perfect team. NY Times Magazine, 26 (2016)
28. Pelegrin, L.: Teamwork Phenomena: Exploring Path Dependency and Learning in Teams during Architectural Design of Sustainable Maritime Shipping Systems [Master of Science in Engineering and Management]. Massachusetts Institute of Technology (2018)

A Review of Know-How Reuse with Patterns in Model-Based Systems Engineering

Quentin Wu[1](✉), David Gouyon[2], Éric Levrat[2], and Sophie Boudau[1]

[1] Zodiac Aero Electric, 7, Rue des Longs Quartiers, 93100 Montreuil, France
{quentin.wu, sophie.boudau}@zodiacaerospace.com

[2] Université de Lorraine, CNRS, CRAN, Faculté des Sciences et Technologies, BP 70239, Vandoeuvre-lès-Nancy, France
{david.gouyon, eric.levrat}@univ-lorraine.fr

Abstract. The increasing complexity of systems to be developed requires engineers to review their practices in order to improve the efficiency of engineering and meet the needs of a competitive market. That is why models supported by formal or semi-formal languages are preferred to avoid the understanding variability of natural languages. In this context, Model-Based Systems Engineering (MBSE) made it possible to change the engineering paradigm by putting forward a unique, shared system model. To promote its adoption, a solution would be to allow reuse of knowledge and know-how, to encourage engineers seizing and adapting MBSE to their needs. This paper aims to review and evaluate the concept of patterns towards reuse in engineering, especially in a MBSE approach.

1 Introduction

The design of increasingly complex systems is implicating longer engineering phases and greater costs during the design lifecycle of a project. Those negative impacts are accentuated by the current document-centred application of Systems Engineering (SE) processes inside companies. Indeed, system development teams are working on standalone models, communicating with other teams through documents written in natural language. This implies a huge work concerning consistency and comprehension, as information shared through those documents has to be comprehensive and unique, to avoid rework and non-conformity to customer expectations. So, there is a challenge concerning the engineering efficiency (how to enhance productivity, quality, communications, and reduce risk) needed in a highly competitive environment, where the need is to shorten engineering cycle period.

In order to manage complexity, maintain consistency, and ensure traceability during systems engineering, the SE community has turned to the Model-Based Systems Engineering (MBSE) (Estefan 2008). Popularized by the International Council on Systems Engineering (INCOSE) with the MBSE Initiative[1], MBSE is defined as “the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and

[1] http://www.omgwiki.org/MBSE/doku.php (visited on 31/05/2018).

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 219–229, 2019.
https://doi.org/10.1007/978-3-030-04209-7_18

continuing throughout development and later life cycle phases". However, adoption of MBSE takes time, as many inhibitors remain such as cultural and general resistance to change, lack of related Knowledge and Know-How (K&KH), or the need to higher degree of guidance and reuse.

For a wider MBSE adoption, several advances seem to be necessary concerning organizational, methodological, and tools perspectives. In particular, from a methodological point of view, reuse seems to be promising. Reusing engineer's Knowledge and Know-How is an act of capitalization on previous experiences or projects, whether it is on the System Of Interest (SOI) or on the Systems Engineering Activities (SEA). But, often, those data are kept in their mind, and works have to be done to formalize them, with the goal of sharing them so they can be reused. The expected benefits make the assumption that reused modelling artefacts satisfy some maturity criteria to grant that they have reached a level of quality compatible with reuse objectives.

This article reviews engineering practices which intent to capitalize on K&KH, and to facilitate information sharing and reuse. A focus is made on reusing K&KH through the concept of "pattern". In this way, the second section presents reuse challenges in engineering and related works, the third a short history on patterns, the fourth a literature review of pattern for SE, and the last section discusses on the interest of using patterns in MBSE.

2 Challenges and Related Works

The fundamental difference between knowledge and know-how is that knowledge provides only solutions and answers to problems and questions, whereas know-how provides solutions but also a manner to construct these solutions (Gzara et al. 2003). Thereby, engineer's know-how is built from their experience, allowing them to reuse information gathered in order to be more efficient in their tasks. However, those "archives" are stuck in engineers' mind, making it difficult to share them to someone else to foster reuse (Mourtzis et al. 2016; Demian and Fruchter 2006). Yet, dynamic information flowing among engineering teams is a critical challenge for many companies who need to manage complex systems as information must be shared, comprehensive, and coherent among the project (Miled 2014). This aspect is very important as it allows a better comprehension of the SOI and SEA. For example in requirement engineering, Darimont et al. (2017) presents the results of a survey where 55–60% of engineers use "copy & paste" and "duplication" techniques, and only 13% use "requirements patterns catalogue". As the complexity, the quantity of K&KH and also engineering artefacts are exploding, those practices are no longer sufficient to answer challenges of nowadays complex system development. That is why there is a need to promote efficient way to transfer K&KH, in order to facilitate their circulation and reuse, and this is why current expectations are to promote models over natural language and its variability of understanding.

Research works have already been done for reuse K&KH in SE (Bollinger and Evins 2015; Barter 1998; Cloutier 2008; Cook and Schindel 2017; Gautam et al. 2007; Gzara et al. 2003; Haskins 2005; Korff 2013; Wang et al. 2010), but as there are many different ways to capitalize K&KH, it is important to define the targeted perimeter or the engineering artefacts before considering reusing. Indeed, "reuse" activities in SE can be distinguished in three different approaches: *Opportunistic reuse*: when the first project was not developed with reusable capacity; *Planned reuse*: when the first project was developed with reusable capacity; *Variance*: on a product line, common core model but different options. Those approaches belong to the process of "knowledge transfer" which consists of two sub-processes defined by Majchrzak et al. (2004). On the one hand, the process by which an entity's K&KH is captured, called "knowledge sharing", and on the other hand, the process by which an entity is able to locate and to use K&KH, called "knowledge reuse". It is important to ensure that the engineering artefact on which a reuse solution is applied may be the SOI or SEA (Pfister et al. 2012).

Within existing reuse approaches for the SOI, the use of Components Off The Shelves (COTS) consists in breaking down a problem into solvable sub-problems by already existing components. However, the advantages of COTS are accompanied by integration issues, early identified by Boehm and Abts (1999) which are: functionality and performance (what it is expected to do), interoperability (no standard exists), product evolution (risk of no longer meeting the need) and vendor behaviour (false promises). For the reuse in SEA which aim to produce the SOI, Darimont et al. (2017) deployed a local reuse library for each engineers and a shared reuse library for improving reuse of requirements across projects, and Majchrzak et al. (2004) identifies a six-stage reuse-for-innovation process, with the capacity to quickly capture and present information on potentially reusable ideas. Other works both addressed the SOI and SEA, such as the extension of the Constructive Systems Engineering Cost Model (COSYSMO) by Wang et al. (2010) that consists in defining reuse categories and weights for each of the category. While COSYSMO considers the whole, the PABRE approach focuses on requirements management (Palomares et al. 2014) and is based on a metamodel of Software Requirements Patterns (SRP), a method of reuse, a catalogue of 111 SRP, and a software tool that supports the management and the use of the catalogue.

As shown, many research works are looking to reuse K&KH to improve engineering efficiency. One way that looks particularly promising is achieved through the adoption of patterns, for both SEA and SOI, to systematize complex systems engineering (Cochard 2017). As they can be used in all stages of the development cycle (Gzara et al. 2003), reuse of patterns seems to be a very suitable form of reuse (Schindel 2005).

3 A Little History of Patterns

Most people in the pattern community attribute the first promoter of the value of "pattern" to Alexander et al. (1977) in a book on architecture, urban design and community liveability. They formalized a "pattern language", made of a myriad of patterns that helped them to express design in terms of relationships between the parts of a house, and the rules to transform those relationships (Coplien 1997). They began to identify patterns with the idea that "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander et al. 1977). The same way engineers reuse their knowledge based on their previous experience, Cloutier (2006) point out that Alexander and his co-authors "did not invent these patterns, they came from observation and testing; and only then were they documented as patterns".

Since these pioneer works, the pattern approach has been introduced in various engineering fields such as Software, Requirements, Telecommunications and Control Systems Engineering (Cloutier 2006). Beck and Cunningham (1987) were the first to propose object-oriented patterns in the Software community. The goal was to improve quality and to facilitate code writing by adopting good practices. Gamma et al. (1995), also known as the "Gang of Four", wrote an authoritative book describing 23 Software Design Patterns such as Composite, Iterator, Command… A Design Pattern is a general, reusable solution to a recurring problem in the design of object-oriented applications; it describes a proven solution for solving software architecture problems. As Design Patterns are not a finished design (concrete algorithm), but a structured description of computer programming, it means they are independent from programming languages. Design Patterns have been widely accepted, and encouraged other domains to write patterns to capture their experience.

In the field of SE, the value of patterns appears towards the growing complexity of systems and the difficulty to capture large body of knowledge. That is why Barter (1998) proposes the creation of a Systems Engineering Pattern Language, which is a collection of patterns that, when combined, address problems larger than the problems that an individual pattern can address. In the same way, Haskins (2003) proposes the use of SE patterns to capture the information in the Systems Engineering Body of Knowledge (SEBOK). Other works have used the concept of pattern in SE, especially in the Product Information System field, where Cauvet et al. (1998), Gzara (2000) propose a methodological framework based on the reuse of patterns during all the lifecycle, or Conte et al. (2001) who proposed patterns libraries to support a methodological framework for the conception of product information system.

After this short history of patterns, the next section aims at improving the comprehension of what is a pattern in SE.

4 Patterns for Systems Engineering

It happens that similar designs are made independently by different engineers (Gaffar and Moha 2005). This phenomenon acknowledges the fact that the same design elements exist in multiple designs, and the study and documentation of such designs foster reuse among projects. Indeed, it prevents from "reinventing the wheel" and provides a vocabulary for the design concepts that projects can share. This is consistent with the notion that patterns "are not created from a blank page; they are *mined*" (Hanmer and Kocan 2004). It appears that SE patterns are embedded in existing designs, and that it is necessary to find a mechanism to identify them. Those patterns are called "buried patterns" by Pfister et al. (2012) and represent a scientific issue. As the process of "Mining" appears to be essential for creating Pattern Languages, various approaches have been identified to write patterns from the element extracted from pattern mining. According to DeLano (1998) classification, it is possible to classify mining's processes into three categories: *Individual contributions* where writers of the pattern used their own experiences or ones from their colleagues; *Second-hand contributions* where patterns are written based on interviews with experts or by guiding another person in the writing of patterns, it can also come from borrowing patterns from the literature or from companies in the same domain; *Workshops/Meeting contributions* that consists of groups of around ten people who brainstorm the elements of a patterns, along with a moderator and a facilitator.

When mining a pattern, depending on the language used (textual or modelling), it appears that a minimal set of information is always provided, as a pattern seems to possess an inherent triptych composed of {Context, Problem, Solution}. Gaffar and Moha (2005) define a "Minimal Triangle" that defines the core meaning of a pattern (Fig. 1). It summarizes the idea that a pattern provides a solution to a recurring problem in a particular context. However, a general consensus enlarges the minimal elements needed in a pattern, Barter (1998) describe a generic pattern with the minimal elements needed to be written (Fig. 2). Cloutier and Verma (2007) conduct a survey that allow them to list a recommended Systems Pattern Form. They also underline the fact that concepts used in Systems Engineering represent higher levels of complexity and abstraction that the prevailing notions of Alexander in architecture. For instance, the architecture of the underlying concepts of control-command requires a more complex notation than the sketch used in Alexander et al. (1977), thus Pfister et al. (2012) used the Enhanced Functional Flow Block Diagram (eFFBD) to represent the model of their control-command and rely on formal conceptual foundations in the form of a meta-model.

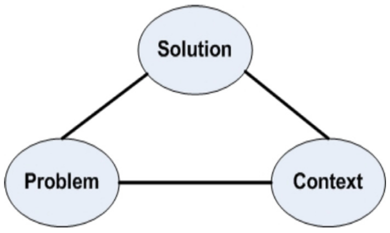


Fig. 1. The minimal triangle, extracted from Gaffar and Moha (2005)

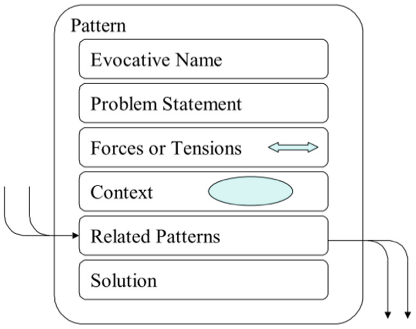


Fig. 2. Generic pattern, extracted from Barter (1998)

Like models, patterns are abstractions or a set of abstractions of the reality and not a magical solution. They allow people to solve complex problems by leveraging experience, K&KH from their predecessors. The results of a study conducted, in the Open Source Software community, by Hahsler (2005) show that the larger the team size was, the greater the use of patterns was for documenting changes: from 11.4% for a unique developer to 82.2% in a team of ten or more developer. The capacity of patterns to deliver at each level of the development the correct amount of information for the stage it is applied, allow its quick adoption and most importantly its active use as Hahsler concludes in his study: “design patterns are adopted for documenting changes and thus for communication in practice by many of the most active open source developers”. Patterns offer the possibility to create a common lexicon between systems architects that foster a common understanding of systems architecture, validated by experts. In this way, the experience acquired by the software community on pattern will be valuable, and help systems engineers to walk in their footsteps in order to develop patterns that will foster reuse, as well as helping control the complexity of a system.

As the interest for MBSE increases, it is important to also examine the work done for integrating the concept of pattern in this framework. The integration of the OMG System Modelling Language (OMG SysML) and its consequences on how to define problems and describe solutions are particularly interesting and will be examined in the next section.

5 Patterns for Model-Based Systems Engineering (MBSE)

Although research works have been made to assess whether the concept of pattern can be applied in the Systems Engineering field such as Pfister et al. (2012), Cloutier (2006), Haskins (2005), the value of patterns in a MBSE framework has not been fully explored. Yet, it appears crucial to consider all the different needs, requirements and constraints of the different stakeholders in the early design stages. Perceived by many companies as a time loss, it appears that introducing or reinforcing reuse capacity in MBSE methodologies allows the design of a new project with much less human effort, benefiting from the reuse of the already existing system models (Shani and Broodney 2015). In this way, the capitalization and reuse of system models through the concept of pattern can be implemented in MBSE, and thus, favour its adoption at a larger scale.

Models are abstraction or a set of abstractions of the reality (i.e. the reality can be represented under different consistent views), which means that it can be easy to reuse a model in a new project since no physical limitations get in the way. However, depending on the type of reuse to do, the complexity of the system under design, and also the heterogeneity of methodologies and tools, it appears that the adoption of MBSE is penalized. Indeed, reusing existing modelling artefacts (even if their designs have been made to be reusable) is harder than expected. As Korff (2013) stated, the “biggest problem is to transfer and manage the knowledge [of] what is actually available for re-use”. He emphasizes on the fact that it is necessary for system engineers to be aware of system assets that can be defined and propagated among teams designing complex systems. However, the creation of assets library is not sufficient, as the purpose is to allow engineers to reuse those assets in their ongoing projects. Korff underlines the fact that users should have the possibility to search, publish, and reuse assets in defined libraries and catalogues, without any specific technical pre-requisite. Contrary to Korff (2013), Paydar and Kahani (2015) do not focus on the creation of assets but propose an approach concerning the adaptation of promising reusable assets during a model reuse process, especially on the adaptation of OMG Unified Modeling Language (OMG UML) activity diagrams to new use cases, in the context of web engineering. This work proposes to semi-automatically create an activity diagram from existing activity diagrams according to the input use case diagram. Even though this approach is not presented in a MBSE framework, the fact that between OMG UML and OMG SysML, use case diagrams are identical and that activity diagrams presents the same use, allows considering a transposition in the SE field.

In the case of variant modelling in MBSE, Oster et al. (2016) propose an approach for building and exploiting composable architectures to the design and development of a product line of complex systems in the aerospace and defence market. They choose OMG SysML as the core language to define descriptive models of the composable system reference architecture and extended it to define parametric models. This methodology allowed the product line to evolve more readily as the impact of information propagation of adding, updating or modifying new components was automatic. As their works consider physical layer, Di Maio et al. (2016) focus their attention on the development of a functional architectures that can accommodate to change due to decisions made in the logical layer for System of Systems (SoS). The results of their study are a MBSE process that consists in the integration of a system model before the consideration of the variants. It requires that the system model should contain both the original configuration and the variant one. This separation is important in case of a new technology is introduced but the older one are not abandoned yet. They also investigate the aspects of including variant modelling into the OMG SysML, with a focus on extending an existing and operating model to support a new variant in the case where a similar technology is used.

The introduction of a reuse capacity in MBSE frameworks has proven to improve engineering efficiency in engineers work. However, the steep learning curve induced for organizations to adopt MBSE methodologies, results in the need of helping the engineers to "quickly identify not only valid architectural solutions, but optimal value solutions for the mission need" (Oster et al. 2016). Thus, it appears that the concept of patterns could be an answer to this challenge. Indeed, works have been done to introduce patterns during various phases of the engineering cycles. Gasser (2012) described behavioural construct patterns (Fig. 3) to facilitate and systematize the modelling of system behaviour. Instead of thinking at the level of atomic graphical elements, he defined a structured way to represent elementary behavioural constructs. In this way, he advocates the use of an "insert policy", like in the construction of Functional Flow Block Diagram (FFBD) where the resizing of the diagram is automatic

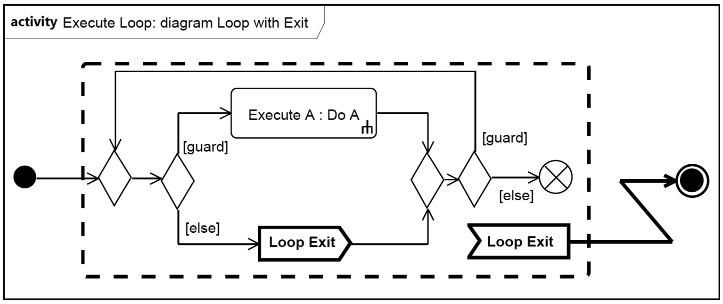


Fig. 3. Loop exit construct, extracted from Gasser (2012)

when new elements are inserted. The proposed behavioural construct patterns will allow engineers to work in an algorithmic way of thinking, which implies a higher modelling level that will permit to focus more on the expected behaviour than on the aesthetics of the diagrams.

In order to help engineers to focus on what is important, patterns should guide the development to avoid deviation. For example, Barbieri et al. (2014) proposed a process for the development of mechatronic systems based on a SysML design pattern. Their intent is to demonstrate that adequate guidelines for modelling can benefits the development process by allowing an efficient traceability of all information within the system model to trace change influences more easily. This approach proves to be particularly helpful for facilitating the impact analysis in later lifecycle phases and for the reuse for future projects.

Pursuing the work of Haskins (2005) on patterns, Schindel (2005) proposed an engineering paradigm where patterns are re-usable models, that enables what he calls Pattern-Based Systems Engineering (PBSE), where patterns can be configured or specialized into product lines or into product systems. With the advent of MBSE, this modelling framework has led to the creation of an INCOSE working group called MBSE Patterns[2]. In this context, Schindel and Peterson (2013) developed their approach, they see "patterns as re-usable models" and apply them to requirements and design. At a high-level, they constitute a generic system pattern model that can be customized according to enterprise needs, configuration, uses, so that engineers can benefit from the concepts of MBSE without being an expert of modelling methodologies. Cook and Schindel (2017) applies it for the Verification and Validation processes, and Bradley et al. (2010) in the pharmaceutical market.

6 Conclusion

As presented in the introduction, a main issue for system engineers is to shorten engineering cycle period, and MBSE appears to be a great candidate to face this challenge. For a wider MBSE adoption, this paper highlights the strong methodological need to capitalize on previous projects to reuse K&KH, and focuses on the concept of pattern, which offers the possibility to make information dynamic between stakeholders during the development of complex systems, in order to share it and foster its reuse for future MBSE projects.

From a methodological perspective, improvements from processes, methods and tools should be made. It appears that the act of capitalization is not self-evident, as patterns need to be mined, and imply the ability to detect and bring out K&KH. A first step is to evaluate the maturity of such capitalized patterns, as done in the automated production systems domain by Vogel-Heuser et al. (2018) on the maturity on control modules in libraries. A second step is to improve the general maturity of reuse approaches as done in the software domain by Manzoni and Price (2003), using for example metrics inspired by Capability Maturity Model. A next step to improve

[2] http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns (visited on 31/05/2018).

engineering effectiveness concerns the development and the adoption of MBSE software tools that integrate patterns libraries supporting their capitalization, selection, reuse, and update.

References

Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Ch. Alexander (1977). https://doi.org/10.2307/1574526

Barbieri, G., Kernschmidt, K., Fantuzzi, C., Vogel-Heuser, B.: A SysML based design pattern for the high-level development of mechatronic systems to enhance re-usability. In: IFAC Proceedings Volumes (IFAC-PapersOnline), vol. 19, pp. 3431–3437. IFAC (2014). https://doi.org/10.3182/20140824-6-za-1003.00615

Barter, R.H.: A systems engineering pattern language. In: INCOSE, pp. 350–353 (1998)

Beck, K., Cunningham, W.: Using pattern languages for object-oriented programs. In: OOPSLA-87 Workshop on the Specification and Design for Object-Oriented Programming (1987)

Boehm, B., Abts, C.: COTS integration: plug and pray? Computer **32**(1), 135–138 (1999). https://doi.org/10.1109/2.738311

Bollinger, L.A., Evins, R.: Facilitating model reuse and integration in an urban energy simulation platform. Procedia Comput. Sci. **51**(1), 2127–2136 (2015). https://doi.org/10.1016/j.procs.2015.05.484

Bradley, J.L., Hughes, M.T., Schindel, W.: Optimizing delivery of global pharmaceutical packaging solutions, using systems engineering patterns. In: 20th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2010, vol. 3, pp. 2441–2447 (2010). https://doi.org/10.1002/j.2334-5837.2010.tb01175.x

Cauvet, C., Rieu, D., Espinasse, B., Giraudin, J.-P., Tollenaere, M.: Ingénierie Des Systèmes d'information Produit: Une Approche Méthodologique Centrée Réutilisation de Patrons. In: Inforsid, pp. 71–90 (1998). http://dblp.uni-trier.de/db/conf/inforsid/inforsid1998.html#CauvetREGT98

Cloutier, R.J.: Applicability of Patterns to Architecting Complex Systems, vol. 466. Stevens Institute of Technology, Hoboken (2006)

Cloutier, R.J.: Model driven architecture for systems engineering. In: Language, no. September (2008). http://personal.stevens.edu/~pkorfiat/CONOPS/Research/1_018.pdf

Cloutier, R.J., Verma, D.: Applying the concept of patterns to systems architecture. Syst. Eng. **10**(2), 138–154 (2007). https://doi.org/10.1002/sys.20066

Cochard, T.: Contribution à La Génération de Séquences Pour La Conduite de Systèmes Complexes Critiques (2017)

Conte, A., Fredj, M., Giraudin, J.-P., Rieu, D.: P-Sigma: Un Formalisme Pour Une Représentation Unifiée de Patrons. In: *XIXème Congrès INFORSID*, no. January, pp. 67–86 (2001). http://liris.cnrs.fr/inforsid/sites/default/files/a366c1YfHw5cvgN2I.pdf

Cook, D., Schindel, W.: Utilizing MBSE patterns to accelerate system verification. Insight **20**(1), 32–41 (2017). https://doi.org/10.1002/inst.12142

Coplien, J.O.: Idioms and patterns as architectural literature. IEEE Softw. **14**(1), 36–42 (1997). https://doi.org/10.1109/52.566426

Darimont, R., Zhao, W., Ponsard, C., Michot, A.: Deploying a template and pattern library for improved reuse of requirements across projects. In: Proceedings—2017 IEEE 25th International Requirements Engineering Conference, RE 2017, pp. 456–457 (2017). https://doi.org/10.1109/re.2017.44

DeLano, D.E.: Patterns Mining. In: Rising, L. (ed.) The Pattern Handbook: Techniques, Strategies, and Applications, pp. 87–96. Cambridge University Press, New York (1998)

Demian, P., Fruchter, R.: An ethnographic study of design knowledge reuse in the architecture, engineering, and construction industry. Res. Eng. Des. **16**(4), 184–195 (2006). https://doi.org/10.1007/s00163-006-0010-x

Estefan, J.A.: Survey of model-based systems engineering (MBSE) methodologies. In: INCOSE MBSE Initiative (2008). https://doi.org/10.1109/35.295942

Gaffar, A., Moha, N.: Semantics of a pattern system. In: Proceedings of the STEP International Workshop on Design Pattern Theory and Practice IWDPTP05 (2005)

Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co. Inc., Boston (1995)

Gasser, L.: Structuring activity diagrams. In: 14th IFAC Symposium on Information Control Problems in Manufacturing, Bucharest, Romania. IFAC (2012). https://doi.org/10.3182/20120523-3-ro-2023.00153

Gautam, N., Chinnam, R.B., Singh, N.: Design reuse framework: a perspective for lean development. Int. J. Prod. Dev. **4**(5), 485–507 (2007). https://doi.org/10.1504/IJPD.2007.013044

Gzara, L.: Résumé - Les Patterns Pour l'Ingénierie Des Systèmes d'Informations Produit (2000)

Gzara, L., Rieu, D., Tollenaere, M.: Product information systems engineering: an approach for building product models by reuse of patterns. Robot. Comput. Integr. Manuf. **19**(3), 239–261 (2003). https://doi.org/10.1016/S0736-5845(03)00028-0

Hahsler, M.: A quantitative study of the adoption of design patterns by open source software developers. In: Free/Open Source Software Development, pp. 103–124. IGI Global (2005)

Hanmer, R.S., Kocan, K.F.: Documenting architectures with patterns. Bell Labs Tech. J. **9**(1), 143–163 (2004). https://doi.org/10.1002/bltj.20010

Haskins, C.: 1.1.2 using patterns to share best results - a proposal to codify the SEBOK. In: INCOSE International Symposium, vol. 13, no. 1, pp. 15–23 (2003). https://doi.org/10.1002/j.2334-5837.2003.tb02596.x

Haskins, C.: Application of patterns and pattern languages to systems engineering. In: 15th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2005, vol. 2, pp. 1619–1627 (2005). http://www.scopus.com/inward/record.url?eid=2-s2.0-84883318751&partnerID=tZOtx3y1

Korff, A.: Re-using SysML system architectures. In: Complex Systems Design and Management—Proceedings of the 4th International Conference on Complex Systems Design and Management, pp. 257–266. Springer, Berlin (2013). https://doi.org/10.1007/978-3-319-02812-5-19

Di Maio, M., Kapos, G.D., Klusmann, N., Allen, C.: Challenges in the modelling of SoS design alternatives with MBSE. In: 2016 11th Systems of Systems Engineering Conference, SoSE (2016). https://doi.org/10.1109/sysose.2016.7542937

Majchrzak, A., Cooper, L.P., Neece, O.E.: Knowledge reuse for innovation. Manage. Sci. **50**(2), 174–188 (2004). https://doi.org/10.1287/mnsc.1030.0116

Manzoni, L.V., Price, R.T.: Identifying extensions required by RUP (Rational Unified Process) to comply with CMM (Capability Maturity Model) levels 2 and 3. IEEE Trans. Softw. Eng. **29**(2), 181–192 (2003). https://doi.org/10.1109/TSE.2003.1178058

Miled, A.B.: Reusing knowledge based on ontology and organizational model. Procedia Comput. Sci. **35**, 766–775 (2014). https://doi.org/10.1016/j.procs.2014.08.159

Mourtzis, D., Doukas, M., Giannoulis, C.: An inference-based knowledge reuse framework for historical product and production information retrieval. Procedia CIRP **41**, 472–477 (2016). https://doi.org/10.1016/j.procir.2015.12.026

Oster, C., Kaiser, M., Kruse, J., Wade, J., Cloutier, R.: Applying composable architectures to the design and development of a product line of complex systems. Syst. Eng. **19**(6), 522–534 (2016). https://doi.org/10.1002/sys.21373

Palomares, C., Quer, C., Franch, X.: Requirements reuse with the PABRE framework. Requir. Eng. Mag. **2014**, 1 (2014)

Paydar, S., Kahani, M.: A semi-automated approach to adapt activity diagrams for new use cases. Inf. Softw. Technol. **57**(1), 543–570 (2015). https://doi.org/10.1016/j.infsof.2014.06.007

Pfister, F., Chapurlat, V., Huchard, M., Nebut, C., Wippler, J.-L.: A proposed meta-model for formalizing systems engineering knowledge, based on functional architecture patterns. Syst. Eng. **15**(3), 321–332 (2012). https://doi.org/10.1002/sys.21204

Schindel, W.: Requirements statements are transfer functions: an insight from model-based systems engineering. In: INCOSE International Symposium, vol. 15, no. 1, pp. 1604–1618 (2005). https://doi.org/10.1002/j.2334-5837.2005.tb00775.x

Schindel, W., Peterson, T.: Introduction to pattern-based systems engineering (PBSE): leveraging MBSE techniques. In: INCOSE International Symposium, vol. 23, no. 1, p. 1639 (2013). https://doi.org/10.1002/j.2334-5837.2013.tb03127.x

Shani, U., Broodney, H.: Reuse in model-based systems engineering. In: 9th Annual IEEE International Systems Conference, SysCon 2015 - Proceedings, pp. 77–83 (2015). https://doi.org/10.1109/syscon.2015.7116732

Vogel-Heuser, B., Fischer, J., Neumann, E.-M., Diehm, S.: Key maturity indicators for module libraries for PLC-based control software in the domain of automated production systems. In: 16th IFAC Symposium on Information Control Problems in Manufacturing (2018)

Wang, G., Valerdi, R., Fortune, J.: Reuse in systems engineering. IEEE Syst. J. **4**(3), 376–384 (2010). https://doi.org/10.1109/JSYST.2010.2051748

Posters

The Systems Engineering Concept

A Practical Hands-on Approach to Systems Engineering

Henrik Balslev(✉)

Certified Systems Engineering Professional (CSEP), Systems Engineering A/S, Livjaegergade 17B, 2nd Floor, 2100 Copenhagen Oe., Denmark
hb@syseng.dk

Abstract. Systems engineering includes a range of different processes defined by ISO 15288, which one can run depending on the requested needs. But ISO 15288 does not provide specific detailed instructions for the execution of these, and very often systems engineering will be performed by specialists responsible for systems engineering activities as an add-on to other project activities, rather as an integral part of the daily life of designers and engineers.

This white paper introduces a concept for systems engineering, The Systems Engineering Concept® (SEC), which can be used by all project participants and not just specialists. SEC is using selected processed from ISO 15288 and in addition IEC/ISO 81346, IEC 61355 and IEC 62023 standard series. Processes are determined by making reverse engineering on proven methods and use these in SEC.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 233–233, 2019.
https://doi.org/10.1007/978-3-030-04209-7_19

From Document Centric Approach to MBSE Approach: BPMN, UML, SysML and Wire Framing Implementation

David Schumacher(✉)

Thales Services, 110 rue Blaise Pascal, Immeuble Viséo,
38334 Montbonnot Cedex, France
david.schumacher@thalesgroup.com

Abstract. Model Based System Engineering (MBSE) approach aims among other to achieve the business objectives for complex/critical systems. This approach is so intended to enhance understanding and quality. Several choices shall be made and shall allow to build a good model which means that is usable and understandable by all stakeholders in order to share and support complex system requirements, business analysis, architecture, design, and verification & validation activities.

To maximize number and various stakeholders, we though language(s) have to be rightly selected.

So, we will discuss an example of a framework implementation based on: BPMN, UML, SysML and wire framing in order to tempt to improve of collaboration, to improve impact assessments and to share the understanding in product development between teams for two different project: IT project (web software context) and a class III medical critical system (hardware and software context).

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 234–234, 2019.
https://doi.org/10.1007/978-3-030-04209-7_20

Towards a Better Modelling and Assessment of Project Management Maturity in Industry 4.0

Felipe Sanchez[1](✉), Davy Monticolo[2], Eric Bonjour[2], and Jean-Pierre Micaëlli[3]

[1] Sopra Steria, Université de Lorraine, ERPI, 5 Place de L'Iris, 92400 Courbevoie, France
felipe.sanchez@soprasteria.com

[2] Université de Lorraine, ERPI, 8 Rue Bastien-Lepage, Nancy 54000, France
{davy.monticolo,eric.bonjour}@univ-lorraine.fr

[3] Jean Moulin Lyon 3 University, IAE Lyon School of Management, 1C avenue des frères Lumière, CS 78242, 69372 Lyon Cedex 08, France
jean-pierre.micaelli@univ-lyon3.fr

Abstract. Companies are currently facing substantial challenges with regard to Industry 4.0. In order to adapt to this changing environment, companies are moving from operations-centered business to project-driven business. This change requires an evolution in project management. Researchers and practitioners, inspired by the PMBOK (Project Management Body of Knowledge), have created maturity models to compare and evaluate organizations, but they did not specify any methodology to create adapted models to face this technological change. Therefore, this paper proposes an approach to understand under on principles existing project management maturity models were based, and how it is possible to create a new project management maturity model applicable in the emerging framework of industry 4.0. Then, we illustrate the new approach with the construction of a project maturity model used to measure the planning capability. Finally, we define limitations of the model and future research directions.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 235–235, 2019.
https://doi.org/10.1007/978-3-030-04209-7_21

Integrated Framework for Design and Testing of Software for Automotive Mechatronic Systems

Nick Van Kelecom[1](✉), Timothy Verstraete[2], Sam Silverans[1], and Mathieu Dutré[1]

[1] Siemens Industry Software NV, Interleuvenlaan 68, 3001 Louvain, Belgium
{nick.van_kelecom, sam.silverans, mathieu.dutre}@siemens.com

[2] University of Antwerp, Groenenborgerlaan 171, 2020 Antwerp, Belgium
Timothy.Verstraete@uantwerpen.be

Abstract. Today's electrification and automation put a lot of pressure on automotive OEM's. Vehicle functions are increasingly executed by complex mechatronic systems, feeding the need for high-efficiency multi-disciplinary teams. Numerous car recalls confirm that establishing these teams is an unobvious process. A framework facilitating design, testing, engineering asset management and planning is essential. This research presents such an envisioned framework with a central development management tool, integrated with different engineering tools to obtain vital information.

First, high level requirements are defined, whereof a software architecture can be deduced. Based on this architecture, requirements can be remapped, refined and implemented. In parallel, test cases are written, verifying a requirement's implementation. Automatic test harness generation will reduce repetitive, time-intensive modelling. The testing status can later be centrally monitored. This status is the initiator for new engineering tasks, managed in a Kanban board providing a clear overview. Complete traceability through this process is ensured.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 236–236, 2019.
https://doi.org/10.1007/978-3-030-04209-7_22

Complex Systems Engineering Approach for Condition Monitoring for the Digital Transformation: Integration into Mining Industry Control Systems

Mariya Guerroum[1,2](✉), Ali El-Alaoui[1,3], Laurent Deshayes[1], Mourad Zegrari[1,2], Janah Saadi[3], and Hicham Medromi[3]

[1] Innovation Lab for Operations, Mohammed VI Polytechnic University, Ben Guerir, Morocco
{Mariya.guerroum,ali.elalaoui}@um6p.ma
[2] Ecole Nationale Supérieure des Arts et Métiers de Casablanca, Casablanca, Morocco
[3] Ecole Nationale Supérieure de l'Electricité et de la Mécanique, Casablanca, Morocco

Abstract. The digital transformation of the Mining Industry is about to affect all organizational levels, from manufacturing to maintenance. This revolution would be impossible without a modern Information and Communication Technologies (ICT) infrastructure. Maintenance management is a complex process requiring an effective combination of technical and economic expertise. This paper presents electromechanical systems condition-monitoring architecture, using the Systems Engineering Approach. The main function of the system is to extract and to identify physical parameters of the studied system for predictive maintenance strategy elaboration. The System's architecture based on the operational, functional and constructional visions is fundamental to define the action scope and its features, leading to avoid production breakdowns, to improve maintenance management and to minimize the related intervention costs. We used SysML diagrams to model the solution and hence to materialize the System targeting maintenance in industrial environment.

Cyber Physical Systems Real Time and Interactive Testing and Governance

Sara Sadvandi[1](✉), Franck Corbier[2], and Eric Mevel[3]

[1] Dassault Systèmes, 10 rue Marcel Dassault, 92940 Velizy, France
Sara.sadvandi@3ds.com
[2] Dassault Systèmes, 35 rue Haroun Tazieff, 54320 Maxeville, France
Franck.corbier@3ds.com
[3] Dassault Systèmes, 120 rue René Descartes, 29280 Plouzané, France
Eric.mebel@3ds.com

Abstract. Cyber Physical Systems (CPS) interconnects the cyber world of communication and computing with the physical via reliable and secure software's. It asserts a critical challenge not only on development of complex systems but also on integration and validation of system of systems (SoS). This article develops a categorization of multiple levels of testing and defines a high level conceptual organization of test based engineering and validation. It introduces a real time and interactive co-execution platform that provides heterogeneous model integration, models validation and monitoring. It presents a generative approach for test variants management to assure dynamic changes and the flexibility in execution and test during the project life cycle. Further, it provides effective deployment domains.

Keywords: Cyber Physical Systems · Model-based testing · MiL SiL · HiL · Progressive integration and validation · Test variant management Test governance · System under tests · Real time and interactive execution Test scenarios

Machine-Executable Model-Based Systems Engineering with Graph-Based Design Languages

Benedikt Walter[1](✉), Dennis Kaiser[2], and Stephan Rudolph[3](✉)

[1] R&D MB Cars, Daimler AG, Sindelfingen, Germany
benedikt.walter@daimler.com
[2] IILS mbH, Leinfelden, Germany
kaiser@iils.de
[3] University of Stuttgart, IFB, Stuttgart, Germany
rudolph@ifb.uni-stuttgart.de

Abstract. Model-Based Systems Engineering (MBSE) structures the design process in form of a V-Model. Along the V-Model, the tasks of model creation, editing and design change propagation to maintain model consistency requires manually much time, effort and cost. While formal languages such as the Unified Modeling Language (UML) or the Systems Modeling Language (SysML) are used already to represent the design process and behavioral product aspects, the UML/SysML-models are mostly still manually assembled and interlinked. Graph-based design languages on the basis of UML combine the advantage of a digital representation of the design process with the advantage of a rule-based execution. This combination of digital representation with machine-execution boosts MBSE towards a repeatable and machine-executable V-Model. This allows for a seamless transition from manual MBSE towards an automated and machine-executable MBSE. The poster will highlight some of the advantages of this fully digital and machine-executable MBSE in an automotive dashboard application.

https://doi.org/10.1007/978-3-030-04209-7_25

Cyber-Physical System Modeling Using a Case Study

Sara Mallah[1,2](✉), Khalid Kouiss[3], Oualid Kamach[1,2,3], and Laurent Deshayes[1,2,3]

[1] ILO, University Mohammed 6 Polytechnic, Ben Guerir, Morocco
sara.mallah@um6p.ma
[2] LTI, ENSA, Tangier, Morocco
[3] SIL, Blaise-Pacal University of Clermont Ferrand, Aubière, France
khalid.kouiss@ifma.fr

Abstract. The fad in today's market for customer-specific products pushed the industry to renew itself and drive value creation initiative. In fact, companies are concerned not only about selling the product as a function, but also about selling the value as a solution. It is reasonable to think that the creation of these new business models involve building flexible manufacturing facilities, digitizing and integrating inter and intra-company systems into one intelligent data management structure which allow physical and software components to interact with each other in a myriad of ways that change with context, in spite of the different spatial and temporal scales they operate on. This synergic interaction can be fulfilled by accomplishing an industry 4.0 environment that aims to transcend mechatronic systems and move to cyber-physical systems (CPS).

In this paper, we present our methodology to model CPS. The results show promising research opportunity for implementing CPS in industry.

https://doi.org/10.1007/978-3-030-04209-7_26

The SERC 5-Year Technical Plan: Designing the Future of Systems Engineering Research

Jon Wade[1(✉)], Dinesh Verma[1], Thomas McDermott[1], and Barry Boehm[2]

[1] Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030, USA
{jon.wade,dinesh.verma,tmcdermo}@stevens.edu
[2] University of Southern California, Los Angeles, USA
boehm@usc.edu

Abstract. The Systems Engineering Research Center (SERC), a US University Affiliated Research Center, developed the 2014–2018 Technical Plan to provide the vehicle by which to align the SERC Vision and Research Strategy with the US Federal Government Sponsor's top research priorities. This paper summarizes the SERC Vision, the Sponsor's needs, and the SERC's response to these needs. It then describes the objectives, approach and content of the original five-year SERC Technical Plan, and provides an overview of the results. Emerging systems challenges are noted along with the approach that is being used to address them in the upcoming Five Year Technical Plan. Finally, this paper describes the status of the new plan and some of the opportunities and challenges that it provides.

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 241–241, 2019.
https://doi.org/10.1007/978-3-030-04209-7_27

Understand Corporate Culture for a Better Steering Model

Paul Maitre, Jérôme Raby-Lemoine, and François Videau(✉)

Easis Consulting, 75 rue de la Fontaine au Roi, 75011 Paris, France
{pmaitre, jrabylemoine, fvideau}@easis-consulting.com

Abstract. Unpredictability being the key-word, continuous transformation is obviously mandatory for most organizations. Transformation programs and projects are launched at an ever-increasing pace.

Most of these transformation programs and projects are focused on Operations, IT systems, processes and indicators, and do not take into account collective subconscious, unspoken thoughts and hidden facts: Corporate culture.

More surprisingly, this finding is also true for Steering model improvement projects. Consulting firms and IT companies are designing and building new reporting processes, Balanced Scorecards or Business Intelligence softwares without even trying to understand what makes any organisation different one from another: its values, its skills, its habits and behaviors.

Based on experience (of hundreds of Transformation projects), our belief is that understanding Corporate culture within a meta-model analysis is a prerequisite for improving significantly the Steering model or to its components.

Correction to: Systemic Design Engineering

Curriculum and Instructional Results

Jon Wade, Steven Hoffenson, and Hortense Gerardo

Correction to:
Chapter "Systemic Design Engineering"
in: E. Bonjour et al. (Eds.):
***Complex Systems Design & Management*,**
https://doi.org/10.1007/978-3-030-04209-7_16

In the original version of the book, the spell error in third author name "Hortense Gerado" should be corrected as "Hortense Gerardo" in chapter "Systemic Design Engineering". The correction chapter and the book have been updated with the change.

The updated version of this chapter can be found at
https://doi.org/10.1007/978-3-030-04209-7_16

Author Index

E. Bonjour et al. (Eds.): CSD&M 2018, *Complex Systems Design & Management*, pp. 243–244, 2019.
https://doi.org/10.1007/978-3-030-04209-7