Boris Shishkov (Ed.)

LNBIP 109

Business Modeling and Software Design

First International Symposium, BMSD 2011
Sofia, Bulgaria, July 2011
Revised Selected Papers

Lecture Notes
in Business Information Processing 109

Series Editors

Boris Shishkov (Ed.)

Business Modeling and Software Design

First International Symposium, BMSD 2011
Sofia, Bulgaria, July 27-28, 2011
Revised Selected Papers

Springer

Volume Editor

Boris Shishkov
Interdisciplinary Institute for Collaboration and Research
on Enterprise Systems and Technology – IICREST
Sofia, Bulgaria
E-mail: b.b.shishkov@iicrest.eu

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-29787-8 e-ISBN 978-3-642-29788-5
DOI 10.1007/978-3-642-29788-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012935862

ACM Computing Classification (1998): J.1, H.4, H.3.5, D.2

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book contains extended and revised versions of a set of selected papers from the First International Symposium on Business Modeling and Software Design (BMSD 2011), held in Sofia, Bulgaria. The symposium was organized and sponsored by the Interdisciplinary Institute for Collaboration and Research on Enterprise Systems and Technology (IICREST), in cooperation with the Center for Telematics and Information Technology (CTIT), the Institute for Systems and Technologies of Information, Control and Communication (INSTICC), and the Technical University of Sofia, and was technically co-sponsored by Sofia Municipality and QlikTech Netherlands B.V.

The purpose of BMSD 2011 was to bring together researchers and practitioners interested in business modeling and its relation to software design. The theme of BMSD 2011 was: "Business Models and Advanced Software Systems," and the scientific areas of interest to the symposium were: (a) business models and requirements; (b) business models and services; (c) business models and software; (d) information systems architectures.

Building adequate business models is of huge importance not only for understanding and re-engineering an organization but also for automating (part of) its processes by means of software systems. Not grasping correctly and exhaustively a business system would inevitably lead to consequent software failures. BMSD 2011 addressed these challenges, by considering a large number of research topics: from more abstract ones, such as essential business models, to more technical ones, such as software specification, from more business-oriented ones, such as business process management and coordination, and requirements specification to IT architectures -related topics.

BMSD 2011 received 58 paper submissions from which 22 papers were selected for publication in the symposium proceedings. From these, 10 papers were selected for a 30-minute oral presentation (Full Papers), leading to a "full-paper" acceptance ratio of 17%; this shows the intention of preserving a high-quality forum for the next editions of this symposium. The 8 papers published in the current book were selected from the BMSD 2011 Full Papers. In all BMSD 2011 selections, a double-blind paper evaluation method was used: each paper was reviewed by at least two internationally known experts from the BMSD Program Committee.

The high quality of the BMSD 2011 program was enhanced by four keynote lectures, delivered by distinguished guests who are renowned experts in their fields, including (alphabetically): Mehmet Aksit (University of Twente, The Netherlands), Dimitar Christozov (American University in Bulgaria - Blagoevgrad, Bulgaria), Hermann Maurer (Graz University of Technology, Austria), and Bart Nieuwenhuis (University of Twente, The Netherlands). Their lectures

inspired the participants to gain a deeper understanding of the business modeling and software design fields.

We hope that you will find these papers interesting and consider them a helpful reference in the future when addressing any of the research areas mentioned above.

February 2012 Boris Shishkov

Symposium Committee

Chair

Boris Shishkov	IICREST, Bulgaria

BMSD Program Committee

Mehmet Aksit	University of Twente, The Netherlands
Antonia Albani	University of St. Gallen, Switzerland
Ognian Andreev	Technical University - Sofia, Bulgaria
Paulo Anita	Delft University of Technology, The Netherlands
Rumen Arnaudov	Technical University - Sofia, Bulgaria
Colin Atkinson	University of Mannheim, Germany
Csaba Boer	Tba, The Netherlands
Boyan Bontchev	Sofia University St. Kliment Ohridski, Bulgaria
Frances Brazier	Delft University of Technology, The Netherlands
Barrett Bryant	University of Alabama at Birmingham, USA
Cinzia Cappiello	Politecnico di Milano, Italy
Kuo-Ming Chao	Coventry University, UK
Ruzanna Chitchyan	Lancaster University, UK
Samuel Chong	Capgemini, UK
Dimitar Christozov	American University in Bulgaria, Bulgaria
Selim Ciraci	University of Twente, The Netherlands
José Cordeiro	Polytechnic Institute of Setúbal, Portugal
Dumitru Dan Burdescu	University of Craiova, Romania
Joop De Jong	Delft University of Technology, The Netherlands
Jan L.G. Dietz	Delft University of Technology, The Netherlands
Lyubka Doukovska	Bulgarian Academy of Sciences, Bulgaria
Joaquim Filipe	Polytechnic Institute of Setúbal, Portugal
Boris Fritscher	University of Lausanne, Switzerland
J. Paul Gibson	T&Msp - Telecom & Management Sudparis, France
Eduardo Goncalves Da Silva	University of Twente, The Netherlands
Rafael Gonzalez	Javeriana University, Colombia
Clever Ricardo Guareis De Farias	University of Sao Paulo, Brazil

Markus Helfert	Dublin City University, Ireland
Philip Huysmans	University of Antwerp, Belgium
Ilian Ilkov	IBM, The Netherlands
Ivan Ivanov	Suny Empire State College, USA
Dimitris Karagiannis	University of Vienna, Austria
Marite Kirikova	Riga Technical University, Latvia
Samuel Kounev	Karlsruhe Institute of Technology, Germany
Kecheng Liu	University of Reading, UK
Leszek Maciaszek	Macquarie University, Australia / University of Economics, Poland
Jelena Marincic	University of Twente, The Netherlands
Michele Missikoff	Institute for Systems Analysis and Computer Science, Italy
Dimitris Mitrakos	Aristotle University of Thessaloniki, Greece
Preslav Nakov	National University of Singapore, Singapore
Ricardo Neisse	University of Kaiserslautern, Germany
Bart Nieuwenhuis	University of Twente, The Netherlands
Selmin Nurcan	University Paris 1 Pantheon Sorbonne, France
Olga Ormandjieva	Concordia University, Canada
Robert Parhonyi	Inter Access, The Netherlands
Marcin Paprzycki	Polish Academy of Sciences, Poland
Oscar Pastor	Universidad Politécnica de Valencia, Spain
Erik Proper	Public Research Centre - Henri Tudor, Luxembourg
Jolita Ralyte	University of Geneva, Switzerland
Gil Regev	EPFL / Itecor, Switzerland
Ella Roubtsova	Open University, The Netherlands
Irina Rychkova	University Paris 1 Pantheon Sorbonne, France
Shazia Sadiq	University of Queensland, Australia
Brahmananda Sapkota	University of Twente, The Netherlands
Tony Shan	Keane Inc., USA
Kamran Sheikh	IBM, The Netherlands
Valery Sokolov	Yaroslavl State University, Russia
Richard Starmans	Utrecht University, The Netherlands
Cosmin Stoica Spahiu	University of Craiova, Romania
Coen Suurmond	RBK Group, The Netherlands
Bedir Tekinerdogan	Bilkent University, Turkey
Linda Terlouw	ICRIS B.V., The Netherlands
Yasar Tonta	Hacettepe University, Turkey
Roumiana Tsankova	Technical University - Sofia, Bulgaria
Marten van Sinderen	University of Twente, The Netherlands
Mladen Velev	Technical University - Sofia, Bulgaria

Kris Ven	University of Antwerpen, Belgium
Maria Virvou	University of Piraeus, Greece
Martijn Warnier	Delft University of Technology, The Netherlands
Shin-Jer Yang	Soochow University, Taiwan
Benjamin Yen	University of Hong Kong, China
Fani Zlatarova	Elizabethtown College, USA

Invited Speakers

Mehmet Aksit	University of Twente, The Netherlands
Dimitar Christozov	American University in Bulgaria - Blagoevgrad, Bulgaria
Hermann Maurer	Graz University of Technology, Austria
Bart Nieuwenhuis	University of Twente, The Netherlands

Table of Contents

Reasoning on Models Combining Objects and Aspects

Ella Roubtsova

Open University of the Netherlands
ella.roubtsova@ou.nl

Abstract. Modelling techniques are instruments for reality reflection. Precision of reality reflection demands coexistence of different abstraction types like objects and aspects in one model. Experiments with extension of modelling techniques aimed to accommodate combinations of objects and aspects in one specification have resulted in aspect-oriented extensions of many conventional modelling semantics. It was found that one of semantics called Protocol Modelling possess a very practical property of local reasoning on objects and aspects about behaviour of the whole model. In this paper the local reasoning property is defined in the reasoning logic and this property is demonstrated with a case study in the Protocol Modelling approach. Then the same case study is presented in aspect-oriented extensions of modelling approaches based on the semantics of contracts, sequence diagrams, workflows and state machines. The case study shows that the extensions of conventional semantics do not possess the local reasoning property. The semantic difference between Protocol Modelling and the listed modelling semantics is discussed and the useful semantic elements are recommended for new aspect-oriented languages and middleware.

Keywords: Local Reasoning, Aspects, Protocol Models, Contracts, Sequence Diagrams, Workflows, State Machines.

1 Introduction

Modelling abstractions were created to mirror systems and reflect the stepwise way of collecting domain knowledge during requirements engineering. Using objects for system decomposition is a very common practice and it is well known that separation of objects often causes crosscutting abstractions scattered through system specification.

In order to implement a crosscutting abstraction a modular unit called *aspect* was designed [7]. An aspect contains an *advice* in the form of a code presenting a concern and *pointcut designators* being the instructions on where, when and how to invoke the advice. The well defined places in the structure of a program or a model where an advice should be attached were named *join points*. Programming community had already accepted a join point model that used method calls as join points and inserted advice before, after or around a method call [7]. This join point model was implemented as various extensions of programming languages.

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 1–18, 2012.

Such extensions gave a new task to compilers: to produce the code with aspects woven in necessary places at the compilation time. This way an aspect is localized only at the design time. In the code it remains scattered through the code.

Another branch of aspect-oriented programming developed middleware for run-time aspect weaving without producing the code of the complete program. The weaving program in the middleware registers aspects and their pointcut designators. At run time the weaving program is intercepting the method invocations and inserting aspects before, after or around specified method invocations. The weaving programs implement sequential composition of method calls and returns of the base program and the method calls and returns of aspects.

However, it was found that the aspect-oriented programming techniques built on the existing aspect definition allow producing so-named invasive aspects [12]. Invasive aspects change the values of variables in other aspects and objects. In the case of invasive aspects no guarantee can be given about preserving behaviour of the base program after adding aspects and it is impossible to keep the reasoning control over the evolving program.

At this point the modelling community decided to investigate the problem and find the semantics that prevents constructing invasive aspects and guarantees safe modelling and system construction. The idea was to recommend such semantics for new aspect-oriented languages and weaving middleware.

The experiments were made in combining objects and aspects in different modelling techniques. These experiments have shown that practical use of modelling semantics combining different abstractions demands the convenient way to reason on models. The most attractive reasoning is the local reasoning on abstractions about behaviour of the whole system. Local reasoning makes the reasoning simple and allows building scalable models of systems and at the end the working systems.

The goal of this paper is to define local reasoning in reasoning logic and demonstrate its presence and absence in different modelling semantics. In the correspondence with the goal, section 2 reminds the reasoning logic and defines the local reasoning in this logic. Section 3 presents models of the same case study in the Protocol Modelling approach that possess the property of local reasoning and in other aspect-oriented modelling approaches that do not have such a property. The case study demonstrates the semantic elements that make the local reasoning impossible. Section 4 summarizes the semantic elements of Protocol Modelling that enable localization of reasoning.

2 Reasoning Logic and Local Reasoning

Let us consider a reasoning logic for state-transition systems: $S = (s_0, S, T)$, where s_0 is an initial state, S is a set of states and T is a set of transitions of type (s_i, s_j) and $s_0, s_i, s_j \in S$.

Let a state $s \in S$ be defined on a set of variables V used to store data, a set E variables used to temporally store events received from the environment and IE variables used to temporary store internal events generated inside the

system $S = V \cup E \cup IE$; $s = (v_1, ..., v_n, e_1, ..., e_m, ie_1, ...ie_k); n, m, k \in N$. (An event, an operation call or return can be stored in a data structure).

Let AP be a set of Atomic Propositions $\phi \in AP$ about values of variables of a system $V \cup E \cup IE$. The examples of atomic propositions are *"Amount = 2000"*, *"Event=Open"*, *"Password=Saved Password"*, etc.

A reasoning logic about a system is traditionally defined on a Kripke structure [1,21]: $\mathcal{M} = (M, R, \mu)$, where (M, R) is a reachability graph of this system. A node $m \subseteq M$ of a reachability graph is a state of the whole system. R is a set of relations on states giving possible transitions, and $\mu : M \to 2^{AP}$ is a function which assigns true values of propositions to each node of this reachability graph.

Function μ states that a predicate ψ is true in state s corresponding to node n of a reachability graph, iff and only iff the state described by the proposition is a sub-set of the set presented by the node of the reachability graph $\mu(\phi) \subseteq n$.

All variety of the reasoning statements is inductively defined as a set of predicates built from atomic propositions called state formulas $\phi \in TP$ about states in the reachability graph. We will analyse reasoning in different behaviour modelling semantics and to cover them all will use a superset CTL* of computational tree logic (CTL):

$$\psi ::= true \mid false \mid \phi \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \psi_1 AU \psi_2 \mid \psi_1 EU \psi_2.$$

The interpretation of satisfaction relations in the reasoning logic has the reachability graph semantics:

1. predicate ϕ is satisfied in all nodes m of the reachability graph where predicate $\phi = true$.
2. predicate $\neg\psi$ is satisfied in all nodes where predicate $\psi = false$.
3. predicate $\psi_1 \vee \psi_2$ is satisfied in all nodes where ϕ_1 or ψ_2 is satisfied.
4. predicate $\psi_1 \wedge \psi_2$ is satisfied in all nodes where both ϕ_1 and ψ_2 are satisfied.
5. predicate $\psi_1 AU \psi_2$ is satisfied in node m if *for every path* $m_0, m_1, ...$ of the reachability graph starting from node $m = m_0$ there is node m_i such that for nodes $m_0, ..., m_{i-1}$ predicate ψ_1 is *true* and for m_i predicate ψ_2 is *true*.
6. predicate $\psi_1 EU \psi_2$ is satisfied in node m if *for some path* $m_0, m_1, ...$ of the reachability graph starting from node $m = m_0$ there is node m_i such that for node $m_0, ..., m_{i-1}$ predicate ψ_1 is *true* and for m_i predicate ψ_2 is *true*.

There are two groups of reasoning statements: state predicates and path predicates.

- State predicates can be formulated about one state, about a set of states and all states. The state predicates are expressed using variables. The two special forms of state variables are: STATE that presents the state from the semantic point of view and EVENT that express the fact that an event of a given data structure has been sent or received or an operation presented as a data structure has been called or returned.
- Path predicates can be formulated about states that follow each other in an existing path, about states that follow each other in a set of existing paths.

The reasoning structure presented before is applied both to the whole program or model and to an abstraction.

Definition 1. Local Reasoning Statement on an Abstraction. *A reasoning statement is local to an abstraction if it is formulated in terms of the states of this abstraction and events or operation calls (returns) accepted by the abstraction and describes the states or paths of the abstraction.*

Definition 2. Local Reasoning Property of a System of Abstractions. *A model/system possesses the property of local reasoning if in any state any reasoning statement about the model/system is a conjunction of finite number of* local reasoning statements *of abstractions of this model/system and there are no other reasoning statements about this model/system.*

In another words, if a model/system possess the property of local reasoning, then the analysis of every system property is reduced to analysis of a conjunction of properties of a finite number of system abstractions.

3 Reasoning in Different Modelling Semantics

3.1 Case Study

The chosen case study is deliberately simple. It is designed to show the difference in composition of abstractions and the different reasoning possibilities in modelling semantics.

Let us consider a customer and a bank account. A customer can be registered. A registered customer can open an account and leave the bank. A customer can be frozen and released from freezing. The customer with the status "frozen" cannot leave the bank and open an account. An active account can be operated. An active account can be closed. An account can be also frozen and released. If an account is frozen then closing, depositing and withdrawing are impossible.

3.2 Protocol Modelling - Modelling with Local Reasoning

To date only one aspect-oriented modelling semantics, namely Protocol Modelling, has proven the possession of property of local reasoning [14]. Let us present the case study in Protocol Modelling and show what local reasoning means in practice. Figure 1 shows the case study in Protocol Modelling semantics.

Structure. A protocol model of a system is a composition of protocol machines *Customer*, *Account*, *Freezing* and *Freeze Control*. Protocol machines are partial descriptions of behaviour classes. For example, the behaviour class *Account* is described by three protocol machines: *Account*, *Freezing* and *Freeze Control*. Behaviour of class *Customer* is described by three protocol machines *Customer*, *Freezing* and *Freeze Control*. *Freezing* and *Freeze Control* are aspects woven into both behaviour classes *Account* and *Customer*. In order to generate own

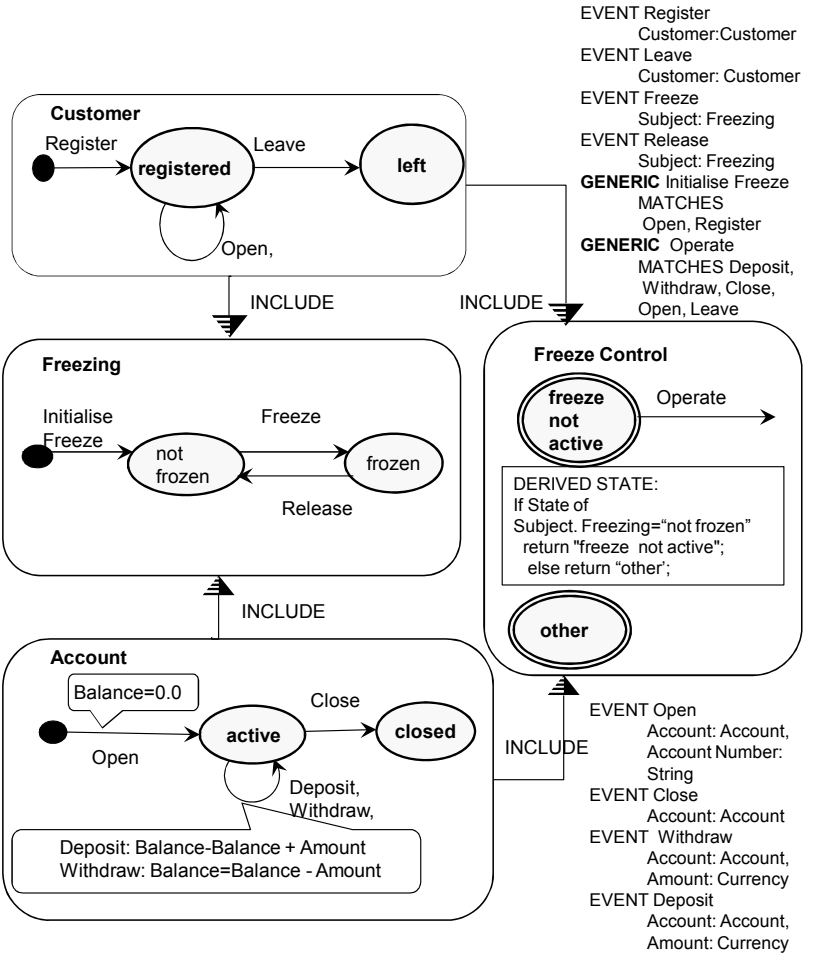


Fig. 1. Protocol Model

instances of *Freezing* and *Freeze Control* for each object of different behaviour classes, behaviours *Freezing* and *Freeze Control* are included into those objects. The INCLUDE-relation, depicted as a half-dashed triangle, gives to Protocol Models the expressiveness of multiple inheritance. Each object has its object identifier and behaviours of aspects are instantiated with instantiation of objects.

Events. A protocol machine has its own alphabet of recognized events. Event types *Open, Close, Deposit, Withdraw, Register, Leave, Freeze, Release* are presented as data structures in Figure 1. An event instance contains values of the attributes. Event alphabets of protocol machines can have a not empty intersection.

For example, the intersection of alphabets of protocol machines *Account* and *Customer* is event *Open*. It is used for synchronization of an instance of a *Customer* and an instance of an *Account*.

State. A protocol machine has its local state. The intersection of local states of protocol machines is always empty. The local state of a protocol machine is presented as a set of attributes and special enumerated attribute STATE. For example, the protocol machine *Account* has attributes *Account Number* and *Balance* and the attribute STATE with values @*new*|*open*|*closed*.

Behaviour Semantics of Event Refusal. A protocol machine presents a system that communicates with its environment. Events are presented to the model by the environment. Being in a suitable state, a protocol machine accepts the presented event, otherwise it *refuses* the event. The states accepting an *event* have an outgoing arc labeled by this event. A transition is depicted as an arc connecting two states *(state, event, state)*. The behaviour of a protocol machine is a set of sequences of accepted events. The sequences of transitions of a protocol machine can be combined into a computation tree and their properties can be described with path predicates.

CSP Parallel Composition. Behaviour of a protocol model is a composition of behaviours of its protocol machines. The composition operator for protocol machines is a variant of the parallel composition operator defined by Hoare [9] in his process algebra Communication of Sequential Processes (CSP). This operator was extended by McNeile and Simons [16] for machines and events with data. Protocol machines use the CSP parallel composition algorithm to form more complex protocol machines. This is the description of the CSP parallel composition algorithm:

- A protocol model handles one event at a time and reaches a well defined quiescent state before handling the next event;
- If all machines of the protocol model, having an event in their alphabet, accept the event, the protocol model accepts it;
- If at least one of protocol machines, having this event in its alphabet, refuses the event, the composition of machines refuses it.

Derived States. A protocol machine can have a state function to derive its states from states of other protocol machines. Derived states are states that are calculated from the state of other protocol machines. For example, the protocol machine *Freeze Control* derives its state from the state of machine *Freezing* using its state function. The state function associated with the protocol machine results in state *'freeze not active'* or state *'other'* (Figure 1).

The derived states should not be topologically connected with other states. The arc of *Freeze Control* labeled with *Operate* does not need the right-end node. The arc means that *Freeze Control* accepts event *Operate*. The output state is

defined by the transitions of stored state protocol machines labeled with event *Operate* or events matching with it.

Protocol Modelling distinguishes protocol machines with derived state from protocol machines with stored states in order to simplify modelling.

Protocol Machines with derived state can be seen as spectative aspects. They observe the state of other protocol machines, calculate state from them and allow or forbid some traces of the system [17].

Execution and Reasoning. The Protocol Model is directly executed in the Modelscope tool [15] that provides a generic interface for execution. All possible events are visible at any step of the execution. All states may be made visible during the execution. As any state variables and any attribute is local to a protocol machine, there is a local reasoning predicate about every state change. Let us go through a sequence of model execution and reasoning.

1. For a new *Customers* the only available event is *Register*.
   The reasoning is local to the object *Customer*:

$$((Customer = @new)EU(Exists\,Until)(Event = Register)).$$

2. If event *Register* takes place, then *Customer* transits into state *'registered'* and instances of two protocol machines *Freezing* and *Freeze Control* are created for the *Customer*.

   Freezing is instantiated in the state *'not frozen'* and *Freeze Control* derives its state 'freeze not active' from *Freezing* .

   Reasoning statements are local to *Customer* and *Freezing*:

$$((Event = Register)EU(Customer = registered)),$$

$$((Freezing = @new)EU(Event = InitialiseFreeze)),$$

$$((Event = InitialiseFreeze)EU(Freezing = not\,frozen)),$$

$$State\,Function:\ FreezeControl(Freezing = not\,frozen) = freeze\,not\,active,$$

$$Generic:\ Initialize\,Freeze\,MATCHES\,Register.$$

   After application of the State Function and substitution of the Generic each of three reasoning statements as well as the conjunction of these local reasoning statements is the true reasoning statement about the behaviour of the whole model at this step.
3. Next, event *Customer.Freeze* becomes possible thanks to the *Freezing* aspect:

$$((Freezing = not frozen)EU(Event = Freeze)).$$

4. Than event *Open* becomes possible thanks to *Customer* and *Freeze Control*:

$$((Customer = registered)EU(Event = Open));$$

$$((FreezeControl = freeze\,not\,active)EU(Event = Operate));$$

$$Generic:\ Operate\,MATCHES\,Open.$$

5. We also can reason that event *Register* for a chosen *Customer* is impossible because the local statements on *Customer*

$$((Customer = registered)\neg EU(not\, exists)(Event = Register)).$$

We can continue the execution and reasoning. Any state change can be explained by a conjunction of local reasoning statements on a limited number of abstractions. The composed model does not have states and paths properties of which cannot be described as a conjunction of local properties of a final number of composed protocol machines.

If a large number of instances of abstractions is involved in a reasoning, then a Protocol Machine with a derived state is created. It derives its states from all instances and reasoning remains local to this Protocol Machine with the derived states. For example, if event *Leave* for *Customer* would be possible only if all corresponding *Accounts* are closed, then we would add a protocol machine *Close Control* with the derived state *'All Accounts of Customer are closed'* and allow acceptance of event *Leave* only in this state. Modelscope provides SELECT functions [15] to select instances and derive states from the selected instances of different abstractions.

Join Points. A join point in Protocol Modelling is a set of events that can be seen identical to each other at the abstraction level of a particular protocol machine. For example, the Freezing abstraction does not see the difference between events *Open* and *Register* and defines join point *GENERIC Initialise Freeze* that matches each of these events. The *Freeze Control* abstraction does not separate events *Deposit, Withdraw, Leave and Close* and defines *GENERIC Operate* that matches each of those events.

The proof presented in [14] shows that the CSP parallel composition of protocol machines guarantees preservation of ordering of traces of aspects and objects in the whole specification. This property is called *observational consistency* [6]. In combination with localization of state and the prohibition for protocol machines to change state of each other, the observational consistency guarantees the property of local reasoning of protocol models.

Small and deterministic protocol machines are verified or tested by direct execution. Any new functionality, even the crosscutting one, is localized in a new protocol machine and synchronized with existing protocol machines. New protocol machines cannot cause any damage to behaviour of other protocol machines except possible forbidding of some traces. But this is directly identified as the conjunction of the reasoning statements local to this new forbidding protocol machine and the local reasoning statements of protocol machines allowing this trace. For example, the conjunction of reasoning statements of the *Customer* and the *Freeze Control* in state 'other':

$$((Customer = registered)EU(Event = Open))AND$$
$$((FreezeControl = other)\neg EU(Event = Operate));$$
$$GENERIC: Operate\, MATCHES\, Open.$$

results is the forbidding statement

$$((FreezeControl = other)\neg EU(Event = Open)).$$

3.3 Visual Contract Language

In this section we model our case study in the contract-based semantics called Visual Contract language (VCL) [2]. VCL explores the declarative way of aspect specification based on composition of sets of operations, attributes, classes and packages. In the VCL specification (Figure 2) classes *Customer* and *Account* are defined as sets of attributes and operations. Operations are depicted as hexagons.

A contract for a class is a set of its operations. Each operation has a corresponding diagram which specifies the operation name, its input (?) and output (!), the pre-conditions in the left hand side field and the post-conditions in the right hand side field. For example, the input of operation *Open* is an *Account Number? String* and the output is the *a! Account*. The post-condition is *Balance=0 AND Account Number=Account Number*. The empty pre-condition field means that there are no restrictions on the values of variables for this operation. (If we define variable STATE for the *Account*, than the precondition will be *(STATE≠Closed)*.

Classes can have relations. For example, *'Customer opens Account'*.

Aspects are specified as classes. Figure 2 shows aspect *Freezing*. The functionality of *Freeze Control* is specified inside *Freezing* as *Freeze Control* functionality does not contain any own operations.

Classes can be combined into packages. A class and a package may have invariants that specify the types, or relations that are not changed during the object life cycle. A package may have operations that are specified as join interfaces *(JI)*. Packages can join on join interfaces. Figure 2 shows how aspect *Freezing* is woven into objects *Account* and *Customer*. Package *AccountJI1* contains join interface *Open* used for weaving operation *Account.Freezing.Initialise Freezing*. Package *AccountJI2* contains join interface *Withdraw, Deposit, Close*. Each of these operations is used for weaving of *Account.Freezing.Get State Freezing*.

There are some general features of the contract semantics used by VCL that should be mentioned.

- The units of behaviour in contracts are operations. An operation itself has a body that may change both the state of its own object and the state of other objects. Pre- and post-conditions are even able to specify the changes of variables of other objects. Such operations cannot be elements of local behaviour of any object or aspect.
- A contract does not define what happens with an operation call if the precondition is not satisfied [18]: "If a precondition is violated, the effect of the section of code becomes undefined and thus may or may not carry out its intended work." Operation calls are not refused. Usually an operation call is kept (somewhere in a stack) waiting for the preconditions to become true. The absence of the refuse semantics in contracts makes the CSP synchronization impossible.

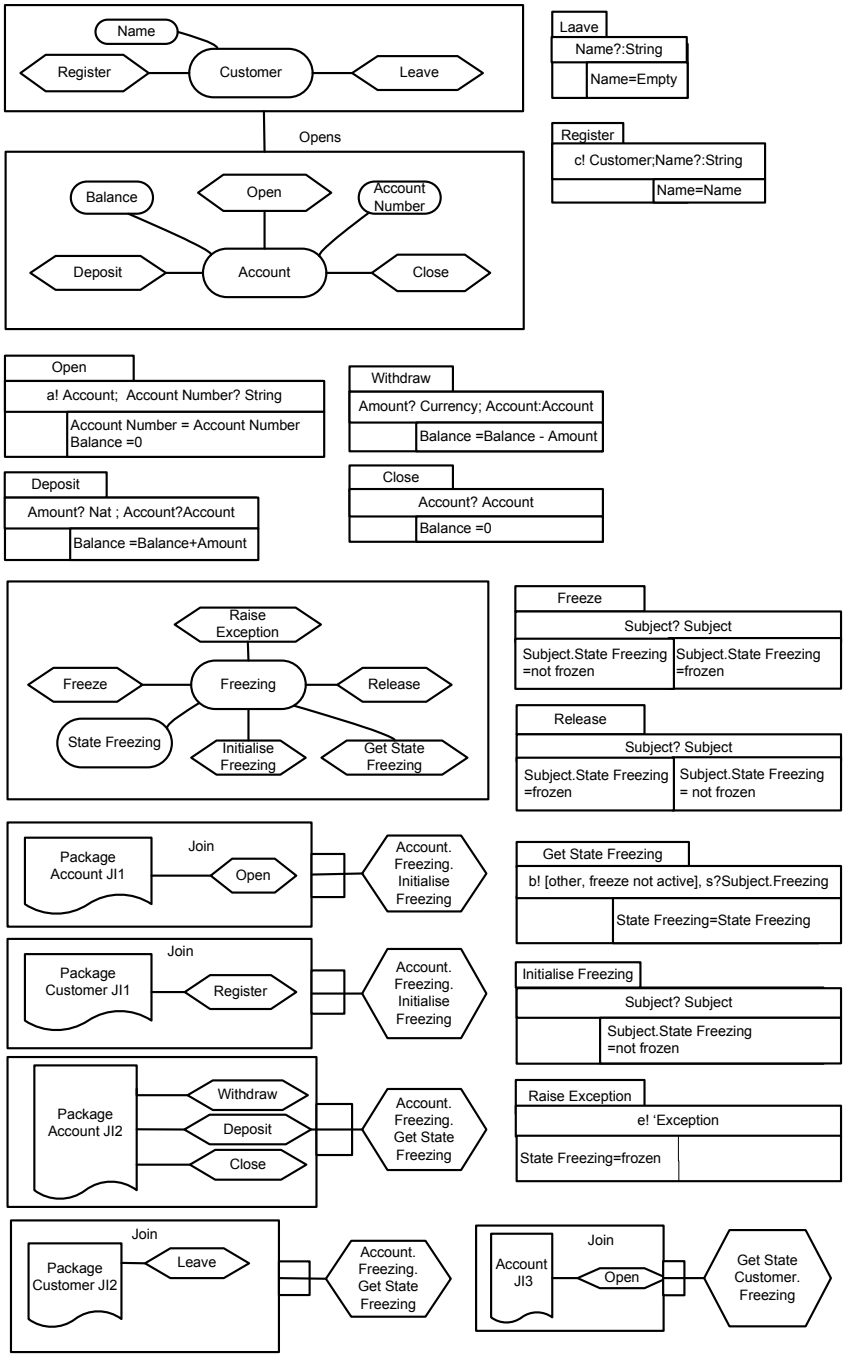


Fig. 2. VCL Model

From carefully specified contacts, having the Z-semantics, it is possible to generate a computation graph that shows the behaviour of the system but only if events happen when they are expected to happen.

– Operations are called one after another even all of them have true preconditions. The operation calls form sequences that can be sequentially composed or inserted between an operation call and return. For example, after *Register(Customer Name)*, *Initialise.Customer.Freezing* can be called and then operation *Register* can proceed to the completion. This is the aspect-oriented 'around invoke' technique. Because of the sequential way of weaving, the behaviour of the whole model (and its computation graph) will always contain states that cannot be composed from the states of the abstractions. It is impossible to reason about such new states using reasoning statements defined on the states of abstractions.

  For example, if $((Customer.Freezing = other)\neg EU(Event = Open))$ and event *Open* is called, it will not be refused immediately as the state of *Customer.Freezing* has to be checked. There will be a state after *Customer.Open* before call *Customer.Get State Freezing* where

$$((Customer.Freezing = other)\neg EU(Event = Open)) \, is \, false.$$

– In order to check a state of another package, the abstraction has to call operation *Get State*. This technique does not allow abstraction A to have derived states corresponding to the states of abstraction B. During the time interval between the call of *Get State* and its return the state of abstraction B can be changed. Therefore, quantification on states and using derived states as join points is impossible.

We can now summarize, that three semantic elements: (1) using operations that can change the state of other objects, (2) absence of operation synchronization and (3) sequential composition of operations, - produce in the whole model extra states and paths that cannot be described as composition of states of local abstractions. The whole model needs global reasoning and complete reachability graph has to be analyzed using theorem proving techniques.

3.4 Sequence Diagrams with Joint Point Diagrams

A set of sequence diagrams with conventional semantics is aimed to present only a part of possible sequences of system behaviour. Sequence diagrams illustrate behaviour of programs and therefore use operation calls and returns as elements of behaviour. The composition techniques of sequence diagrams are restricted to sequential composition, alternatives, cycles and insertion of sequences of operations calls and returns.

For Aspect-Oriented Modelling (AOM) the conventional sequence diagrams were extended with Join Point Designation Diagrams (JPDDs) [23]. Conventional sequence diagrams usually specify sequences to the completion of a use case. Sequence diagrams with JPDDs specify sequences of base objects as well as fragments of sequences of repeated aspects and join points.

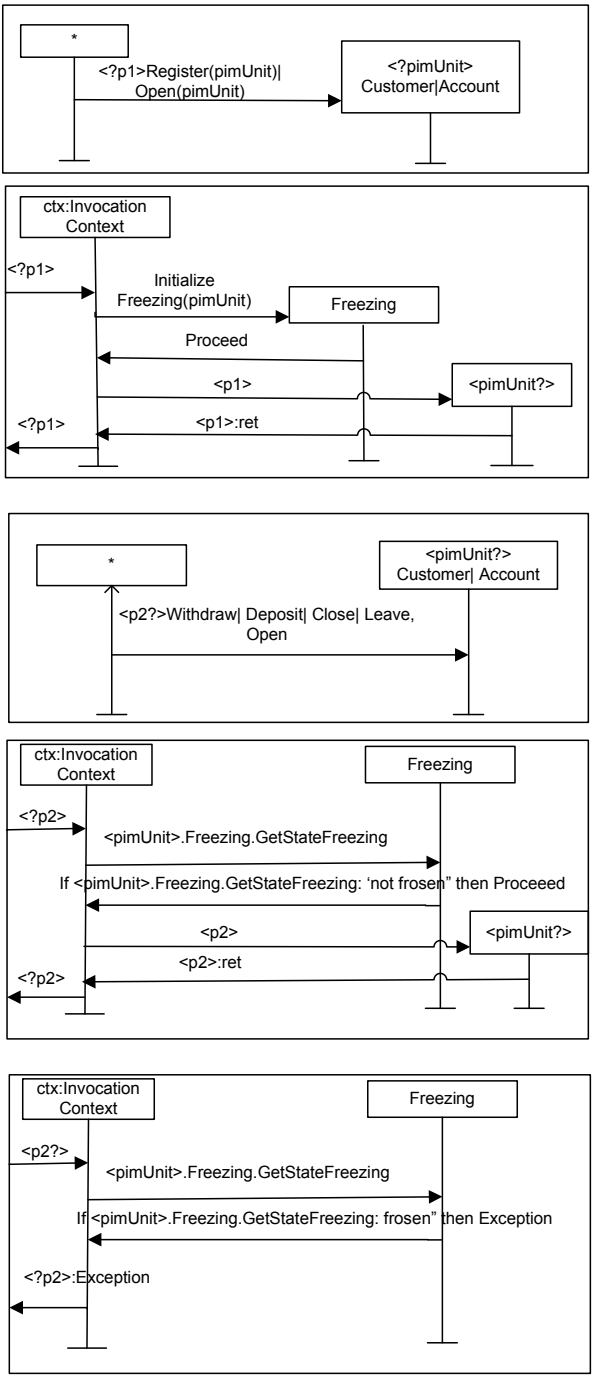


Fig. 3. JPDD diagrams

Figure 3 specifies JPDDs for our case study. The figure does not show the sequences of the base objects *Customer* and *Account* but specifies join points and advice fragments. JPDDs (1) and (3) are modelling means to graphically represent join point queries on *Customer* and *Account*. They use lists of operations that serve as join points. Diagram 2 presents the advice for initializing of aspect *Freezing* and diagrams 4 and 5 specify two different advice traces of the aspect *Freeze Control*.

In reality the set of sequences is infinite and sequence diagrams with JPDDs do not present the complete system behaviour to reason on it. However, even when all possible sequences are specified for a simple model then the sequences are combined into a computation graph using the same sequential composition technique as in contracts. Sequence diagrams use all three semantic elements that make local reasoning impossible.

Several approaches such as Theme [3], GrACE (Graph-based Adaptation, Configuration and Evolution [4], RAM (Reusable Aspect Models) [11] use JPDDs in combination with class diagrams. All approaches use global reasoning techniques [3,4,11].

3.5 Workflows as Aspect-Oriented Notations

Activity and workflow based approaches are aimed to specify complete system behaviour that can be analyzed and verified against required properties. The workflows are often used in AOM approaches as integration means to combine specified aspects. For example, the Theme approach [5] uses an activity diagram as an integration view. There are also AOM approaches that define fragments of workflows and compose these fragments into the complete workflow. An example is the approach called Activity moDel supOrting oRchestration Evolution (ADORE) [19].

Figure 4 renders our case study in ADORE. ADORE specifies a computation graph (a process) combining several basic abstractions. In our case it combines behaviours of *Customer* and *Account* in the workflow. Repeated partial behaviours of abstractions are specified as workflow fragments (or aspects). For example, behaviours *Freezing*, *Freeze Control* and *Leaving* are specified as fragments. Each fragment corresponds to a specific aspect and it is used as a partial point of view on its target. A fragment contains special activities, called predecessors P, successors S and hooks (assimilated as a Proceed in AspectJ). The hook predecessors *(P)* are the immediate predecessors of the first activity in the target block, and the hook successors *(S)* are the immediate successors of the last activity in the block.

The binding or weaving instructions assigning predecessors and successors are specified in a separate file. For example, the fragment $P3; S3$ of *Freeze Control* from Figure 4 can be bound as follows

$$P3 = i : Withdraw; S3 = r : Withdraw,$$

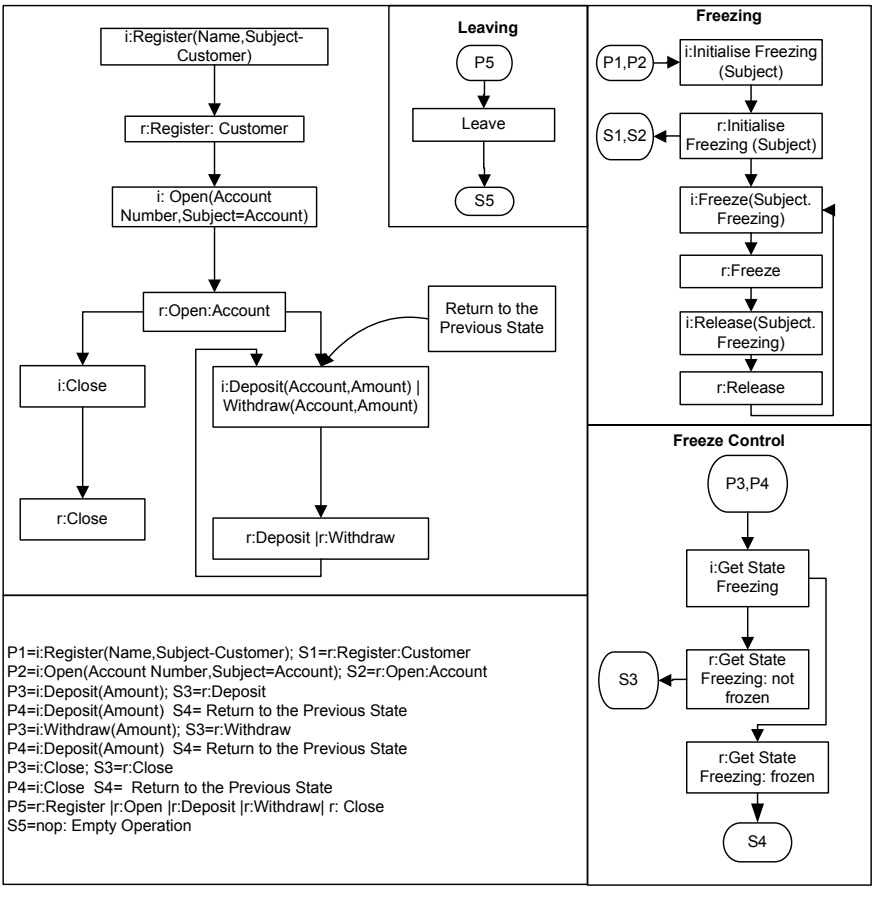


Fig. 4. ADORE Workflow diagram

where i is the invocation and r is the return of operation *Withdraw*. The hook is the sequence of activities

$$hook = i : Get\,State\,Freezing; r : GetStateFreazing = not frozen.$$

The complete orchestration is generated from fragments according to the binding instructions.

As with such an approach there is no guarantee that local properties of aspects are propagated to the complete orchestration, the side effects of separating of concerns and composition in ADORE are formulated as rules. The rule violation not always signals a mistake. It may indicate a "bad-smell", like, for example, the non-determinism caused by two conditions evaluated to false at the same time. The "bad-smells" are analyzed by the designer of the orchestration.

All workflow fragments are combined at design or runtime into a computation graph. This means that the behaviour composition technique is the same as in contract-based and sequences based notations.

It is possible to synchronize operation calls and returns in workflows using a synchronization construction. It is also possible to work on the level of events and do not separate calls and returns. However, one semantic feature makes this synchronization different from the CSP parallel composition used in Protocol Modelling. Namely, workflows do not have the semantics of event refusal. At any state several events may happen and the events are kept in bags or stack structures. The event may wait until the model transits to the state where this event is accepted. The computation graph of workflows depends on the state of those stacks or bags and the system has states that cannot be described as conjunction of states of system abstractions. Such a composition semantics does not leave any other possibility for reasoning than the global reachability analysis.

3.6 Aspect-Oriented Extension of State Machines

A UML Behaviour State Machines (BSM) [20] usually presents behaviour of one class. There are several approaches trying to extend BSM to enable several BSMs for one class. Mahoney et al. [13] suggested to exploit the *AND-composition* of several independent (orthogonal) statecharts defined by D.Harel [8]. "The key feature of orthogonal statecharts is that events from every composed statechart are broadcast to all others. Therefore an event can cause transitions in two or more orthogonal statecharts simultaneously" [13].

The ideas proposed by Mahoney et al. were further developed in the approach called High-Level Aspects (HiLA) [10]. HiLA modifies the semantics of BSM allowing classifiers to apply additional or alternative behaviour. Aspects extend the behaviour specified for classes.

The basic static structure usually contains one or more classes. Each base state machine is attached to one of these classes and specifies its behaviour. Figure 5 shows two state machines *Customer* and *Account* that look similar to protocol machines, but have different semantics.

- The first difference is in the semantics of labels on the arcs. A label of a protocol machine presents an *event* but a label of a state machine presents some state information before and after the event that run to completion: $[precondition]\, event/\, [postcondition]$ [20].
- The second difference is the absence of event refusal. Events coming from the environment are kept in a queue or a stack of active events [20]. There are complex rules for handling or keeping events in the queue until the state is appropriate for their handling.

The HiLA approach does not change both mentioned semantic features but offers the patterns of aspect weaving. High-level aspects apply to state machines and specify additional or alternative behaviour to be executed at certain "appropriate" points in time of the base machines execution.

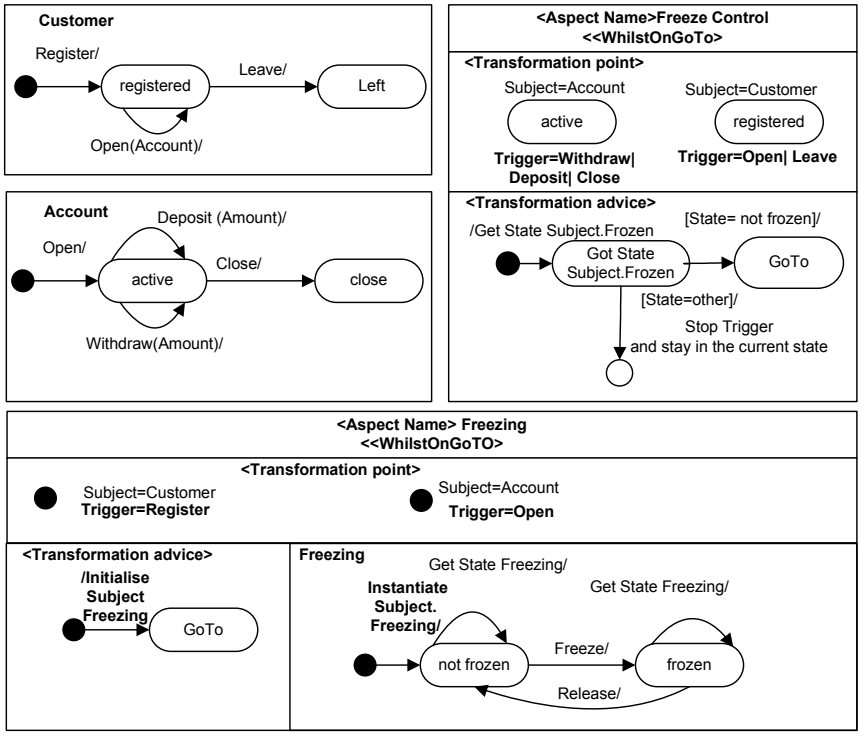


Fig. 5. HiLA State Machines

HiLA introduces patterns for specification of dynamic aspects. Any pattern has an *Aspect Name*, a *Pattern Type*, a *Transformation Point* and a *Transformation Advice*.

In Figure 5 we use pattern of type *"WhilstOnGoTo"* to specify aspects of our case study.

For example, the transformation point of aspect *Freezing* shows that the event *Initilise Freezing* happens in two situations (1) *Whilst Customer* is in the initial state, shown by the black dot, and when event *Register* takes place and becomes the trigger and (2) *Whilst Account* is in the initial state and event *Open* becomes the trigger. A pattern *Whilst* always has a specification of a state and an annotation $Trigger = e$. Conceptually it selects the compound transition from *State* with *Trigger* e, but if this transition does not exist, it is created [10,24]. This means that an aspect is added while an action is in the stack of active actions.

As the authors of the approach indicate [10,24], weaving of aspects into basis BSM results in another UML state machine which is analyzed using the model checking component of Hugo/RT model checking tools. Hugo/RT translates the state machine and the assertions into the input language of a back-end model checker SPIN. SPIN then is used to verify the given properties presented in Linear Temporal Logic for global analysis of the model behaviour.

4 Conclusion

This paper presents a survey of modelling semantics designed to accommodate aspects and objects in one model. The need of scalable models and reasoning control over complex models shows that accommodation of different abstraction types in one model demands more than just instructions on where, when and how to invoke advice of aspects. It is desired that modelling semantics possess the property of local reasoning on abstractions about behaviour of the whole model to reduce the analysis of any whole model property to the analysis of a finite number of local properties of model abstractions.

In this paper we have given the definition of local reasoning in the reasoning logic. We have applied the definition to show that the models built in many modelling semantics do not possess local reasoning property as they have extra states and paths that cannot be described as conjunction of states and paths of their abstractions.

Using the definition we have shown that Protocol Models possess local reasoning property. Other approaches may use semantic findings of Protocol Modelling as a notation independent basis for combining objects and aspects. The semantic elements that are needed for local reasoning are the following:

- considering events as instances of data structures but not as operations and this way avoiding state transformation defined inside operation bodies;
- using semantics of event refusal allowing synchronization of behaviour abstractions and the CSP parallel composition;
- handing one event at a time until the system stays in a quiescent state;
- allowing abstractions to transform their own state and read but not modify the state of other abstractions;
- allowing abstractions to derive their state from the state of other abstractions.
- mapping events to an alias if events are not differentiated at the level of a particular abstraction.

As further experiments show [22], with involving yet other abstractions into design, a combination of composition techniques might be necessary. The theory of system modelling has developed many composition techniques, but their combinations have not been investigated yet and have not been implemented in suitable modelling and programming tools. As the systems become more complex and use abstractions with different communication techniques, the practical modelling approaches, using combination of composition techniques and providing local reasoning when possible, are yet to be found and implemented.

References

1. Alur, R., Courcoubetis, C., Dill, D.: Model-checking in dense real-time. Information and Computation 104(1), 2–34 (1993)
2. Amálio, N., Kelsen, P.: VCL, a Visual Language for Modelling Software Systems Formally. In: Goel, A.K., Jamnik, M., Narayanan, N.H. (eds.) Diagrams 2010. LNCS, vol. 6170, pp. 282–284. Springer, Heidelberg (2010)

3. Baniassad, E., Clarke, S.: Theme: An Approach for Aspect-Oriented Analysis and Design. In: Proceedings of the 26th International Conference on Software Engineering, ICSE 2004, pp. 158–167. IEEE (2004)
4. Ciraci, S., Havinga, W.K., Akşit, M., Bockisch, C.M., van den Broek, P.M.: A Graph-Based Aspect Interference Detection Approach for UML-Based Aspect-Oriented Models. Technical Report TR-CTIT-09-39, Enschede (September 2009)
5. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison Wesley (2005)
6. Ebert, J., Engels, G.: Observable or invocable behaviour-you have to choose. Technical report. Universität Koblenz, Koblenz, Germany (1994)
7. Filman, R., Elrad, T., Clarke, S., Akşit, M.: Aspect-Oriented Software Development. Addison-Wesley (2004)
8. Harel, D., Gery, E.: Executable Object Modelling with Statecharts. IEEE Computer 30(7), 31–42 (1997)
9. Hoare, C.: Communicating Sequential Processes. Prentice-Hall International (1985)
10. Hölzl, M.M., Knapp, A., Zhang, G.: Modeling the Car Crash Crisis Management System Using HiLA. T. Aspect-Oriented Software Development 7, 234–271 (2010)
11. Kienzle, J., Al Abed, W., Klein, J.: Aspect-oriented Multi-view Modeling. In: Proceedings of the International Conference on Aspect-Oriented Software Development, AOSD 2009, Charlottesville, Virginia, USA, pp. 87–98 (2009)
12. Katz, S.: Aspect Categories and Classes of Temporal Properties. In: Rashid, A., Aksit, M. (eds.) Transactions on AOSD I. LNCS, vol. 3880, pp. 106–134. Springer, Heidelberg (2006)
13. Mahoney, M., Bader, A., Elrad, T., Aldawud, O.: Using Aspects to Abstract and Modularize Statecharts. In: The 5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004 (2004)
14. McNeile, A., Roubtsova, E.: CSP parallel composition of aspect models. In: AOM 2008: Proceedings of the 2008 AOSD Workshop on Aspect-Oriented Modeling, pp. 13–18 (2008)
15. McNeile, A., Simons, N.: http://www.metamaxim.com/
16. McNeile, A., Simons, N.: State Machines as Mixins. Journal of Object Technology 2(6), 85–101 (2003)
17. McNeile, A., Simons, N.: Protocol Modelling. A Modelling Approach that Supports Reusable Behavioural Abstractions. Software and System Modeling 5(1), 91–107 (2006)
18. Meyer, B.: Object-Oriented Software Construction. Prentice Hall (1997)
19. Mosser, S., Blay-Fornarino, M., France, R.: Workflow Design Using Fragment Composition - Crisis Management System Design through ADORE. T. Aspect-Oriented Software Development 7, 200–233 (2010)
20. OMG. Unified Modeling Language: Superstructure version 2.1.1 formal/2007-02-03 (2003)
21. Pnueli, A.: The temporal logic of programs. In: Proc. 18th IEEE Symp. Foundations of Computer Csience (FOCS 1977), Providence, RI, USA, pp. 46–57 (1977)
22. Roubtsova, E., McNeile, A.: Abstractions, Composition and Reasoning. In: AOM 2009: Proceedings of the 13th Workshop on Aspect-Oriented Modeling, Charlottesville, Virginia, USA (2009)
23. Stein, D., Hanenberg, S., Unland, R.: Visualizing Join Point Selections Using Interaction-Based vs. State-Based Notations Exemplified With Help of Business Rules. In: EMISA 2005, pp. 94–107 (2005)
24. Zhang, G., Hölzl, M.: HiLA: High-Level Aspects for UML State Machines. In: Ghosh, S. (ed.) MODELS 2009. LNCS, vol. 6002, pp. 104–118. Springer, Heidelberg (2010)

Model-Based Techniques for Performance Engineering of Business Information Systems

Samuel Kounev[1], Nikolaus Huber[1], Simon Spinner[2], and Fabian Brosig[1]

[1] Karlsruhe Institute of Technology (KIT)
Am Fasanengarten 5
76131 Karlsruhe, Germany
{kounev,fabian.brosig,nikolaus.huber}@kit.edu
[2] FZI Research Center for Information Technology
Haid-und-Neu-Straße 10
76131 Karlsruhe, Germany
spinner@fzi.de

Abstract. With the increasing adoption of virtualization and the transition towards Cloud Computing platforms, modern business information systems are becoming increasingly complex and dynamic. This raises the challenge of guaranteeing system performance and scalability while at the same time ensuring efficient resource usage. In this paper, we present a historical perspective on the evolution of model-based performance engineering techniques for business information systems focusing on the major developments over the past several decades that have shaped the field. We survey the state-of-the-art on performance modeling and management approaches discussing the ongoing efforts in the community to increasingly bridge the gap between high-level business services and low level performance models. Finally, we wrap up with an outlook on the emergence of self-aware systems engineering as a new research area at the intersection of several computer science disciplines.

Keywords: business information systems, performance, scalability, predictive modeling, simulation.

1 Introduction

Modern business information systems are expected to satisfy increasingly stringent performance and scalability requirements. Most generally, the *performance* of a system refers to the degree to which the system meets its objectives for timeliness and the efficiency with which it achieves this [55,25]. Timeliness is normally measured in terms of meeting certain response time and/or throughput requirements, *response time* referring to the time required to respond to a user request (e.g., a Web service call or a database transaction), and *throughput* referring to the number of requests or jobs processed per unit of time. Scalability, on the other hand, is understood as the ability of the system to continue to meet its objectives for response time and throughput as the demand for the services it

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 19–37, 2012.

provides increases and resources (typically hardware) are added. Numerous studies, e.g., in the areas of e-business, manufacturing, telecommunications, health care and transportation, have shown that a failure to meet performance requirements can lead to serious financial losses, loss of customers and reputation, and in some cases even to loss of human lives. To avoid the pitfalls of inadequate Quality-of-Service (QoS), it is important to analyze the expected performance and scalability characteristics of systems during all phases of their life cycle. The methods used to do this are part of the discipline called *Performance Engineering*. Performance Engineering helps to estimate the level of performance a system can achieve and provides recommendations to realize the optimal performance level. The latter is typically done by means of performance models (e.g., analytical queueing models or simulation models) that are used to predict the performance of the system under the expected workload.

However, as systems grow in size and complexity, estimating their performance becomes a more and more challenging task. Modern business information systems based on the Service-Oriented Architecture (SOA) paradigm are often composed of multiple independent *services* each implementing a specific business activity. Services are accessed according to specified workflows representing business processes. Each service is implemented using a set of software components distributed over physical tiers as depicted in Figure 1. Three tiers exist: presentation tier, business logic tier, and data tier. The presentation tier includes Web servers hosting Web components that implement the presentation logic of the application. The business logic tier normally includes a cluster of application servers hosting business logic components that implement the business logic of the application. Middleware platforms such as Java EE, Microsoft .NET, or Apache Tomcat are often used in this tier to simplify application development by leveraging some common services typically used in enterprise applications. Finally, the data tier includes database servers and legacy systems that provide data management services.

The inherent complexity of such architectures makes it difficult to manage their end-to-end performance and scalability. To avoid performance problems, it is essential that systems are subjected to rigorous performance evaluation during the various stages of their lifecycle. At every stage, performance evaluation is conducted with a specific set of goals and constraints. The goals can be classified in the following categories, some of which partially overlap:

Platform Selection: Determine which hardware and software platforms would provide the best scalability and cost/performance ratio?

Platform Validation: Validate a selected combination of platforms to ensure that taken together they provide adequate performance and scalability.

Evaluation of Design Alternatives: Evaluate the relative performance, scalability and costs of alternative system designs and architectures.

Performance Prediction: Predict the performance of the system for a given workload and configuration scenario.

Performance Tuning: Analyze the effect of various deployment settings and tuning parameters on the system performance and find their optimal values.

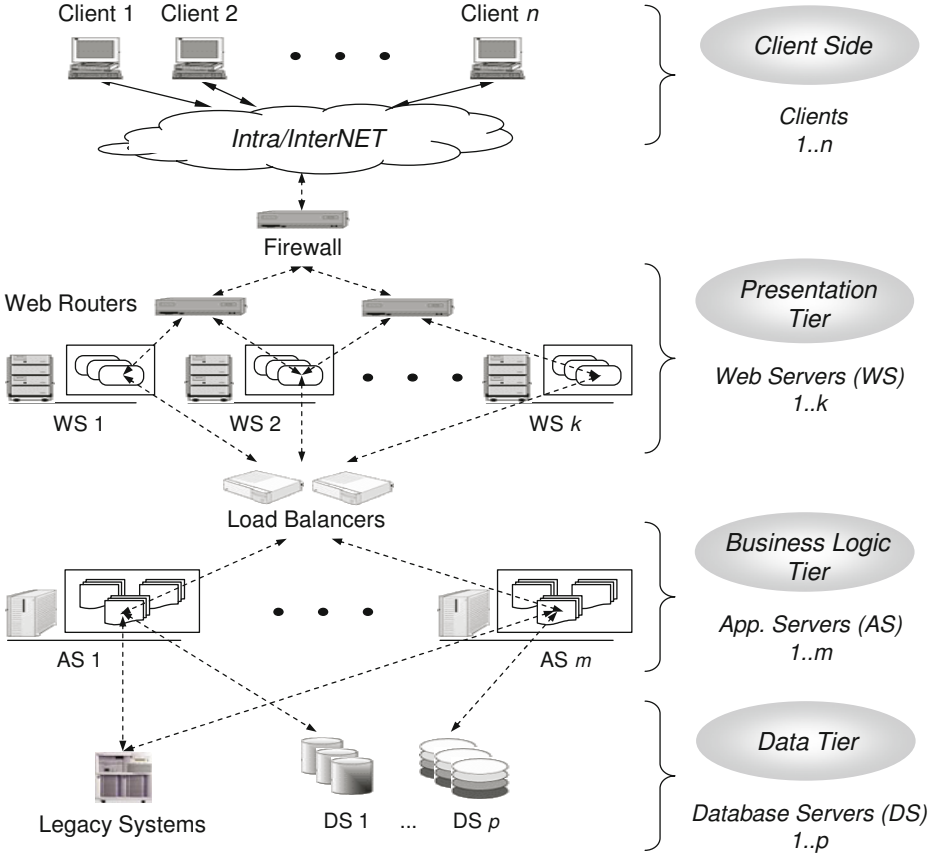


Fig. 1. Modern business information system

Performance Optimization: Find the components with the largest effect on performance and study the performance gains from optimizing them.

Scalability and Bottleneck Analysis: Study the performance of the system as the load increases and more hardware is added. Find which system components are most utilized and investigate if they are potential bottlenecks.

Sizing and Capacity Planning: Determine how much hardware resources are required to guarantee certain performance levels.

Run-Time Performance and Power Management: Determine how to vary resource allocations during operation in order to ensure that performance requirements are continuously satisfied while optimizing power consumption in the face of frequent variations in service workloads.

Two broad approaches are used in Performance Engineering for performance evaluation of software systems: performance measurement and performance modeling. In the first approach, load testing tools and benchmarks are used to generate artificial workloads on the system and to measure its performance. In the

second approach, performance models are built and then used to analyze the performance and scalability characteristics of the system.

In this paper, we focus on performance modeling since it is normally much cheaper than load testing and has the advantage that it can also be applied in the early stages of system development before the system is available for testing. We present a historical perspective on the evolution of performance modeling techniques for business information systems over the past several decades, focusing on the major developments that have shaped the field, such as the increasing integration of software-related aspects into performance models, the increasing parametrization of models to foster model reuse, the increasing use of automated model-to-model transformations to bridge the gap between models at different levels of abstraction, and finally the increasing use of models at run-time for online performance management.

The paper starts with an overview of classical performance modeling approaches which is followed by an overview of approaches to integrate performance modeling and prediction techniques into the software engineering process. Next, automated model-to-model transformations from architecture-level performance models to classical stochastic performance models are surveyed. Finally, the use of models at run-time for online performance management is discussed and the paper is wrapped up with some concluding remarks.

2 Classical Performance Modeling

The performance modeling approach to software performance evaluation is based on using mathematical or simulation models to predict the system performance under load. A *performance model* is an abstract representation of the system that relates the workload parameters with the system configuration and captures the main factors that determine the system performance [45].

A number of different methods and techniques have been proposed in the literature for modeling software systems and predicting their performance under load. Most of them, however, are based on the same general methodology that proceeds through the steps depicted in Figure 2 [46,55,24]. First, the goals and objectives of the modeling study are specified. After this, the system is described in detail in terms of its hardware and software architecture. Next, the workload of the system is characterized and a workload model is built. The workload model is used as a basis for building a performance model. Before the model can be used for performance prediction, it has to be validated. This is done by comparing performance metrics predicted by the model with measurements on the real system obtained in a small testing environment. If the predicted values do not match the measured values within an acceptable level of accuracy, the model must be refined and/or calibrated. Finally, the validated performance model is used to predict the system performance for the deployment configurations and workload scenarios of interest. The model predictions are analyzed and used to address the goals set in the beginning of the modeling study.

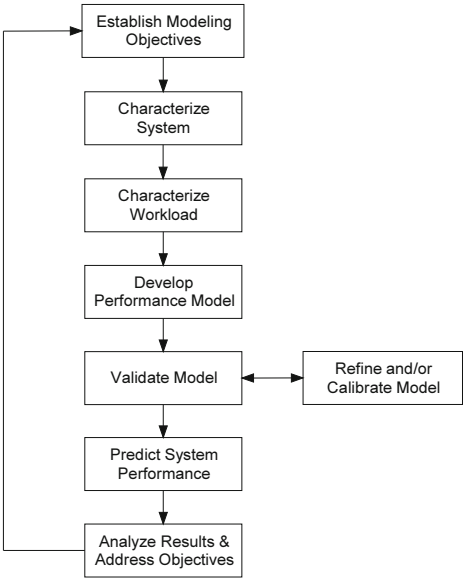


Fig. 2. Performance modeling process

2.1 Workload Characterization

Workload characterization is the process of describing the workload of the system in a qualitative and quantitative manner [44]. The result of workload characterization is a nonexecutable workload model that can be used as input to performance models. Workload characterization usually involves the following activities [55,43]:

1. *The basic components of the workload are identified.* Basic component refers to a generic unit of work that arrives at the system from an external source [42]. Some examples include HTTP requests, Web service invocations, database transactions, and batch jobs. The choice of basic components and the decision how granular they are defined depend on the nature of the services provided by the system and on the modeling objectives.
2. *Basic components are partitioned into workload classes.* To improve the representativeness of the workload model, the basic components are partitioned into classes (called *workload classes*) that have similar characteristics. The partitioning can be done based on different criteria, depending on the type of system modeled and the goals of the modeling effort [42,48].
3. *The system components and resources used by each workload class are identified.* For example, an online request to place an order might require using a Web server, application server, and backend database server. For each server, the concrete hardware and software resources used must be identified and characterized.

4. *The inter-component interactions and processing steps are described.* The aim of this step is to described the processing steps, the inter-component interactions, and the flow of control for each workload class. Also for each processing step, the hardware and software resources used are specified.
5. *Service demands and workload intensities are quantified.* The goal is to quantify the load placed by the workload components on the system. *Service demand parameters* specify the average total amount of service time required by each workload class at each resource. *Workload-intensity parameters* provide for each workload class a measure of the number of units of work, that contend for system resources.

One of the greatest challenges in workload characterization is to obtain values for service demand parameters. Most techniques require the availability of a system to take measurements. If this is not possible, some techniques can also be used to estimate service demand parameters in the early stages of system development before the system is available for testing [47]. The process to obtain service demands through measurements at a running systems usually consists of three steps. First, the performance metrics that need to be monitored for quantifying service demands are selected. It is usually not possible to measure the service demands directly. Instead, the service demands must be derived from other metrics, which can be readily observed at the system. Typical metrics are the aggregate CPU utilization, the throughput and the transaction response time. Second, measurement data for the selected metrics is gathered from the system. This is usually done either by running controlled experiments in a test environment or by monitoring production systems while serving real workloads. Finally, the measurement data gathered in the previous step is analyzed and transformed in order to derive service demands.

A common approach to derive service demands from indirect measurements is based on the Service Demand Law [46]. This operational law states that the service demand $D_{i,r}$ of class r transactions at resource i is equal to the average utilization $U_{i,r}$ of resource i by class r transactions divided by the average throughput $X_{0,r}$ of class r transactions during the measurement interval, i.e.

$$D_{i,r} = \frac{U_{i,r}}{X_{0,r}}. \tag{1}$$

However, system monitors usually provide only statistics of the overall utilization of a resource aggregated over all workload classes. There are two approaches to determine values for $U_{i,r}$: conduct a single experiment injecting transactions from all workload classes simultaneously or conduct several experiments injecting only transactions of a single workload class at a time. In the former case, interactions between workload classes are not included in the service demands. In the latter case, methods to apportion the measured total utilization between workload classes are required, as described in [42,45,46]. However, there is often some unattributed resource usage due to system overheads. It is hard to find a fair distribution of the unattributed resource usage between workload classes [23].

Other approaches to resource demand estimation have been proposed over the years, e.g., based on linear regression [3,51,49], general optimization techniques [62,35,1], or Kalman filters [63,33]. Each of these approaches makes certain assumptions, e.g., regarding the type, amount and quality of measurements. The decision which of these estimation approaches can be used depends heavily on the modeled system.

2.2 Stochastic Performance Models

Performance models have been employed for performance prediction of software systems since the early seventies. In 1971, Buzen proposed modeling systems using queueing network models and developed solution techniques for several important classes of models. Since then many advances have been made in improving the model expressiveness and developing efficient model analysis techniques as well as accurate approximation techniques. A number of modeling techniques utilizing a range of different performance models have been proposed including standard queueing networks, extended and layered queueing networks, stochastic Petri nets, queueing Petri nets, stochastic process algebras, Markov chains, statistical regression models and simulation models. Performance models can be grouped into two main categories: *simulation models* and *analytical models*. One of the greatest challenges in building a good model is to find the right level of abstraction and granularity. A general rule of thumb is: Make the model as simple as possible, but not simpler! Including too much detail might render the model intractable, on the other hand, making it too simple might render it unrepresentative.

Simulation Models. Simulation models are software programs that mimic the behavior of a system as requests arrive and get processed at the various system resources. Such models are normally stochastic because they have one or more random variables as input (e.g., the times between successive arrivals of requests). The structure of a simulation program is based on the states of the simulated system and events that cause the system state to change. When implemented, simulation programs count events and record the duration of time spent in different states. Based on these data, performance metrics of interest (e.g., the average time a request takes to complete or the average system throughput) can be estimated at the end of the simulation run. Estimates are provided in the form of confidence intervals. A confidence interval is a range with a given probability that the estimated performance metric lies within this range. The main advantage of simulation models is that they are very general and can be made as accurate as desired. However, this accuracy comes at the cost of the time taken to develop and run the models. Usually, many long runs are required to obtain estimates of needed performance measures with reasonable confidence levels.

Several approaches to developing a simulation model exist. The most time-consuming approach is to use a general purpose programming language such as C++ or Java, possibly augmented by simulation libraries (e.g., CSIMor SimPack, OMNeT++, DESMO-J). Another approach is to use a specialized

simulation language such as GPSS/H, Simscript II.5, or MODSIM III. Finally, some simulation packages support graphical languages for defining simulation models (e.g., Arena, Extend, SES/workbench, QPME). A comprehensive treatment of simulation techniques can be found in [34,2].

Analytical Models. Analytical models are based on mathematical laws and computational algorithms used to derive performance metrics from model parameters. Analytical models are usually less expensive to build and more efficient to analyze compared to simulation models. However, because they are defined at a higher level of abstraction, they are normally less detailed and accurate. Moreover, for models to be mathematically tractable, usually many simplifying assumptions need to be made impairing the model representativeness. Queueing networks and generalized stochastic Petri nets are perhaps the two most popular types of models used in practice.

Queueing networks provide a very powerful mechanism for modeling hardware contention (contention for CPU time, disk access, and other hardware resources). A number of efficient analysis methods have been developed for a class of queueing networks called product-form queueing networks allowing models of realistic size and complexity to be analyzed with a minimum overhead [9]. The downside of queueing networks is that they do not provide direct means to model software contention aspects accurately (contention for processes, threads, database connections, and other software resources), as well as blocking, simultaneous resource possession, asynchronous processing, and synchronization aspects. Even though extensions of queueing networks, such as extended queueing networks [37] and layered queueing networks (also called stochastic rendezvous networks) [60], provide some support for modeling software contention and synchronization aspects, they are often restrictive and inaccurate.

In contrast to queueing networks, generalized stochastic Petri net models can easily express software contention, simultaneous resource possession, asynchronous processing, and synchronization aspects. Their major disadvantage, however, is that they do not provide any means for direct representation of scheduling strategies. The attempts to eliminate this disadvantage have led to the emergence of queueing Petri nets [4], which combine the modeling power and expressiveness of queueing networks and stochastic Petri nets. Queueing Petri nets enable the integration of hardware and software aspects of system behavior in the same model [28]. A major hurdle to the practical use of queueing Petri nets, however, is that their analysis suffers from the state space explosion problem limiting the size of the models that can be solved. Currently, the only way to circumvent this problem is by using simulation for model analysis [29].

Details of the various types of analytical models are beyond the scope of this article. The following books can be used as reference for additional information [9,57,5]. The Proceedings of the ACM SIGMETRICS Conferences and the Performance Evaluation Journal report recent research results in performance modeling and evaluation. Further relevant information can be found in the Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE), the Proceedings of the International Conference on

Quantitative Evaluation of SysTems (QEST), the Proceedings of the Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), and the Proceedings of the International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS).

3 Software Performance Engineering

A major hurdle to the adoption of classical performance modeling approaches in industry is the fact that performance models are expensive to build and require extensive experience and expertise in stochastic modeling which software engineers typically do not possess. To address this issue, over the last fifteen years, a number of approaches have been proposed for integrating performance modeling and prediction techniques into the software engineering process. Furthermore, using models and automatic transformation and processing can simplify the software performance engineering approach, making it less error-prone.

3.1 Software Performance Meta-models

Efforts introducing performance models in the software engineering process were initiated with Smith's seminal work pioneered under the name of *Software Performance Engineering (SPE)* [53]. Since then a number of languages (i.e., meta-models) for describing performance-relevant aspects of software architectures and execution environments have been developed by the SPE community, the most prominent being the UML SPT profile (UML Profile for Schedulability, Performance and Time) and its successor the UML MARTE profile (UML Profile for Modeling and Analysis of Real-time and Embedded Systems). The latter are extensions of UML (Unified Modeling Language) as the de facto standard modeling language for software architectures. Other proposed architecture-level performance meta-models include SPE-MM [54], CSM [50] and KLAPER [17]. The common goal of these efforts is to enable the automated transformation of architecture-level performance models into analytical or simulation-based performance models that can be solved using classical analysis techniques (see Section 3.2).

In recent years, with the increasing adoption of Component-Based Software Engineering (CBSE), the SPE community has focused on adapting and extending conventional SPE techniques to support component-based systems. A number of architecture-level performance meta-models for component-based systems have been proposed as surveyed in [31]. Such meta-models provide means to describe the performance-relevant aspects of software components (e.g., internal control flow and resource demands) while explicitly capturing the influences of their execution context. The idea is that once component models are built they can be reused in multiple applications and execution contexts. The performance of a component-based system can be predicted by means of compositional analysis techniques based on the performance models of its components. Over the last five years, research efforts have been targeted at increasing the level

of parametrization of component models to capture additional aspects of their execution context.

An example of a mature modeling language for component-based systems is given by the Palladio Component Model (PCM) [6]. In PCM, the component execution context is parameterized to explicitly capture the influence of the component's connections to other components, its allocated hardware and software resources, and its usage profile including service input parameters. Model artifacts are divided among the developer roles involved in the CBSE process, i.e., component developers, system architects, system deployers and domain experts.

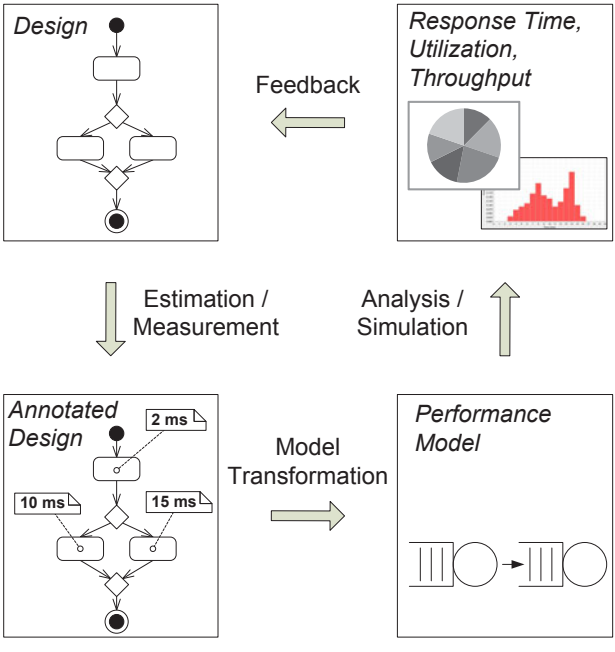


Fig. 3. Model-driven Performance Engineering Process

3.2 Model-to-Model Transformations

To bridge the gap between architecture-level performance models and classical stochastic performance models, over the past decade the SPE community has focused on building automated model-to-model transformations (see Figure 3) which make it possible to exploit existing model solution techniques from the performance evaluation community [39]. In the following, we provide an overview of the most common transformations available in the literature.

Marco and Inverardi transform UML models annotated with SPT stereotypes into a multichain queueing network [38]. UML-ψ, the UML Performance SImulator [40], transforms a UML instance annotated with the SPT profile to a simulation model. The results from the analysis of the simulation model are reported

back to the annotated UML instance [39]. Another approach uses the stochastic process algebra PEPA as analysis model [56]. In this case, only UML activity diagrams are considered, which are annotated with a subset of the MARTE profile. A software tool implementing this method is also available. Bertolino and Mirandola integrate their approach into the Argo-UML modeling tool, using the RT-UML performance annotation profile [8]. An execution graph and a queueing network serve as the target analysis formalisms.

Other approaches use UML, but do not use standardized performance profile annotations: the approach in [18] uses XSLT, the eXtensible Stylesheet Language Transformations, to execute a graph pattern based transformation from a UML instance to LQNs. Instead of annotating the UML model, it has to be modeled in a way so that the transformation can identify the correct patterns in the model. The authors of [7] consider only UML state charts and sequence diagrams. A transformation written in Java turns the model into GSPN sub-models that are then combined into a final GSPN. Gomaa and Menascé use UML with custom XML performance annotation [16]. The performance model is not described in detail, but appears to be based on queueing networks. In [61], the authors use UML component models together with a custom XML component performance specification language. LQN solvers are used for the analysis.

Further approaches exist that are not based on UML: [11,10] builds on the ROBOCOP component model and use proprietary simulation framework for model analysis. [15] proposes a custom control flow graph model notation and custom simulation framework. [20] employs the COMTEK component technology, coupled with a proprietary analysis framework. In [52], the authors specify component composition and performance characteristics using a variant of the big-O notation. The runtime analysis is not discussed in detail.

Several model-to-model transformations have been developed for the Palladio Component Model (PCM). Two solvers are based on a transformation to Layered Queueing Networks (LQNs) [32] and a transformation to Stochastic Regular Expressions [30], respectively. Stochastic Regular Expressions can be solved analytically with very low overhead, however, they only support single user scenarios. Henßproposes a PCM transformation to OMNeT++, focusing on a realistic network infrastructure closer to the OSI reference network model [19]. The PCM-Bench tool comes with the SimuCom simulator [6] which is based on a model-to-text transformation used to generate Java code that builds on DESMO-J, a general-purpose simulation framework. The code is then compiled on-the-fly and executed. SimuCom is tailored to support all of the PCM features directly and covers the whole PCM meta-model. Meier et al. present a transformation of the PCM to Queuing Petri Nets (QPN) [41]. This transformation enables the analysis of PCM model instances with simulation and analysis techniques developed for QPNs [29]. The work also illustrates and compares important aspects concerning the accuracy and overhead of the solvers for PCM model instances (SimuCom, LQNS and LQSim) and SimQPN, the solver for QPNs. The results show that LQN-based solvers are less accurate regarding mean response times and that solvers can have significantly different analysis overhead.

Finally, a number of intermediate languages (or kernel languages) for specifying software performance information have been proposed in the literature. The aim of such efforts is to reduce the overhead for building transformations, i.e., only $M + N$ instead of $M \cdot N$ transformations have to be developed for M source and N target meta-models [39]. Some examples of intermediate languages include SPE-MM [54], KLAPER (Kernel LAnguage for PErformance and Reliability analysis) [17] and CSM (Core Scenario Model) [50].

4 Run-Time Performance Management

With the increasing adoption of virtualization and the transition towards Cloud Computing platforms, modern business information systems are becoming increasingly complex and dynamic. The increased complexity is caused by the introduction of virtual resources and the resulting gap between logical and physical resource allocations. The increased dynamicity is caused by the complex interactions between the applications and services sharing the physical infrastructure. In a virtualized service-oriented environment changes are common during operation, e.g., new services and applications can be deployed on-the-fly, service workflows and business processes can be modified dynamically, hardware resources can be added and removed from the system, virtual machines (VMs) can be migrated between servers, resources allocated to VMs can be modified to reflect changes in service workloads and usage profiles. To ensure adequate performance and efficient resource utilization in such environments, capacity planning needs to be done on a regular basis during operation. This calls for *online* performance prediction mechanisms.

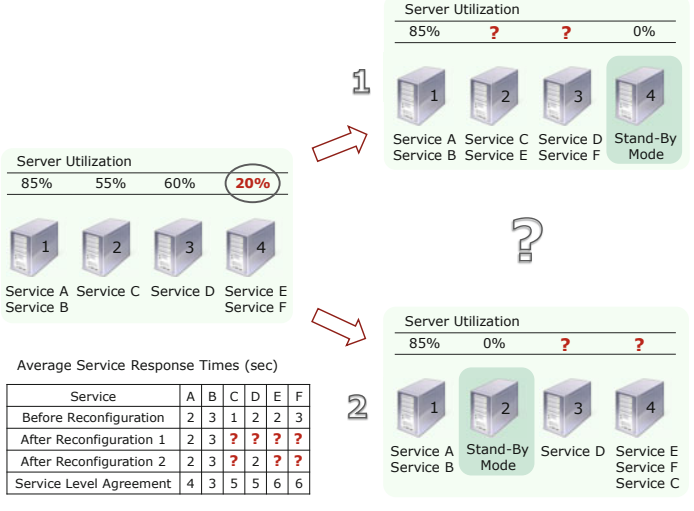


Fig. 4. Online performance prediction scenario

An example of a scenario where online performance prediction is needed is depicted in Figure 4. A service-oriented system made of four servers hosting six different services is shown including information on the average service response times, the response time service level agreements (SLAs) and the server utilization. Now assume that due to a change in the demand for services E and F, the average utilization of the fourth server has dropped down to 20% over an extended period of time. To improve the system's efficiency, it is considered to switch one of the servers to stand-by mode after migrating its services to other servers. Two possible ways to reconfigure the system are shown. To ensure that reconfiguring the system would not break the SLAs, the system needs a mechanism to predict the effect of the reconfiguration on the service response times.

Given the variety of changes that occur in modern service-oriented environments, online performance prediction techniques must support variations at all levels of the system including variations in service workloads and usage profiles, variations in the system architecture, as well as variations in the deployment and execution environment (virtualization, middleware, etc). To predict the impact of such variations, *architecture-level* performance models are needed at run-time that explicitly capture the influences of the system architecture, its configuration, and its workload and usage profiles.

While many architecture-level performance prediction techniques exist in the literature, most of them suffer from two significant drawbacks which render them impractical for use at run-time: i) performance models provide limited support for reusability and customization, ii) performance models are static, creating and maintaining them manually during operation is prohibitively expensive [59].

While techniques for component-based performance engineering have contributed a lot to facilitate model reusability, there is still much work to be done on further parameterizing component models before they can be used for online performance prediction. In particular, current techniques do not provide means to model the layers of the component execution environment explicitly. The performance influences of the individual layers, the dependencies among them, as well as the resource allocations at each layer should be captured as part of the models. This is necessary in order to be able to predict at run-time how a change in the execution environment (e.g., modifying resource allocations at the virtualization layer) would affect the overall system performance.

As to the second issue indicated above, building architecture-level performance models that accurately capture the different aspects of system behavior is a challenging task and requires a lot of time when applied manually to large and complex real-world systems. Often, no explicit architecture documentation of the system exists and hence, the model must be built from scratch. Additionally, experiments and measurements must be conducted to parameterize the model such that it reflects the system behavior accurately.

Current performance analysis tools used in industry mostly focus on profiling and monitoring transaction response times and resource consumption. They often provide large amounts of low-level data while important information about

the end-to-end performance behavior is missing (e.g., service control flow and resource demands).

In research, approaches such as [58,13] use systematic measurements to build black-box mathematical models or models obtained with genetic optimization. However, these approaches are purely measurement-based, the models serve as interpolation of the measurements, and neither a representation of the system architecture nor its performance-relevant factors are extracted. Services are modeled as black boxes and many restrictive assumptions are often imposed such as a single workload class, single-threaded components, homogeneous servers or exponential request interarrival times. Given these limitations, such models are rarely applied in practice and instead ad hoc mechanisms for performance and resource management are employed. Performance-relevant details of the virtualization platform and the applications running inside the hosted virtual machines are not considered explicitly preventing detailed performance predictions which are necessary for efficient resource management. Approaches aiming at the extraction of architectural information are presented in, e.g., [21,22,12]. They use call path tracing, a form of dynamic analysis to gain reliable data on the actual execution of an application. However, the model extraction techniques for architecture-level performance models are usually not automated or applicable at run-time [21,22] and they do not consider the virtualization layer explicitly [12].

Existing work concerning the quantification of virtualization overheads is mainly based on system benchmarking. The performance of several virtualization solutions such as Xen, VMware workstation, Linux-VServer, OpenVZ, etc. are compared. However, the focus is on the overall performance overhead, individual performance-influencing factors are not analyzed. In contrast, Lu et al. [36] present a calibration process based on application usage traces that covers the main resource types CPU, memory, network and disk I/O and is applicable at run-time. Approaches to automatically extract performance models of the virtualization layer are normally very specific and do not provide models which can be used for performance predictions at the application level. Therefore, there is a major need for an approach which addresses these deficiencies and combines methods for automated model extraction and performance prediction in virtualized environments at run-time.

The heart of the problem is in the fact that architecture-level performance models are normally designed for offline use and as such they are decoupled from the system components they represent. Models do not capture dynamic aspects of the environment and therefore they need to be updated manually after every change in the system configuration or workload. Given the frequency of such changes, the amount of effort involved in maintaining performance models is prohibitive and therefore in practice such models are rarely used after deployment.

Research Challenges. The described limitations of the existing work lead to the following two general research challenges:

– Designing abstractions for modeling performance-relevant aspects of services in *dynamic virtualized* environments. The abstractions should be structured

around the system components involved in processing service requests. Individual layers of the software architecture and execution environment, context dependencies and dynamic system parameters should be modeled explicitly.
– Developing methods for the automatic online model extraction, maintenance, refinement and calibration *during operation*. This includes the efficient resolution of service context dependencies including dependencies between service input parameters, resource demands, invoked third-party services and control flow of underlying components.

To address these challenges, current research efforts are focusing on developing *online* architecture-level performance models designed specifically for use at run-time, e.g., the Descartes Meta-Model [14,27]. Such models aim at capturing all information, both static and dynamic, relevant to predicting the system's performance on-the-fly. They are intended to be integrated into the system components and to be maintained and updated automatically by the underlying execution platform (virtualization and middleware) reflecting the evolving system environment.

Online performance models will make it possible to answer performance-related queries that arise during operation such as: What would be the effect on the performance of running applications if a new application is deployed in the virtualized infrastructure or an existing application is migrated from one physical server to another? How much resources need to be allocated to a newly deployed application to ensure that SLAs are satisfied? How should the system configuration be adapted to avoid performance issues or inefficient resource usage arising from changing customer workloads?

The ability to answer queries such as the above provides the basis for implementing techniques for *self-aware* performance and resource management [27]. Such techniques will be triggered automatically during operation in response to observed or forecast changes in application workloads. The goal will be to *proactively* adapt the system to such changes in order to avoid anticipated QoS problems or inefficient resource usage. The adaptation will be performed in an autonomic fashion by considering a set of possible system reconfiguration scenarios (e.g, changing VM placement and/or resource allocations) and exploiting the online performance models to predict the effect of such reconfigurations before making a decision.

Self-aware systems engineering [14,26] is currently emerging as a new research area at the intersection of several computer science disciplines including software architecture, computer systems modeling, autonomic computing, distributed systems, and more recently, Cloud Computing and Green IT. It raises a number of big challenges that represent emerging hot topics in the systems engineering community and will be subject of long-term fundamental research in the years to come. The resolution of these challenges promises to revolutionize the field of systems engineering by enabling guaranteed QoS, lower operating costs and improved energy efficiency.

5 Concluding Remarks

We presented a historical perspective on the evolution of model-based performance engineering techniques for business information systems, focusing on the major developments over the past four decades that have shaped the field, such as the increasing integration of software-related aspects into performance models, the increasing parametrization of models to foster model reuse, the increasing use of automated model-to-model transformations to bridge the gap between models at different levels of abstraction, and finally the increasing use of models at run-time for online performance management. We surveyed the state-of-the-art on performance modeling and management approaches discussing the ongoing efforts in the community to increasingly bridge the gap between high-level business services and low level performance models. Finally, we concluded with an outlook on the emergence of self-aware systems engineering as a new research area at the intersection of several computer science disciplines.

References

1. Computing missing service demand parameters for performance models. In: CMG 2008, pp. 241–248 (2008)
2. Banks, J., Carson, J.S., Nelson, B.L., Nicol, D.M.: Discrete-Event System Simulation, 3rd edn. Prentice Hall, Upper Saddle River (2001)
3. Bard, Y., Shatzoff, M.: Statistical methods in computer performance analysis. Current Trends in Programming Methodology, III (1978)
4. Bause, F.: Queueing Petri Nets - A formalism for the combined qualitative and quantitative analysis of systems. In: Proceedings of the 5th International Workshop on Petri Nets and Performance Models, Toulouse, France, October 19-22 (1993)
5. Bause, F., Kritzinger, F.: Stochastic Petri Nets - An Introduction to the Theory, 2nd edn. Vieweg Verlag (2002)
6. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. Journal of Syst. and Softw. 82, 3–22 (2009)
7. Bernardi, S., Donatelli, S., Merseguer, J.: From UML sequence diagrams and statecharts to analysable petri net models. In: Proc. on WOSP 2002, pp. 35–45 (2002)
8. Bertolino, A., Mirandola, R.: CB-SPE Tool: Putting Component-Based Performance Engineering into Practice. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) CBSE 2004. LNCS, vol. 3054, pp. 233–248. Springer, Heidelberg (2004)
9. Bolch, G., Greiner, S., Meer, H.D., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, 2nd edn. John Wiley & Sons, Inc. (April 2006)
10. Bondarev, E., de With, P., Chaudron, M., Muskens, J.: Modelling of input-parameter dependency for performance predictions of component-based embedded systems. In: Proc. on EUROMICRO 2005, pp. 36–43 (2005)
11. Bondarev, E., Muskens, J., de With, P., Chaudron, M., Lukkien, J.: Predicting real-time properties of component assemblies: a scenario-simulation approach. In: Proc. of the 30th Euromicro Conference, pp. 40–47 (2004)

12. Brosig, F., Huber, N., Kounev, S.: Automated Extraction of Architecture-Level Performance Models of Distributed Component-Based Systems. In: 26th IEEE/ACM International Conference On Automated Software Engineering (ASE 2011), Oread, Lawrence, Kansas (November 2011)
13. Courtois, M., Woodside, M.: Using regression splines for software performance analysis. In: Proceedings of the International Workshop on Software and Performance (2000)
14. Descartes Research Group (December 2011), http://www.descartes-research.net
15. Eskenazi, E., Fioukov, A., Hammer, D.: Performance Prediction for Component Compositions. In: Crnković, I., Stafford, J.A., Schmidt, H.W., Wallnau, K. (eds.) CBSE 2004. LNCS, vol. 3054, pp. 280–293. Springer, Heidelberg (2004)
16. Gomaa, H., Menascé, D.: Performance Engineering of Component-Based Distributed Software Systems. In: Dumke, R.R., Rautenstrauch, C., Schmietendorf, A., Scholz, A. (eds.) WOSP 2000 and GWPESD 2000. LNCS, vol. 2047, pp. 40–55. Springer, Heidelberg (2001)
17. Grassi, V., Mirandola, R., Sabetta, A.: Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. Journal of Systems and Software 80(4), 528–558 (2007)
18. Gu, G.P., Petriu, D.C.: XSLT transformation from UML models to LQN performance models. In: Proc. on WOSP 2002, pp. 227–234 (2002)
19. Henss, J.: Performance prediction for highly distributed systems. In: Proc. on WCOP 2010, vol. 14, pp. 39–46. Karlsruhe Institue of Technology (2010)
20. Hissam, S., Moreno, G., Stafford, J., Wallnau, K.: Packaging Predictable Assembly. In: Bishop, J.M. (ed.) CD 2002. LNCS, vol. 2370, pp. 108–124. Springer, Heidelberg (2002)
21. Hrischuk, C.E., Woodside, M., Rolia, J.A., Iversen, R.: Trace-Based Load Characterization for Generating Performance Software Models. IEEE Trans. on Softw. Eng. (1999)
22. Israr, T., Woodside, M., Franks, G.: Interaction tree algorithms to extract effective architecture and layered performance models from traces. J. Syst. Softw. (2007)
23. Kounev, S.: Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction. PhD Thesis. Shaker Verlag (December 2005)
24. Kounev, S.: Performance Modeling and Evaluation of Distributed Component-Based Systems using Queueing Petri Nets. IEEE Transactions on Software Engineering 32(7), 486–502 (2006)
25. Kounev, S.: Software Performance Evaluation. In: Wiley Encyclopedia of Computer Science and Engineering, Wiley-Interscience, John Wiley & Sons Inc. (September 2008) ISBN-10: 0471383937, ISBN-13: 978-0471383932
26. Kounev, S.: Self-Aware Software and Systems Engineering: A Vision and Research Roadmap. GI Softwaretechnik-Trends 31(4) (November 2011)
27. Kounev, S., Brosig, F., Huber, N., Reussner, R.: Towards self-aware performance and resource management in modern service-oriented systems. In: Proc. of the 7th IEEE Intl. Conf. on Services Computing (SCC 2010). IEEE Computer Society (2010)
28. Kounev, S., Buchmann, A.: Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets. In: Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software (2003)

29. Kounev, S., Buchmann, A.: SimQPN - a tool and methodology for analyzing queueing Petri net models by means of simulation. Performance Evaluation 63(4-5), 364–394 (2006)
30. Koziolek, H.: Parameter dependencies for reusable performance specifications of software components. PhD thesis, University of Karlsruhe, TH (2008)
31. Koziolek, H.: Performance evaluation of component-based software systems: A survey. Performance Evaluation 67(8), 634–658 (2009)
32. Koziolek, H., Reussner, R.: A Model Transformation from the Palladio Component Model to Layered Queueing Networks. In: Kounev, S., Gorton, I., Sachs, K. (eds.) SIPEW 2008. LNCS, vol. 5119, pp. 58–78. Springer, Heidelberg (2008)
33. Kumar, D., Tantawi, A., Zhang, L.: Real-time performance modeling for adaptive software systems. In: VALUETOOLS 2009: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, pp. 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels (2009)
34. Law, A., Kelton, D.W.: Simulation Modeling and Analysis, 3rd edn. McGraw Hill Companies, Inc. (2000)
35. Liu, Z., Wynter, L., Xia, C.H., Zhang, F.: Parameter inference of queueing models for IT systems using end-to-end measurements. Performance Evaluation 63(1), 36–60 (2006)
36. Lu, L., Zhang, H., Jiang, G., Chen, H., Yoshihira, K., Smirni, E.: Untangling mixed information to calibrate resource utilization in virtual machines. In: Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC 2011, pp. 151–160. ACM, New York (2011)
37. MacNair, E.A.: An introduction to the Research Queueing Package. In: WSC 1985: Proceedings of the 17th Conference on Winter Simulation, pp. 257–262. ACM Press, New York (1985)
38. Di Marco, A., Inverardi, P.: Compositional generation of software architecture performance QN models. In: Working IEEE/IFIP Conf. on Software Architecture, p. 37 (2004)
39. Di Marco, A., Mirandola, R.: Model Transformation in Software Performance Engineering. In: Hofmeister, C., Crnković, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, pp. 95–110. Springer, Heidelberg (2006)
40. Marzolla, M., Balsamo, S.: UML-PSI: The UML performance simulator. Quantitative Eval. of Syst., 340–341 (2004)
41. Meier, P., Kounev, S., Koziolek, H.: Automated Transformation of Palladio Component Models to Queueing Petri Nets. In: 19th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2011), Singapore, July 25-27 (2011)
42. Menascé, D., Almeida, V.: Capacity Planning for Web Performance: Metrics, Models and Methods. Prentice Hall, Upper Saddle River (1998)
43. Menascé, D., Almeida, V.: Scaling for E-Business - Technologies, Models, Performance and Capacity Planning. Prentice Hall, Upper Saddle River (2000)
44. Menascé, D., Almeida, V., Fonseca, R., Mendes, M.: A Methodology for Workload Characterization of E-commerce Sites. In: Proceedings of the 1st ACM Conference on Electronic Commerce, Denver, Colorado, United States, pp. 119–128 (November 1999)
45. Menascé, D.A., Almeida, V., Dowdy, L.W.: Capacity Planning and Performance Modeling - From Mainframes to Client-Server Systems. Prentice Hall, Englewood Cliffs (1994)

46. Menascé, D.A., Almeida, V., Dowdy, L.W.: Performance by Design. Prentice Hall (2004)
47. Menascé, D.A., Gomaa, H.: A Method for Desigh and Performance Modeling of Client/Server Systems. IEEE Transactions on Software Engineering 26(11) (November 2000)
48. Mohr, J., Penansky, S.: A forecasting oriented workload characterization methodology. CMG Transactions 36 (June 1982)
49. Pacifici, G., Segmuller, W., Spreitzer, M., Tantawi, A.: CPU demand for web serving: Measurement analysis and dynamic estimation. Performance Evaluation 65(6-7), 531–553 (2008)
50. Petriu, D., Woodside, M.: An intermediate metamodel with scenarios and resources for generating performance models from UML designs. Software and Systems Modeling (SoSyM) 6(2), 163–184 (2007)
51. Rolia, J., Vetland, V.: Parameter estimation for performance models of distributed application systems. In: CASCON 1995: Proceedings of the 1995 Conference of the Centre for Advanced Studies on Collaborative Research, p. 54. IBM Press (1995)
52. Sitaraman, M., Kulczycki, G., Krone, J., Ogden, W.F., Reddy, A.L.N.: Performance specification of software components. SIGSOFT Softw. Eng. Notes 26(3), 3–10 (2001)
53. Smith, C.U.: Performance Engineering of Software Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (1990)
54. Smith, C.U., Lladó, C.M., Cortellessa, V., Di Marco, A., Williams, L.G.: From UML models to software performance results: an SPE process based on XML interchange formats. In: WOSP 2005: Proceedings of the 5th International Workshop on Software and Performance, pp. 87–98. ACM Press, New York (2005)
55. Smith, C.U., Williams, L.G.: Performance Solutions - A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley (2002)
56. Tribastone, M., Gilmore, S.: Automatic extraction of PEPA performance models from UML activity diagrams annotated with the MARTE profile. In: Proc. on WOSP 2008 (2008)
57. Trivedi, K.S.: Probability and Statistics with Reliability, Queueing and Computer Science Applications, 2nd edn. John Wiley & Sons, Inc. (2002)
58. Westermann, D., Happe, J.: Towards performance prediction of large enterprise applications based on systematic measurements. In: WCOP (2010)
59. Woodside, M., Franks, G., Petriu, D.: The future of software performance engineering. In: Future of Software Engineering (FOSE 2007), pp. 171–187. IEEE Computer Society, Los Alamitos (2007)
60. Woodside, M., Neilson, J., Petriu, D., Majumdar, S.: The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-Like Distributed Software. IEEE Transactions on Computers 44(1), 20–34 (1995)
61. Wu, X., Woodside, M.: Performance modeling from software components. SIGSOFT Softw. E. Notes 29(1), 290–301 (2004)
62. Zhang, L., Xia, C.H., Squillante, M.S., Iii, W.N.M.: Workload service requirements analysis: A queueing network optimization approach. In: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS 2002, p. 23. IEEE Computer Society, Washington, DC (2002)
63. Zheng, T., Woodside, C.M., Litoiu, M.: Performance model estimation and tracking using optimal filters. IEEE Transactions on Software Engineering 34(3), 391–406 (2008)

Enabling Enterprise Collaboration Using Service Source Descriptions

Brahmanadna Sapkota and Marten van Sinderen

University of Twente, The Netherlands
{b.sapkota,m.j.vansinderen}@utwente.nl

Abstract. We witness an increasing need for cross-enterprise collaboration to respond to continuous changes in market demands and opportunities. Approaches based on Web services have been proposed, mainly because they provide standards-based service abstractions which can be combined into business-oriented functions. Collaboration requirements are expressed as business processes representing the coordination of several services whose descriptions are published in service registries. Current service registries are passive and, consequently, the published service descriptions may become readily outdated. In this paper, we present an approach to guarantee the freshness and correctness of published service descriptions. Unlike existing approaches, which use a service description for each individual service, our approach uses so-called service source descriptions which specify the collection of service functionalities and contain only the information required for finding services. This description is published in the public service registry to facilitate discovery of service providers whereas the detailed information remains at the source.

Keywords: Service Oriented Architecture, Web services, Enterprise Application Integration, Cross-organisational Collaboration, Business Requirements, Business Processes, Business-IT alignment.

1 Introduction

In order to respond effectively and efficiently to changes in market demands and to seize opportunities that inevitably occur due to such changes, enterprise collaboration has become increasingly important. Service Oriented Architecture (SOA) [1] has been widely recognized as one of the enablers of enterprise collaboration because of its support for service abstractions. SOA provides flexible support for enterprise collaboration based on service abstractions and Internet standards. Collaboration requirements are expressed as business processes, which represent the coordination of several published services as well as the implementation of a composite value-added service. In this way, enterprise functions are aggregated using multi-stage business processes fulfilling the specific requirements of an enterprise [2]. In addition to supporting enterprise function aggregation, such solutions must be flexible enough to allow adaptation in a timely

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 38–55, 2012.

and cost-effective manner to support continuously evolving enterprise collaboration. SOA based technologies such as Web services [3] support adaptation of such an evolution through the concept of service composition.

Core elements of service composition for supporting enterprise collaboration are the discovery of services and the exchange of information between these services. In reaslising the enterprise functions, suitable services are searched and selected based on the published information on the available services. The service discovery thus plays a central role in enterprise collaboration and application integration. Service discovery is performed based on the notion of service description and service registry. In general, service descriptions are used for specifying: 1) what functionalities are offered by the service to its users (i.e., the interface definition); 2) how the service is provided (i.e., the service binding); and 3) where the service can be accessed (i.e., the service endpoint information). The service registries are thus used to announce the offered services. In a typical scenario, service providers publish their service descriptions to a publicly accessible service registry. The service users search over these registries and find the information needed to select and use the required services. This approach of publishing, finding and using the services forms the so called SOA triangular operational model which clearly separates the role of service provider, service user and service registry [4]. SOA, thus, enables flexible on-demand enterprise collaboration and application integration. Despite these sound principles of SOA, a number of practical complexities still exists, which are preventing enterprises to fully exploit the potential benefits of SOA.

With the increase in user-centric service provisioning, enterprises are increasingly required to engage in on-demand collaborations. An on-demand collaboration can only be achieved if 1) the participants required for the collaboration can be found when needed and 2) the participants can communicate with each other. In the SOA based approach, the former requirement is supported by defining the service functionality and the later requirement is supported by defining the messages being sent. These definitions are combined together and are published as service description. This can be realised relatively easily by using pre-meditated message structures and function libraries, if the enterprises collaborate in a closed environment. If autonomous enterprises are to collaborate on-demand, pre-meditated message structures or function libraries cannot be used. Therefore, semantics of the information provided through the service descriptions should be well defined and the service registry should contain valid service descriptions. While the initiatives around Semantic Web services have defined formalisms such as WSMO [5], OWL-S [6] and SAWSDL [7], to define service descriptions semantically, they cannot ensure that the published descriptions correctly reflect the offered services at the time of their discovery. Therefore, a mechanism is required for ensuring the correctness of the service descriptions during the complete life cycle of SOA-based enterprise collaboration and application integration.

In this paper, we present a new approach to guarantee the correctness and freshness of the published service descriptions. Unlike in the existing approaches, which use a service description for each individual service, the proposed approach uses so-called service source descriptions. A service source description specifies the collection of service functionalities. It specifies what type of services are offered by a particular service provider. The service source description clearly separates the detailed information required for invoking a service and the more coarse information required for finding a service that can be provided by a specific service provider. The service source description is encoded following the LinkedData principles [8], which encourages storage of data at its source. In the proposed approach, we assume that the services are described using Web Service Definition Language [9] and stored in the local repositories of the service providers. We also assume that these repositories are exposed as services and can be queried to retrieve detailed information required to invoke a particular service. With these assumptions, we provide a mechanism to generate a service source description based on the WSDL documents stored at the local service registry of a service provider. Each service source description is published in the public service registry to facilitate discovery of service providers whereas the detailed service descriptions remain at their sources relieving the service providers of the burden of updating the published descriptions when corresponding services are changed due to market developments. Though a service provider still has to generate and publish the service source descriptions (based on the individual service descriptions at the source, which in turn are manually provided), it can be automated.

The major design goals of the proposed approach is simplicity and extensibility. The service source description is essentially the collection of summaries of the WSDL files stored in the local registry of the provider which simplifies the process of service description updates. The service providers need to update only the local registry. If the new service providers arise, they can simply publish the service source description to the public service registry. We do not introduce a new technology but built upon existing technologies following the SOA triangular operational model except that service source descriptions are published to the service registry instead of publishing the individual service descriptions.

The rest of the paper is structured as follows: Background technologies based on which the solution proposed in this paper is developed is briefly discussed in Section 2. The service discovery challenges are discussed in Section 3. The proposed solution is presented in Section 4 and its use to support cross-organisational collaboration is presented in Section 5. Some of the relevant existing works are discussed in Section 6. Finally, the work presented in this paper is concluded in Section 7 by highlighting possible future directions.

2 Background

We briefly introduce the tools and technologies based on which the solution presented in this paper is developed.

2.1 Web Service Description Language

WSDL is a commonly used XML-based language for describing Web services. As shown pictorially in Figure 1, the newer version of WSDL, i.e., WSDL 2.0 is structured into five main elements which are *description*, *interface*, *service*, *binding*, and *types*. The *description* element is used to specify the root element of the WSDL file and contains the remaining four elements as its sub elements. The *interface* element is used to describe operations that a web service provides, messages that these operations consume or produce, and the mechanisms for fault handling. The *service* element describes a collection of endpoints (i.e., URLs) specifying the location of invoking the service operations. The *binding* element describes how to invoke the service by specifying required communication protocols. The data types used in messages exchanged between the user and the Web service as well as in the fault messages are specified in the *types* element.

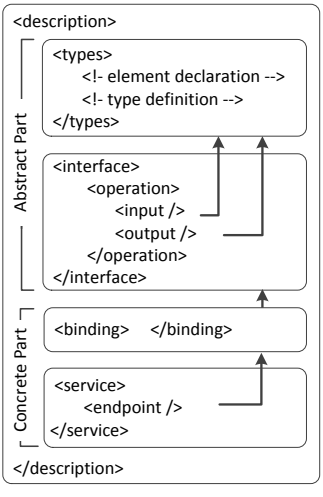


Fig. 1. WSDL 2.0 document structure

In addition, WSDL 2.0 also contains *documentation* and *import* elements which can be used to specify additional textual information regarding the service and to use externally defined XML Schema respectively. A WSDL document contains both the abstract information describing the functionality offered by a service and the concrete information describing how to use these functionalities. The service functionality is described in terms of the messages it sends and receives. These messages are associated to an operation which are grouped together under an interface. The concrete information required for using the functionality of a service is described in terms of transport protocol binding for interfaces.

2.2 Resource Description Framework

Resource Description Framework (RDF) [10] is an application neutral data model for describing knowledge that needs to be shared between applications. In RDF, information is presented in a directed labeled graphs as shown in Figure 2. In this model, nodes represent resources or values and arrows represent the relationship between the connected nodes. In the graph, ovals are used to denote

Fig. 2. Example RDF Graph

resources whereas rectangles are used to represent values. Two nodes connected with an arrow is called an RDF statement in which the nodes connected to the tail and the head of the arrow are called subject and object of the statement respectively whereas the label of the arrow is called its predicate. Therefore, RDF statements contain triples and are encoded in the form <*subject*, *predicate*, *object*>. This graph-based data model can be serialised in different formats such as RDF/XML, Notation 3, and Turtle. Using Turtle format, Listing 1 presents same information given in Figure 2

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .

<http://www.utwente.nl/ewi/is/>
    dc:title "Information Systems Group" ;
    dc:publisher "University of Twente" .
```

Listing 1. Example RDF graph in Turtle format

Though RDF is originally designed to describe Web resources, it can be used to describe anything with URIs.

2.3 Linked Data

Linked Data [8] is a set of guidelines or principles defined for publishing data on the Web. It advocates the use of URI as the identifier of published data and RDF as the model for describing the data. It also encourages to include URIs pointing to the external information such that the relevant information could be looked upon when needed. Because the data is described using RDF, the standard query language SPARQL [11] can be used to retrieve the useful information. In addition, the use of URI for linking different information allows applications to navigate through different sources to discover the required information. Figure 3 illustrates interconnection of data from three different sources. In the figure, the dotted arrows indicate that the information stored in different sources are indicated because of the use of same URIs. One of the benefits of such an interconnection is the better reuse and lesser redundancies of data.

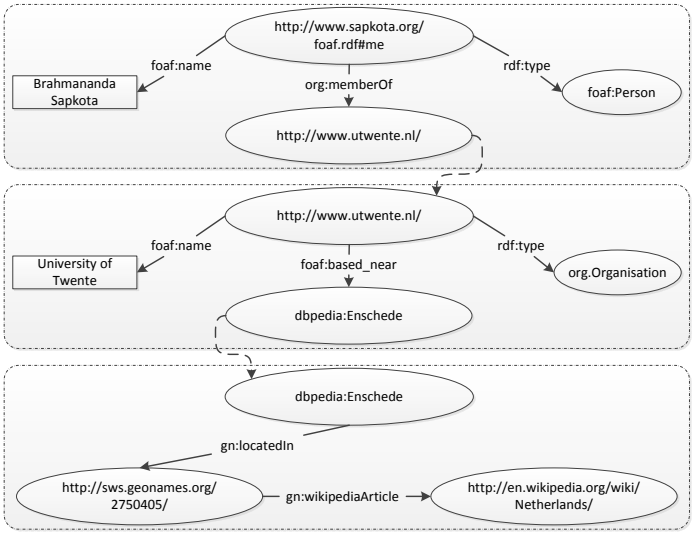


Fig. 3. Example Linked Data

The URIs used in connecting different sources are called RDF links. A RDF link is a RDF triple where the subject is a URI reference in the namespace of one source, the predicate and or the object are URI references pointing to the namespaces of other source. Therefore RDF links have types. Such a typed link allows for describing the relationship between the sources that are connected. A RDF link of type *org:memberOf*, for example, may be used to connect a person to an organisation. It makes integration of data from different sources easier because of the connection that a RDF link makes between different namespaces. In the example shown in Figure 3, since the link www.utwente.nl appears in a triple *<http://sapkota.org/foaf.rdf#me org:memberOf http://www.utwente.nl>*, anyone looking for information about *http://sapkota.org/foaf.rdf#me* can navigate through the link and get extra information such as Brahmananda Sapkota works at the University of Twente which is located in Enschede and Enschede is in the Netherlands. The Linked Data, thus provides a simple and flexible mechanism for integrating information from different sources.

3 Service Discovery Challenges

In an open environment, it is difficult to support on-demand collaboration if the published service descriptions are outdated consequently providing incorrect information. This difficulty escalates when service descriptions contain limited information, i.e., some information may be relevant for the discovery of services, but since this information depends on the runtime state, it cannot be included in the service descriptions [12,13]. So, besides the "correctness" (or "outdated

information") problem, we also have the "state-dependency" (or "dynamic information") problem that leads to a poor discovery results.

The correctness problem arises due to the fact that service registries are passive. If the functionalities of the offered services are changed, service providers should take the initiative to update the corresponding descriptions published in the service registry. In practice, the published descriptions are rarely updated. Instead of publishing the service descriptions to the service registries, they are published on the Web [14]. Such a practice, violates the original SOA model (i.e., the triangular operational model) and consequently undermines the role of service registries in SOA [15]. This shift in practice is due to the lack of efficient and elegant support to update a service description in the service registry whenever the corresponding service is changed. The latter essentially requires an additional effort on the part of the service providers. It becomes more problematic because new service registries emerge and the existing one disappear [16] and therefore service registries cannot be fully relied upon for service discovery. The results of the investigation of Web services on the Web published in [17] reveals that only around 63% of the discovered Web services are in fact active. This lack of reliable and persistent service repositories makes on-demand collaboration between autonomous enterprises difficult, if not impossible.

Besides these technical difficulties, there are other reasons why public service registries have not been successful so far. One of the reasons is that a considerable amount of the published service descriptions are unusable [12]. Some of the information (e.g., the information that depends on the change in state) which might be important for discovery purposes cannot be included in the service descriptions in a simple way. In addition, some providers may not be willing to disclose information related to non-functional properties because of the fear of bargain or competition from other providers [13]. This introduces the "sensitive information" problem and contributes to the retrieval of imprecise or at least incomplete service descriptions leading to false positives and consequently reducing the usability and the reliability of service registries.

The sensitive information problems could be resolved if a service provider is allowed to store its service descriptions locally. The locally stored descriptions can then be requested at the time of service discovery. If the service descriptions are stored locally, the correctness and state-dependency problems can also be tackled since any changes on the service can be quickly updated locally. This can be achieved, if service providers are equipped with a tool that extract summary information of the offered functionalities from their own registry, builds an service source description and publishes it a public service registry. In this way, only the part of the description that is needed to find the required functionality is published in the public registry whereas the part of the description which specifies what detailed information is needed for invoking the service is kept in the local registry.

4 Service Discovery Solution

We describe a two step mechanism for the publication of service source description and the discovery of services. In the first step, we generate the service source description and publish it to a public service registry. In the second step, we use the published information to discover required services. Since the abstract information is sufficient to discover services [18], we only include this information in the service source description. The overall approach for publishing the service source description is depicted in Figure 4.

The benefits of this approach are: 1) service providers do not have to publish all the information. It will also reduce the extra effort required for maintaining and updating the service registry. Service providers do not need to update service registry every time the service description is updated, updates are done locally. 2) service users can use these SSP descriptions to find the potential providers, and finally obtain the up-to-date information.

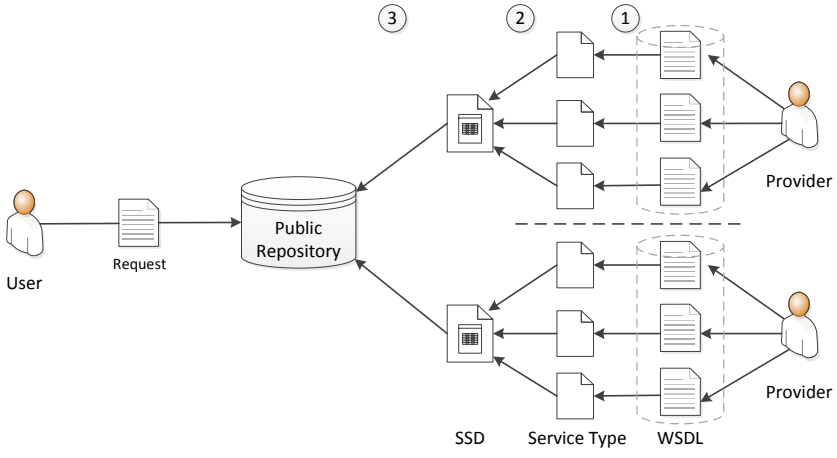


Fig. 4. Overall approach for publishing service source descripting

As indicated in the figure, first we extract the service type information form the WSDL files stored in the service provider's local registry. Then we use this information to generate service source description. Finally, the service source description is published in the public service registry. These steps and their results are presented in detail in the following subsections.

4.1 Service Types

One of the purposes of using service type information, in the proposed approach, is to provide indication of what types of services are offered by a service provider. It is generally the case that a service provider does not necessarily provide one type of

service. This kind of information serves the purpose of *guiding* service requests towards the potential service providers. This approach is expected to help in narrowing down the search space and allowing applications to deal with the ever increasing number of Web services. We extract this information from service descriptions encoded in WSDL, which is a commonly used service description language.

In WSDL, *types* element specifies the data type of input output messages in terms of XML Schema definition. The *types* element essentially models the domain knowledge and is thus used for creating local domain ontology. We use XML Schema to RDF Schema [19] mapping techniques presented in [20] and [21] to generate local domain model. Using their mapping techniques, given an input shown in Listing 2, we obtain the semantic domain model shown in Listing 3.

```
<wsdl:types>
 <xsd:schema targetNamespace="http://org.example.com/resources/AvailabilityService">
  <xsd:element name="ServiceRequest">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="product" type="xsd:string"/>
        <xsd:element name="date" type="xsd:string"/>
        <xsd:element name="quantity" type="xsd:float"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="ServiceResponse" type="response"/>
  <xsd:simpleType name="response">
    <xsd:restriction base="xsd:boolean"/>
  </xsd:simpleType>
  </xsd:schema>
</wsdl:types>
```

Listing 2. WSDL Types definition

Representing the data type definitions in RDF has several advantages. RDF provides an abstract model for describing resources with properties, scoping them to a particular application domain through RDF Schema and defining relationships between these resources. In addition, standard RDF query language SPARQL [11] can be used to provide flexible means to allow users to express their requests.

```
<rdfs:Class rdf:ID="http://org.example.com/resources/AvailabilityService#ServiceRequest"/>
<rdfs:Class rdf:ID="http://org.example.com/resources/AvailabilityService#ServiceResponse"/>

<rdf:Property rdf:about="http://org.example.com/resources/AvailabilityService#response">
  <rdfs:domain rdf:resource="http://org.example.com/resources/AvailabilityService#ServiceResponse"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</rdf:Property>
<rdf:Property rdf:ID="http://org.example.com/resources/AvailabilityService#product">
  <rdfs:domain rdf:resource="http://org.example.com/resources/AvailabilityService#ServiceRequest"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Property>
<rdf:Property rdf:ID="http://org.example.com/resources/AvailabilityService#date">
  <rdfs:domain rdf:resource="http://org.example.com/resources/AvailabilityService#ServiceRequest"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</rdf:Property>
<rdf:Property rdf:ID="http://org.example.com/resources/AvailabilityService#quantity">
  <rdfs:domain rdf:resource="http://org.example.com/resources/AvailabilityService#ServiceRequest"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</rdf:Property>
```

Listing 3. RDF representation of WSDL types

In Listing 3, the attribute *data*, for example, is represented as the property of the class *ServiceRequest* whose value is of the type string.

4.2 Service Source Description

A service source description models the service provider as a collection of services and enumerates all the offered services. In particular, the service source description specifies the types of services that are offered by a particular service provider. These descriptions provide information sufficient enough to filter out the completely irrelevant service providers. In WSDL, as shown in Listing 4, the *interface* element specifies the operations that can be invoked from the service. We use these information from a given WSDL document to specify the functionality of a service in the service source description. We use the service types information to describe domain specific concepts and to avoid ambiguities between services from different application domains.

```
<wsdl:description
  targetNamespace="http://org.example.com/services/AvailabilityService/"
  xmlns="http://org.example.com/services/AvailabilityService/"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  .
  .
  <wsdl:interface name="AvailabilityServiceInterface">
    <wsdl:operation name="RequestOperation" pattern="http://www.w3.org/ns/wsdl/in-out">
      <wsdl:input element="ServiceRequest"/>
      <wsdl:output element="ServiceResponse"/>
    </wsdl:operation>
  </wsdl:interface>
  .
  .
</wsdl:description>
```

Listing 4. WSDL Interface definition

We define a service source description as <n, e, Q>, where n is the URL of the service provider, e is the endpoint of the service provider through with the detailed service description can be queried and Q is a collection of (<op(i, o), l>) pairs, where i and o represents the input and output types associated with the operation op whereas l represents the interface to which the operation op is associated with. This combination of interface and operation allows one to determine through which interface a particular operation of service is made available and is sufficient to look for a corresponding WSDL file from the local registry of the service provider. Listing 5 illustrates a service source description containing a service definition shown in Listing 4.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://www.example.org/ontology#> .
@prefix dc: <http://www.w3.org/2001/XMLSchema#> .
@prefix p: <http://org.example.com/ontology#> .

<http://org.example.com> rdf:type ex:ServiceProvider ;
    ex:localRegistry <http://org.example.com/services/sparql> ;
    ex:functionality [ ex:operation p:RequestOperation> ;
                       ex:input   p:ServiceRequest> ;
                       ex:output  p:ServiceResponse> ;
                       ex:interface p:AvailabilityServiceInterface> ].
```

Listing 5. Example service source description

We assume that the service provider allows to query its local registry during the discovery process through a web service endpoint. This endpoint is linked in

the service source description via *localRegistry* property. The number of instances of functionalities provided by a service provider in its local registry is typically far larger than the number of distinct interfaces. Inclusion of all instances in the service source description is therefore likely to increase its size as close to as the size of the local registry. In order to avoid such problems, the service source description includes only the functionality types and not their instances. This allows to select the relevant service provider based on this type information. For example, if there are a large number of printing service providers and only few of them are providing book printing services it is much more efficient to send a book printing service requests only to those that provide the book printing services. This type of service source description will be helpful in selecting the service providers in situations where fine grained information is needed for answering the service requests.

4.3 Finding Potential Services

In order to find the required service we follow two step discovery mechanism. In the first step, the discovery request is sent to the service registry from which a list of potential service providers are identified. In the second step, direct communication with the potential service providers is established to find the most appropriate service provider. The goal of the first step is to reduce the search space whereas the second step is intended to select the right service at the most appropriate service provider based on up-to-date information.

We assume that the service providers employ the mechanisms discussed above for extracting the information needed for generating service source descriptions. In order to find the potential services, the service requester initiates the lookup over service source descriptions. The required functionalities are matched against the functionalities offered by different service providers. The endpoints of the service providers whose offer matching functionality is returned to the user. In this step, only the functionality of the service is queried and hence it cannot be expected that the service indeed matches the request. Therefore, detailed information is required for finding the required service. The request is then issued to these endpoints so that the detailed information can be obtained. The overall approach for finding the potential services is illustrated in Figure 5.

When the detailed information is obtained, existing service discovery mechanism can be applied to find the potential services. When a list of potential services are found, an appropriate service can be selected using the existing service selection mechanism. In the figure, we showed that the local registry of only one service provider is queried to find the detailed information. It was only for the illustration purposes and there can be multiple service providers and it is possible that the local registry of more than one service provider is queried.

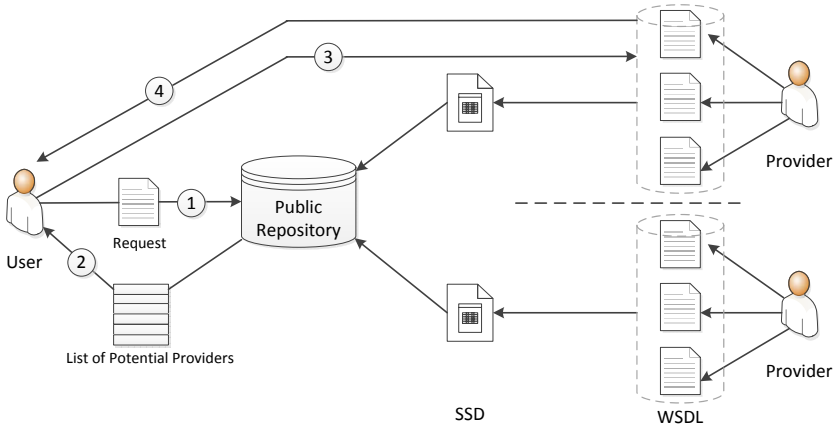


Fig. 5. Overall approach for service discovery

5 Enterprise Collaboration

Let us now return to our original goal of facilitating enterprise collaboration using SOA. We previously concluded that the SOA architectural triangle with its 'publish-find-use' paradigm is in principle very convenient to enterprises to utilize distributed capabilities that may be under the control of different ownership domains. The convenience stems from the loose coupling of services - supporting flexible composition - and the external-oriented representation of services - allowing interactions between users and providers irrespective of their internal implementation. However, we also concluded that the 'publish-find' part of the triangle has practical limitations, which so far has prevented the successful uptake of public-registry/open-discovery based enterprise collaboration. The main limitations are: (a) it is hard to find and compose services based on current service descriptions, since the descriptions lack unambiguous and precise semantics; (b) it is expensive and laborious to maintain service descriptions, as the corresponding services continuously evolve and therefore the descriptions require frequent and manually managed updates; (c) trust is a hindrance for publishing service descriptions and using services discovered with public registries.

We propose to leverage the publish-find-use paradigm by using public descriptions that are automatically generated and semantically enhanced, as described in Section 4. In the following, we first show which interactions are necessary for enterprise collaboration using our approach, and subsequently discuss the potential benefits of our approach.

Figure 6 illustrates the basic interactions:

1. A business organisation acting as a service provider can publish relevant information on all its services using a single keyword-based description (service source description) at a service broker that maintains a public registry. If services offered by the business organization evolve, the service source

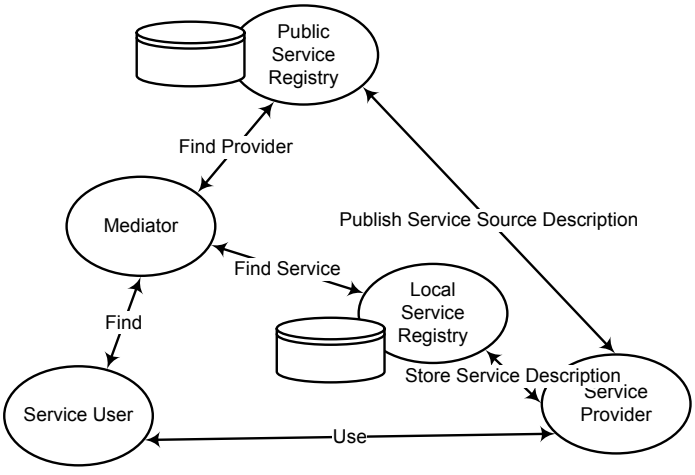


Fig. 6. Proposed service discovery approach

description is re-generated or incrementally updated, depending on the nature of the change. The updated service source description can be pushed to the service broker, in similar to the original publication, where it is used to replace the old service source description. Alternatively, the service broker periodically asks the service provider for updates.

2. A business organisation looking for partners that can offer certain services can contact the service broker and find service source descriptions based on keyword matching. Although keyword matching can already be reasonable efficient [22], we can further improve the recall and precision of service discovery by exploiting the RDF-based semantics of service source descriptions.
3. If a service source description fulfills the search criteria, the service provider is contacted via the endpoint that is part of the service source description. Then the local registry of the service provider is used to find available services. Both step (2) and step (3) may be performed via an intermediary, making the two-step service discovery transparent to the requesting business organization. For example, the service broker may take the intermediary role. Possibly, the requested services are not available as single services or are not available from a single service provider. In that case, the services have to be offered as a bundle or must be composed. Again, an intermediary may automate or support this process [23].
4. Once the (composed) services are found, the requesting business organization can start using them, effectively entering collaborations with one or more partner business organizations that are involved in the offering of these services.

Comparing this with the enterprise collaboration through public UDDI registries, we observe the following benefits:

- The service source description is based on extracting keywords from WSDL type definitions, and represents these keywords and their relationships with RDF. In this way, the semantic properties of keywords can be captured [21,20]. This addresses the limitation (a) mentioned above.
- Since the extraction is automatic, the burden for service providers to update descriptions is dramatically lowered. Moreover, if the service broker is able to poll for updates, the problem of 'disappearing' business organizations and 'ghost' services can be tackled. If a business organization no longer supports its previously published services, e.g. because it no longer exists, a poll for updates by the service broker gets no reaction and the service broker can decide to remove the service source description from its registry. This addresses the limitation (b) mentioned above.
- The two-step service discovery approach has the advantage that it first determines the services providers that offer potentially relevant services, and then limits the search for services to those of the selected service providers. Although we still have to confirm this with experiments, we believe that this approach has a better scalability than one-level semantic search. Furthermore, by favoring services from the same or a few providers, it is more likely that these services are defined and implemented in a consistent way, making search and composition easier and more efficient [24].
- In order to address limitation (c) mentioned above, the local registry of a service provider may be enhanced in two ways. First, the service provider may monitor who wants to access the local registry, and expose information on its services depending on some trust classification scheme (e.g., based on previous collaborations). Secondly, the service provider may provide additional information through the local registry, which facilitates the (non-) selection of services. For example, non-functional properties based on resource availability or historical data may be published, including information on trust, security or privacy aspects.
- The implications for existing standards, most notably UDDI, is minimal. Most of the interactions described above can be supported with UDDI as is.

6 Related Work

The problem of guaranteeing correctness of the published service descriptions and reducing the effort required for providing such guarantees is starting to attract attention from the research communities. In this section several approaches are briefly presented and compared to our approach. The characteristics of interest (and which are thus subject of comparison) are: effort to keep service descriptions up to date, deal with state-dependent information, build on existing infrastructure (e.g., re-incorporate use of existing WSDLs, and not replace them), complexity of finding services.

The work presented in [13] is in may respects similar to our approach. The focus of the work in that paper is to ensure that the published service description indeed represents the offered service. In order to support this, an estimation step followed by a single execution step is proposed. In the estimation step, additional information compared to the information that is available in the service description is gathered from the service provider whereas in the execution step, the actual invocation of the service is performed. The approach proposed by the authors is similar to our approach except their focus is on semantic services and do not support non-semantic services.

A preference-based selection of highly configurable web services is presented in [25]. It focuses on defining algorithms required for finding optimal service configurations while selecting the services. Unlike our approach, it neither considers minimisation of the effort required to update the service registry when service descriptions are changed nor maximising the correctness of the published service descriptions. To address the problem due to changes in service descriptions, a RSS-based mechanism to announce such changes is proposed in [12]. The RSS-based approach is service provider dependent because the changing information should come from the provider.

Treating services from an economical point of view, [26] defines what is referred to as the universal service description language. The proposed language is defined to describe both the IT and non-IT services. Provisions for defining dynamic information are however poorly defined in that language. The approach presented in [27] defines a mechanism for identifying and reducing irrelevant information in service composition and execution. It targets increased efficiency and correctness of the composition and execution. Consequently, it works only after the services are discovered and does not eliminate the possibility of discovering services with incorrect information, which is the focus of our approach.

In contrast to many other traditional approaches, [28] proposes a LinkedData based approach for integrating data providing services. The approach in that paper is suitable for sharing data which might change over time. In comparison to our approach, it requires the service providers to describe the offered services using LinkedData principles and does not support sharing of already existing service descriptions which are described using WSDL. We propose mechanisms to allow usage of WSDL while still dealing with changing and state-dependent data.

A crawl based approach for collecting, annotating and classifying public Web services has been proposed in [15]. It entails an attempt to increase the role of service registry in service-oriented architecture by providing correct information. Publicly available web service descriptions are crawled, annotation information is gathered, Web services are annotated and classified based on this information. Through such classification, the authors aim at supporting better discovery of services. In this direction, the work presented in [22] aims at enabling rich discovery of Web services by projecting weak semantics from structural specifications. Similarly, a clustering scheme is proposed in [29] to facilitate discovery of service described in WSDL. In comparison to our approach, the approaches presented

in [22,29] are however unable to deal with the sensitive and state-dependent information problem.

A combination of document classification and ontology alignment schemes is proposed in [21] to semantically enrich Web services. Though this scheme helps in efficiently discovering required services, it does not tackle the problem of outdated service descriptions. The work presented in [23] specifies mechanisms for runtime discovery, selection and composition of semantic services. The proposed approach supports semantic descriptions of the services but lacks support for dealing with outdated service descriptions. An approach presented in [30] uses SPARQL query pattern to describe user goals and service descriptions with an aim to improve service discovery result. It describes service functionality in terms of pre- and post-conditions. The pre- and post-condition is important for finding services only if the service descriptions are static.

We summarise the mentioned approaches with respect to the characteristics stated above in Table 1.

Table 1. Related Works

	update effort	state-dependent info.	Use of WSDL	discovery complexity
[13]	-	Yes	No	Low
[25]	No	No	Yes	High
[12]	Medium	No	Yes	Medium
[26]	High	No	No	High
[27]	High	No	Yes	-
[28]	-	Yes	No	Medium
[15]	High	No	Yes	Medium
[22]	Low	No	Yes	Medium
[29]	Medium	No	Yes	High
[21]	High	No	No	Medium
[23]	-	No	No	Medium
[30]	-	No	No	Medium

7 Conclusions

Our research demonstrates how service source descriptions allows us to reduce maintainability cost at the service providers' side while still providing the relevant information required for service discovery. This type of approach has the ability to guarantee better service results due to the separation of abstract descriptions and detailed descriptions. Current approaches either do not provide adequate support for publishing accurate information of the offered services or the offered solutions are too restrictive in terms of cost and time required for maintaining the published descriptions. This is mainly because the service descriptions are valid only at the time they are created and subject to frequent change caused by in market developments. In the proposed work, we generate

service type information from the given WSDL information. The service type information is currently treated separately per services. It is possible that some of the concepts used in the service type information is used across multiple services. It would therefore be appropriate to merge these items of type information. We aim at looking at this direction in our future research.

Acknowledgments. This material is based upon works jointly supported by the IOP GenCom U-Care project (http://ucare.ewi.utwente.nl) sponsored by the Dutch Ministry of Economic Affairs under contract IGC0816 and by the DySCoTec project sponsored by the Centre for Telematics and Information Technology (CTIT), University of Twente, The Netherlands.

References

1. Erl, T.: Service-Oriented Architecture Concepts, Technology, and Design. Prentice Hall Professional Technical Reference (2005)
2. van Sinderen, M., Almeida, J.P.A.: Empowering Enterprises through Next-Generation Enterprise Computing. Enterprise Information Systems 5(1), 1–8 (2011)
3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Springer, Heidelberg (2004)
4. van Sinderen, M.: From Service-Oriented Architecture to Service-Oriented Enterprise. In: Proc. of the Third International Workshop on Enterprise Systems, pp. 3–16 (2009)
5. Roman, D., Lausen, H., Keller, U. (eds.): Web Service Modeling Ontology (WSMO). WSMO Working Group (October 2006)
6. Martin, D. (ed.): OWL-S: Semantic Markup for Web Services. W3C Member Submission (November 2004)
7. Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema (August 2007)
8. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool (2011)
9. Chinnici, R., Moreau, J.J., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 (June 2007)
10. Klyne, G., Carroll, J.J. (eds.): Resource Description Framework: Concepts and Abstract Syntax. W3C Recommendation (February 2004)
11. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Candidate Recommendation (June 2007)
12. Treiber, M., Dustdar, S.: Active Web Service Registries. IEEE Internet Computing 11(5), 66–71 (2007)
13. Küster, U., König-Ries, B.: Supporting Dynamics in Service Descriptions - The Key to Automatic Service Usage. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 220–232. Springer, Heidelberg (2007)
14. Michlmayr, A., Rosenberg, F., Platzer, C., Treiber, M., Dustdar, S.: Towards Recovering the Broken SOA Triangle: A Software Engineering Perspective. In: Proc. of the 2nd International Workshop on Service Oriented Software Engineering, pp. 22–28 (2007)

15. AbuJarour, M., Naumann, F., Craculeac, M.: Collecting, Annotating, and Classifying Public Web Services. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 256–272. Springer, Heidelberg (2010)
16. Sabou, M., Pan, J.: Towards semantically enhanced Web service repositories. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 142–150 (2007)
17. Al-Masri, E., Mahmoud, Q.H.: Investigating Web Services on the World Wide Web. In: Proc. of the World Wide Web Conference, pp. 759–804 (2008)
18. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: Proc. of the 30th VLDB Conference, pp. 372–383 (2004)
19. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (February 2004)
20. Thuy, P.T.T., Lee, Y.K., Lee, S., Jeong, B.S.: Exploiting XML Schema for Interpreting XML Documents as RDF. In: Proc. of the 2008 IEEE International Conference on Services Computing, pp. 555–558 (2008)
21. Crasso, M., Zunino, A., Campo, M.: Combining Document Classification and Ontology Alignment for Semantically Enriching Web Services. New Generation Computing 28, 371–403 (2010)
22. Obrst, L., McCandless, D., Bankston, M.: Enabling Rich Discovery of Web Services by Projecting Weak Semantics from Structural Specifications. In: Proc. of Semantic Technology for Intelligence, Defense, and Security (2010)
23. da Silva, E.G., Pires, L.F., van Sinderen, M.: Towards runtime discovery, selection and composition of semantic services. Computer Communications 34(2), 159–168 (2011)
24. Forestiero, A., Mastroianni, C., Papuzzo, G., Spezzano, G.: A Proximity-Based Self-Organizing Framework for Service Composition and Discovery. In: Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 428–437 (2010)
25. Lamparter, S., Ankolekar, A., Grimm, S.: Preference-based Selection of Highly Configurable Web Services. In: Proc. of the 16th International Conference on World Wide Web, pp. 1013–1022 (2007)
26. Cardoso, J., Winkler, M., Voigt, K.: A Service Description Language for the Internet of Services. In: Proc. of the International Symposium on Services Science (2009)
27. Truong, H.-L., Comerio, M., Maurino, A., Dustdar, S., De Paoli, F., Panziera, L.: On Identifying and Reducing Irrelevant Information in Service Composition and Execution. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 52–66. Springer, Heidelberg (2010)
28. Speiser, S., Harth, A.: Integrating Linked Data and Services with Linked Data Services. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 170–184. Springer, Heidelberg (2011)
29. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering WSDL Documents to Bootstrap the Discovery of Web Services. In: Proc. of the 2010 IEEE International Conference on Web Services, pp. 147–154 (2010)
30. Iqbal, K., Sbodio, M.L., Peristeras, V., Giuliani, G.: Semantic Service Discovery using SAWSDL and SPARQL. In: Proc. of the Fourth International Conference on Semantics, Knowledge and Grid, pp. 205–212 (2008)

Revisiting Goal-Oriented Requirements Engineering with a Regulation View

Gil Regev[1,2] and Alain Wegmann[1]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL),
School of Computer and Communication Sciences, CH-1015 Lausanne, Switzerland
{Gil.Regev,Alain.Wegmann}@epfl.ch
[2] Itecor, Av. Paul Cérésole 24, cp 568, CH-1800 Vevey 1, Switzerland
g.regev@itecor.com

Abstract. Goal-Oriented Requirements Engineering (GORE) is considered to be one of the main achievements that the Requirements Engineering field has produced since its inception. Several GORE methods were designed in the last twenty years in both research and industry. In analyzing individual and organizational behavior, goals appear as a natural element. There are other organizational models that may better explain human behavior, albeit at the expense of more complex models. We present one such alternative model that explains individual and organizational survival through continuous regulation. We give our point of view of the changes needed in GORE methods in order to support this alternative view through the use of maintenance goals and beliefs. We illustrate our discussion with the real example of a family practitioner association that needed a new information system.

Keywords: Goals, Requirements, Regulation, Survival, Appreciative System, Norms, Beliefs.

1 Introduction

Since the advent of requirements engineering (RE) research as an academic discipline, its flagship methods have been Goal-Oriented Requirements Engineering (GORE). GORE methods have been lauded as one of the main achievements of the RE community [31, 19]. GORE is still a very active field of research with dedicated workshops and conference tracks (e.g. the i* International workshop series, International Workshop on Requirements, Intentions and Goals in Conceptual Modeling). Some prominent GORE methods were also developed by RE practitioners, e.g. Goal-Oriented Use Cases [7] and Essential Use Cases [8].

The emergence of GORE methods coincides with a less software centric view of requirements. RE evolved out of software specification methods by capturing more and more of the environment of the envisioned software system [21], i.e. the composite system [31]. GORE methods are based on the understanding that goals justify and explain requirements that are assigned to agents in the composite system (software system and its environment), and that they help detect and resolve conflicts

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 56–69, 2012.

among different stakeholder viewpoints [9]. In GORE methods and subsequently in RE, it is assumed that the behavior of the stakeholders in the environment of the software system is predominantly goal-oriented, see for example [16, 21]. Very few RE researchers have challenged this assumption and there is little debate concerning the epistemological roots of GORE methods.

We take Gause and Weinberg's view that RE is about discovering what is desired [12]. In this paper, we show that what people desire has more to do with the way they regulate their affairs than with the goals they pursue. We base our proposal mostly on Vickers' concept of Appreciative System [26, 32, 33]. Vickers believed that goal achievement was not the ultimate explanation for individual and organizational behavior. He argued that [33]:

> to explain a doing solely by reference to its intended results would seem to raise insoluble pseudo-conflicts between ends and means, rules and purposes, while it leaves the ongoing activities of norm-holding with their inherent, ongoing satisfactions hanging in the air as a psychological anomaly called action done for its own sake.

We show that goals are only the visible part of the way individuals and organizations regulate their norms in order to survive.

GORE methods have most of the necessary constructs to model this behavior, e.g. maintenance goals and beliefs. We advocate a more systematic and widespread use of these concepts for better understanding regulation and for improving GORE methods.

For Nuseibeh and Easterbrook there is a type of RE for each end product [21]. They give the following examples: RE for information systems, RE for embedded control systems and RE for generic services e.g. networking and operating systems. In this classification our discussion applies mostly to RE for information systems. In this type of RE, the composite system is the organization that the envisioned computer-based system serves. Nuseibeh and Easterbrook further state that the context of most RE and software activities is in this field of information systems development. Hence our discussion is applicable to many RE projects.

We explain our proposal with the data we gathered during a recent project we were involved in. The project began with the goal of replacing a spreadsheet (for tracking interns specializing as family doctors) with an on-line application, but it turned out to be about the maintenance of the community of family doctors.

We begin by presenting the family doctor project (Section 2). We proceed by reviewing the relevant GORE research (Section 3). We then introduce the regulation organizational model and show that it can be used as the underlying mechanism, of which goal-oriented behavior is but the visible part (Section 4). We propose improvements to GORE methods based on the regulation-oriented view (Section 5). We review the related work in Section 6.

2 The Family Practitioner Example

The project we were involved in was initiated when we were approached by a Family Practitioner (FP) we call Mark. Mark is the president of an association of family

practitioners that we will call the FPA (for Family Practitioner Association). All discussions of the FPA in this paper represent our own partial understanding of this case.

To earn a license to practice, freshly graduated medical doctors need to do six years of internships in domains related to their specialty. For the specialty of family practitioner, however, there is no specifically related internship. For example, a surgery intern might do 3 internships in surgery and 3 in other domains. An FP intern may do internships in 6 different domains. Selecting the right set of internships for FP interns is difficult. FPA members advise FP interns on the internships that will qualify them for a license in family practice.

The FPA has a list of interns who are interested in becoming FPs. To record information about these FP interns and available internships, the FPA uses a non-centralized, paper-based system.

The FPA secretary records on a spreadsheet the available internships and tracks the internships of each intern. The secretary connects the FP interns with FPA members who act as advisors. The advisors meet with the interns and explain which internships should be completed in order to qualify as an FP. The interns walk away from these meetings with a hand-written paper containing their personalized internship plan. During their internship period, the interns will meet with several advisors in different regions.

Mark explained that the previous president systematically compiled internship positions and reserved certain positions for FP interns. This represented a lot of work for his secretary and did not guarantee internships to all FP interns. Because of these difficulties, Mark decided to stop providing this service and focus on the advice given to FP interns concerning their internships. His goal is to improve the consistency of the recommendations given to the interns by the various advisors they meet during their internship period. He is therefore interested in a web-based system that FPA members and FP interns can use concurrently.

3 An Overview of GORE Methods

GORE methods use the concept of goal as the main construct for defining requirements. The use of the concept of goal in RE has emerged from research in software engineering where requirements began to supplant specifications as a way to describe the envisioned system. Researchers needed to analyze the environment of the envisioned system, because the system by itself could not guarantee the results expected by its stakeholders. The first papers linking goals and requirements date back to the beginning of the RE discipline, e.g. [9, 11, 27]. The seminal GORE paper was written by Dardenne et al. [10]. It introduced vocabulary inherited from Artificial Intelligence [20], e.g., goals, agents, roles, objectives, constraints, obstacles, and and/or graphs. Goals have been found to enable requirements engineers to: provide a higher-level and more stable view than requirements that implement them; to generate alternative solutions and select among them; and to analyze the requirements completeness and traceability [24].

Many GORE methods have been defined over the years. The most prominent are: KAOS, GBRAM [1], i* [37], GRL [13], TROPOS [18], ESPRIT CREWS [28], MAP [29] and Goal-oriented Use Cases [7]. Numerous goal types have been defined in these methods. The following is a partial list of goal types: achievement, maintenance, softgoal, feedback and satisfaction goals.

Goals very quickly became a central concept in RE. Zave and Jackson, for example, describe RE as, "Requirements Engineering is about the satisfaction of goals" [38]. Zave [39] defines RE as "the branch of software engineering concerned with the real-world goals for functions of and constraints on software systems." The call for papers of the Requirements Engineering conference series also strongly links requirements and goals, e.g. Requirements Engineering Conference 2004 [22]:

> Requirements Engineering (RE) is the branch of systems engineering concerned with the goals, desired properties and constraints of complex systems, ranging from embedded software systems and software-based products to large enterprise and socio-technical systems that involve software systems, organizations and people.

3.1 The Assumptions behind GORE

The focus on goals is understandable because it relates with the goal-seeking organizational model prevalent in the neighboring discipline of Information System [5]. Many GORE methods take for granted this goal-seeking model. Enterprise Modeling, for example, is said to include the goals of its agents [16] or members [21].

The base assumption underlying most GORE methods is that high-level enterprise goals can be gradually refined into requirements that can be assigned to the envisioned system [10, 31]. This refinement is most often done with the help of and/or goal graphs inherited from [20].

Although many types of goals have been defined in GORE methods, the most popular goal type is the achievement goals; a goal that is to be achieved once and for all. The next most popular goal type is the softgoal. Both KAOS and GBRAM introduced the concept of maintenance goal, a goal that "is satisfied as long as its target condition remains true" (Anton and Potts, 1998). Maintenance goals have not received much attention and remain largely unused. This is particularly unfortunate because maintenance goals have been identified as "high-level goals with which achievement goals should comply" [2]. However, no explanation has been given to this high-level status of maintenance goals or to their relation with achievement goals.

3.2 The Lack of Theoretical Grounding

Several RE researchers have reported problems with the application of GORE methods, Goal discovery and goal refinement do not seem to be straightforward tasks. In particular, [1, 28, 38] note that "goals by themselves do not make a good starting point for requirements engineering." and that "Almost every goal is a subgoal with some higher purpose." They show that goal abstraction may lead to unrealistic or unwanted alternatives.

The proposed remedies are to bound goal abstraction and refinement with the subject matter of the organization [38], to use interview transcripts and organizational documents for goal discovery [1] and to use scenario and goal reasoning together so that they inform one another [28].

We believe that the problems identified by RE researchers are a sign of a deeper issue, necessitating a broader view of GORE and its relationships to individual and organizational behavior In their suggested roadmap for RE, Nuseibeh and Easterbrook [21] provide a very broad perspective on RE research. They explain that "RE is a multi-disciplinary, human-centered process"; that it uses cognitive and social sciences as theoretical grounding; and that [21]:

> RE must concern itself with an understanding of beliefs of stakeholders (epistemology), the question of what is observable in the world (phenomenology), and the question of what can be agreed on as objectively true (ontology).

There has been very little theoretical grounding of GORE, and the three concerns identified by Nuseibeh and Easterbrook are missing in most GORE research. Questions such as the following receive little attention in this stream of research: what is the nature of goal-oriented behavior. Are goals the ultimate explanation of human and organizational behavior (as often assumed in RE)? Is there an explanation to the source of goals themselves (apart from scenarios, interviews and transcripts)? What is the relationship between goals, beliefs, observations and agreements?

3.3 Goal Refinement and Abstraction

Most GORE methods place goals in some hierarchy. Goal refinement is used to identify lower-level goals by asking how a given goal is achieved. Goal abstraction is used to identify higher-level goals by asking why a given goal needs to be achieved.

Although both why and how questions are encouraged by GORE methods, the how is much more prevalent in GORE publications. Most often, a so-called high-level goal is postulated to be strategic for the organization under analysis and is refined into subgoals. For example, van Lamsweerde gives the following examples for "high-level, strategic concern": "serve more passengers" for a train transportation system" and "provide ubiquitous cash service for an ATM network system." [31]. It is not clear why these should be considered as high-level, strategic goals and how they can be satisfied. If the train transportation system serves a few more passengers, is this goal achieved? How many passengers are considered to be enough? Is there a limit to the number of passengers? Should the transportation system be designed to serve an infinite number of passengers? What will the system do next once this goal is achieved? What if this goal is never achieved? What would the ATM network system do once the ubiquitous cash service is provided? What are the criteria of achievement for a ubiquitous cash service? Similarly, i* highest-level diagram is called a "strategic dependency model" but nowhere is it explained what makes the goals expressed in this diagram strategic.

Applying the same pattern to the FPA example, we would begin with the goal of maintaining advice consistency. We would then refine this goal into achievement subgoals: all advisors to an FP intern use FP intern record and Each FP intern uses own FP intern record. These goals in turn will be supported by system level goals such as, FP intern record available to FP intern advisors and FP intern record available to FP intern. Goal abstraction is even more difficult. To paraphrase Zave and Jackson [38]: what is the goal of maintaining the advice consistency? Is it to keep the FP interns comfortable? Is it to improve family practice? Shouldn't the goal of the FPA be to improve medical practice in general? To satisfy this latter goal, should the FPA consider to become the medical practice association (MPA)?

As can be seen in this example, no rationale is given to the goal refinement and abstraction. Why would the use of the same record guarantee the consistency of the advice? Should the FPA change to the MPA? In agent programming, this rationale is often provided with the use of beliefs. In KAOS, GBRAM, ESPRIT CREWS and MAP, goals are not embodied in an agent and they do not have a concept corresponding to a belief. In i* goals are embodied within agents and the concept of belief is defined, but is very seldom used.

4 Survival and Regulation as the Source of Goals

In this section we describe an epistemological view that explains the source of goals as emerging from the regulation mechanisms that are at the base of the survival of an organization in a changing environment. We present Vickers' appreciative system as a possible base for thinking about GORE methods.

4.1 Survival as the Maintenance of Norms

As we have seen, GORE methods seek to define the highest-level goals that are adequate for defining requirements for an envisioned system. Despite their important advancement, this is one aspect that the mainstream GORE methods (e.g. i*, KAOS) have not defined yet. So-called high-level goals are often described as strategic goals in GORE papers. It is therefore important to understand what a strategic goal is. In the most general case, the highest-level goal that can be ascribed to an organization is to survive. We then have to define what we mean by survival. We have shown elsewhere, e.g. [24, 25], that survival can be understood in terms of the maintenance of an identity for a given observer. Maintaining this identity requires the maintenance of stable states within the boundaries defined by the observer [24]. These stable states are often called norms. Observers use the norms maintained by an organization to identify it as a separate entity from other organizations. Survival is therefore not an absolute measurement. It depends as much on the observer as on the observed. This means that some observers will select some norms as defining the organization and others will select other norms.

It is crucial that norms remain stable for an observer to recognize the organization over time. If a given feature of the organization changes its state beyond some threshold, the observer will not be able to identify it as the same feature as before the

change and will therefore fail to recognize the feature and the organization. When two organizations merge, their norms become indistinguishable for observers.

The FPA, for example, has norms that separate it from other similar associations, such as, a unique name, mission, statutes, a logo and offices. It also has members, a president and a secretary. The individuals who fill these positions change over time, but relatively slowly. The FPA remains more or less the same (most of its norms remain the same) even when some individuals leave the association and some others join it.

4.2 Regulation as a Source of Goals

In a changing environment, organizational norms remain stable due to incessant effort to counter change [35, 36]. This is often called regulation. Regulation is therefore a powerful source of action designed to bring a state closer to the norm. Norms also place constraints on possible actions by defining what is permissible and what is not.

The FPA attempts to maintain some norms that it deems important for the survival of the family practitioner practice, e.g. the number of family practitioners in its region, their level of expertise, their sense of community, their recognition among other MDs, among politicians and the general population. To maintain these norms, the FPA may take many actions, such as: presenting the FP practice to students; promote FP courses in the curriculum; provide advice to FP interns; Create FP communities in different regions.

4.3 Organizations as Open Systems

An organization needs energy in order to track the stability of norms, to spawn regulative actions when needed and to change norms when needed. In a closed system, energy is finite. When it is spent, the organization will lose its ability to maintain its norms and will disintegrate. The organization needs to have relationships with individuals and organizations in its environment in order to exchange energy with them and therefore have the means to maintain its norms. This is a consequence of the open system model of organizations [24]. These relationships must themselves be maintained within very specific threshold associated with a norm for the organization to be able to leverage them for maintaining its norms.

The FPA needs a continuous flow of FP interns either to replace retiring FPA members or to increase the number of FPs. The FPA also needs a steady stream of FP interns that it must convert into FPA members. FP interns must fit very stringent quality levels, which means that they need to be within the threshold range of the family practitioner norm. Internships must fit the tolerance range for qualifying interns for their FP license.

To promote the FP practice, the FPA has to be representative of this practice. This means that it has to insure that its members represent the majority of the FPs. The FPA needs as many members as possible, however, the members must be FPs or else the FPA will lose its FP identity.

4.4 Changing Norms to Fit the Environment

The environment around the organizations continually changes and this induces changes to the organization's norms. Hence, to maintain its norms relatively stable does not mean that the norms do not change at all. Organizations that survive over the long-run make changes to their norms to fit the environment, but in a very controlled way. This means that changes must be maintained within the boundaries defined by its stakeholders in order for the organization to maintain its identity for these stakeholders.

The FPA remains the association for family practitioners. It does not become the association for surgeons or other specialty. However, it does change some of its norms over time. The new president is not interested in some of his predecessor's ways of operating, e.g., tracking and reserving internships, but is more concerned with the consistency of the advice given to FP interns.

4.5 Vickers' Appreciative System and Goal Concepts

A feedback regulator, for example, a thermostat or an automatic pilot, maintains a given state stable, the temperature or the course, by sensing the current state comparing it with the given state, and applying some action if the difference is above the tolerance level. Vickers [32, 33] proposes the concept of the appreciative system, by extending this model of a feedback regulation to human and organizational regulation. Vickers's appreciative system has three components [6, 26, 32, 33]: reality judgments, value judgments and action judgments. With reality judgments, some aspect of reality is singled out to be the study of attention. In value judgments, this reality judgment is matched to a category within which it is then compared to the norm (what ought to be). In Action judgment, some action might be taken to bring the reality judgment closer to the norm. The three judgments function as a complete system so that change to one of them requires change to the others. Hence, the way we view and judge the world affects our actions and our actions affect our view and judgments [32, 33].

Action judgments are exercised when the reality judgment is considered to be outside the threshold associated with a relevant norm within the category in which the reality judgment was placed. This comparison brings about a host of debates about the course of action taken by the organization. For example, is the FPA promoting the FP practice enough, too much or too little? How important is the consistency of the advice? Should the FPA track the available internships? Is it useful to have a central registry of FP interns?

In a simple automaton the information to be sensed, the norm, the threshold to compare with and the kind of actions to be taken are all given by its designer. In an appreciative system, they are all subject to continuous change; nothing is set once and for all by a designer. Hence, an appreciative system creates its own dynamics, which an automaton does not.

Even though the appreciative system creates its own judgments, it is nevertheless a rather stable construct, i.e. it creates its own norms. Vickers calls these norms readiness. An individual or an organization has a readiness (a tendency) to see things

in certain ways, to value them in certain ways and to act in certain ways. All these are rather stable in time. It is often the role of the analyst to try and shake-up these readinesses.

Vickers [33] insists on the fact that when a current state of affairs is outside of the tolerance level, no (externally visible) action is necessarily taken. It may very well be that either the reality judgments, the value judgments or both may change. Changing these judgments leads to the acceptance of the state of affairs rather than an attempt to change it. Likewise, action judgments can also change when the organization changes its behavior when dealing with similar situations. These changes to the appreciative system result in an adaptation of the organization to its environment. Changes to action judgments are much more visible because they result in changes to visible behavior. Changes to reality and value judgments are often much less visible,

This adaptation is visible in the FPA example. Recall that Mark revised his view of the FP intern problem during the project. He gave much less importance to the advice consistency problem and much more importance to the maintenance of a sense of community between FP interns and FPA members. In the appreciative system model, this means that he changed his value judgments. The reality judgment saying that the advices were inconsistent didn't change, but this inconsistency is now given much less value. However, maintaining the community is given much more value.

5 Improving GORE Methods

Based on the regulation view we have proposed in the previous section, we propose a number of improvements to GORE methods.

5.1 Maintenance Is Higher-Level Than Achievement

As we have shown, the concept of maintenance goal is an approximation of the concept of norms, and norms relate directly to the survival of the organization for different stakeholders. Achievement goals model actions that are taken most often to maintain a related norm (modeled by a maintenance goal). This clarifies the status of maintenance goals as being higher-level than related achievement goals.

Mark's description of the system he was looking for made us focus on the following maintenance goal: *Maintain the consistency of the advice given to FP interns*. The lower level achievement goals can be: *All advisors to an FP intern use FP intern record* and *Each FP intern uses own FP intern record*.

5.2 Maintenance Goals and Tolerance Levels

The threshold associated with a norm defines the tolerance level of stakeholders. What they define as a problem or what they can live with. Maintenance goals are an approximation of norms because they lack the concept of tolerance levels. When does it become clear that a certain state does not satisfy the related maintenance goal? Maintenance goals should be augmented with tolerance levels so that they are better suited for modeling norms.

In the FPA example, this means that we need to augment the maintenance goal of maintaining the advice consistency with tolerances about the consistency. When are two advices considered to be inconsistent? How would different advisors consider the consistency of their advices?

5.3 High-Level Goals and Alternatives

Understanding that high-level goals model norms can help requirements engineering to seek the norms that different stakeholders consider as identifying the organization. This can lead to a more widespread use of maintenance goals, as the highest-level goals that model an organization survival are therefore necessary.

Identifying the norms that are considered essential for survival helps us solve the unacceptable goal alternatives identified by Zave and Jackson [38]. Identifying these strategic norms places the appropriate bounds on what alternatives are acceptable and not acceptable.

For the FPA, instead of simply taking the expressed goals of maintaining the consistency among advisors, it is interesting to understand what norms are essential for the survival of the FPA for different stakeholders. As we have seen, for Mark, the most important norm is the maintenance of a community of FPs. A higher-level goal to maintaining the advice consistency is therefore to maintain a sense of community between FP interns and their advisors. This goal can be refined into the maintenance goal of maintaining regular dinners between FP interns and advisors. This goal refinement is connected by beliefs as described below.

5.4 The Appreciative System and GORE Concepts

Linking the appreciative system and GORE aspects, we suggest to refine the framework we proposed in [24] by considering maintenance goals as a reflection of norms, beliefs can be as a reflection of reality and value judgments, and achievement goals as a reflection of action judgments.

From this point of view, GORE methods have mainly concentrated on the third stage, action judgments (achievement goals), and neglected the other two, norms (maintenance goals), and reality and value judgments (beliefs). This is easily understandable, if we consider that action judgments are much more visible than reality and value judgments.

Considering beliefs as reflecting reality and value judgments, means that beliefs should become a major concept in GORE methods. Because the three judgments of Vickers' appreciative system are interlocked, goals cannot be changed without changing the beliefs that justify them. This means as much as possible, stakeholders goals must be justified by corresponding beliefs. Alternative goal refinements (i.e., the or part of the and/or graph) must be justified by different sets of beliefs. Goal abstraction, likewise, leads to beliefs rather than directly to higher-level goals.

For the FPA, asking Mark why he wants a centralized system elicited the answer that FP interns were receiving inconsistent advice from the advisors they met. Mark believed that this inconsistency might distance FP interns from the FP practice and

from the FPA. A relevant higher-level goal is to keep the FP interns within the FPA community. A subgoal is to maintain a sense of community between FP interns and their advisors. The underlying belief is that a sense of community keeps people together). Mark has now the subgoal of putting in place community building mechanisms, such as regular dinners bringing together FP interns and their advisors. The underlying belief is that these regular dinners will strengthen the sense of community between the FP interns and their advisors and will therefore reduce the risk of FP interns leaving the FP practice because of the inconsistency of the advice they receive.

The following example shows other goals that were influenced by beliefs during this project. To stop tracking internship information, Mark and his secretary must believe that it is not necessary anymore. An interview with the secretary showed that she believed that tracking available internships was still necessary, even though Mark believed that it was not.

6 Related Work

Several conceptual studies of GORE methods have been published over the years, e.g. [14, 15]. These studies assume a viewpoint from within the RE research paradigm. They do not ground their research in an external body of knowledge, which limits their explanatory power of goals. In recent years, i* has become the main GORE method. Much research has applied i* in different contexts. Recently, a well received study of the i* graphical notation was published [17], noting that major improvements are needed in order to make i* user friendly. The study, however, was limited to the graphical level and did not investigate the epistemological or ontological aspects of i*.

Our work is similar in nature to Checkland and Holwell's conceptual cleansing [5] of the field of information systems. Checkland [4] has worked extensively to popularize Vickers's work with Soft System Methodology (SSM). Ours is a very short description of Vickers's appreciative system. More elaborate descriptions are available in Vickers's writings [32, 33], and in [4, 6, 26]. We proposed an explanation of goals based on General Systems Thinking and Vickers's work in [24].

Sutcliffe and Maiden [30] proposed a notable kind of goal in a paper that seems to have received little attention by GORE researchers. They proposed 6 classes of goals. One of the classes is called "feedback goals." They describe these goals as maintaining a desired state with a related tolerance range, spawning corrective actions when the state is considered to be outside the tolerance range [30]. This class of goals seems to have gone unnoticed by subsequent GORE research. We have ourselves added feedback into GORE research about 10 years later [23, 24] without noticing the significance of Sutcliffe and Maiden's feedback goal class at the time.

7 Conclusions

RE is about understanding peoples' desires and maybe designing some automated system to help them to obtain or maintain them. RE must therefore find the balance

between what is desired and what is feasible. To understand what is desired, it is necessary above all to understand individual and organizational behavior. GORE methods have made major contributions to the practice of RE but have modeled this behavior in too simplified terms, mostly as goals to be achieved. In this paper, we have shown that goals can be seen as the visible part of regulation. Regulation models the way individuals and organizations attempt to survive in a changing environment.

Regulation results in the establishment of norms, stable states that define the identity and therefore the survival for a given observer. A long lasting organization manages its internal and external relationships in a way that controls the changes to these norms but still allows them to change when needed. Looking at regulation rather than goals shifts the attention to the way people manage stability and change by managing relationships.

GORE methods already have useful concepts needed for the study of regulation, e.g. maintenance goals, and beliefs. These concepts need to be used much more than they have been until now and they need to be extended with more regulation concepts. We have shown some of these concepts, e.g., the threshold that defines the tolerance of deviations from a norm, reality judgments and value judgments. Obviously, much more research can be done on modeling reality and value judgments as beliefs. Many regulation concepts are described in [35 and 36]. It will be useful to add them to GORE methods.

In this paper we limited our epistemological discussion to Vickers' appreciative system. There are many other conceptual frameworks that can be used, for example, Weick [34] and the Viable System Model (VSM) [3].

References

1. Anton, A.I.: Goal-based requirements analysis. In: Proc. ICRE 1996 Second International Conference on Requirements Engineering. IEEE (1996)
2. Anton, A.I., Potts, C.: The use of goals to surface requirements for evolving systems. In: International Conference on Software Engineering, ICSE 1998. IEEE (1998)
3. Beer, S.: The viable system model: its provenance, development, methodology and pathology. Journal of the Operational Research Society 35(1), 7–25 (1984)
4. Checkland, P.: Soft System Methodology: a 30-year retrospective. Wiley, Chichester (1999)
5. Checkland, P., Holwell, S.: Information, systems and information systems - making sense of the field. Wiley, Chichester (1998)
6. Checkland, P.: Webs of significance: the work of Geoffrey Vickers. Systems Research and Behavioral Science 22(4) (2005)
7. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Reading (2001)
8. Constantine, L.: Essential modeling: Use cases for user interfaces. ACM Interactions 2(2), 34–46 (1995)
9. Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed concept acquisition in requirements elicitation. In: Sixth International Workshop on Software Specification and Design, IWSSD 1991. ACM (1991)
10. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal Directed Requirements Acquisition. Science of Computer Programming 20(1-2), 3–50 (1993)

11. Dubois, E.: A Logic of Action for Supporting Goal-oriented Elaborations of Requirements. In: Fifth International Workshop on Software Specification and Design, IWSSD 1989. ACM (1989)
12. Gause, D.C., Weinberg, G.M.: Exploring Requirements: Quality BEFORE Design. Dorset House, N.Y. (1989)
13. ITU-T, Telecommunication Standardization Sector of ITU: User requirements notation (URN) – Language definition (Z.151) (2008)
14. Kavakli, E.: Goal-Oriented Requirements Engineering: A Unifying Framework. Requirements Engineering 6(4), 237–251 (2002)
15. Kavakli, E., Loucopoulos, P.: Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. Information Modeling Methods and Methodologies, 102–124 (2005)
16. Loucopoulos, P., Kavakli, E.: Enterprise Modelling and the Teleological Approach to Requirements Engineering. Intelligent and Cooperative Information Systems 4(1), 45–79 (1995)
17. Moody, D., Heymans, P., Matulevičius, R.: Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requirements Engineering 15(2), 141–175 (2010)
18. Mylopoulos, J., Kolp, M., Castro, J.: UML for Agent-Oriented Software Development: The Tropos Proposal. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 422–441. Springer, Heidelberg (2001)
19. Mylopoulos, J.: Goal-Oriented Requirements Engineering: Part II. Keynote Talk. In: 14th IEEE Requirements Engineering Conference, RE 2006, Minneapolis, MS (September 2006), http://www.ifi.uzh.ch/req/events/RE06/ConferenceProgram/RE06_slides_Mylopoulos.pdf (accessed May 2011)
20. Nilsson, N.J.: Problem Solving Methods in Artificial Intelligence. McGraw-Hill (1971)
21. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: International Conference on The Future of Software Engineering, ICSE 2000. ACM (2000)
22. RE04, 2004. Requirements Engineering International Conference (2004), http://www.re04.org/ (accessed May 2011)
23. Regev, G., Wegmann, A.: Defining Early IT System Requirements with Regulation Principles: the Lightswitch Approach. In: Proc. 12th IEEE International Requirements Engineering Conference (RE 2005), Kyoto, Japan (2004)
24. Regev, G., Wegmann, A.: Where do Goals Come From: the Underlying Principles of Goal-Oriented Requirements Engineering. In: Proc. 13th IEEE International Requirements Engineering Conference (RE 2005), Paris (2005)
25. Regev, G., Gause, D.C., Wegmann, A.: Creativity and the Age-Old Resistance to Change Problem in RE. In: Proc. 14th IEEE International Requirements Engineering Conference (RE 2006), Minneapolis, MN (2006)
26. Regev, G., Hayard, O., Wegmann, A.: Service Systems and Value Modeling from an Appreciative System Perspective. In: IESS1.1, Second International Conference on Exploring Services Sciences. Springer, Heidelberg (2011)
27. Robinson, W.N.: Integrating multiple specifications using domain goals. In: Fifth International Workshop on Software Specification and Design, IWSSD 1989. ACM (1989)
28. Rolland, C., Souveyet, C., Ben Achour, C.: Guiding goal modeling using scenarios. IEEE Trans. Software Eng. 24, 1055–1071 (1998)
29. Rolland, C., Salinesi, C.: Modeling Goals and Reasoning with Them. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements (EMSR). Springer (2005)

30. Sutcliffe, A.G., Maiden, N.A.M.: Bridging the Requirements Gap: Policies, Goals and Domains. In: 7th International Workshop on Software Specification and Design, IWSSD 1993. IEEE (1993)
31. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: 5th IEEE International Symposium on Requirements Engineering, RE 2001. IEEE (2001)
32. Vickers, S.G.: Value Systems and Social Process. Tavistock, London (1968)
33. Vickers, S.G.: Policymaking, Communication, and Social Learning. In: Adams, G.B., Forester, J., Catron, B.L. (eds.), Transaction Books, New Brunswick, NJ (1987)
34. Weick, K.E.: The Social Psychology of Organizing, 2nd edn. McGraw-Hill (1979)
35. Weinberg, G.M.: An Introduction to General Systems Thinking. Wiley & Sons, New York (1975)
36. Weinberg, G.M., Weinberg, D.: General Principles of Systems Design. Dorset House, New York (1988)
37. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: Third IEEE International Symposium on Requirements Engineering, RE 1997. IEEE (1997)
38. Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology 6(1), 1–30 (1997)
39. Zave, P.: Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys 29(4) (1997)

On the Impact of Modular Dependencies on Innovation in Organizations

Philip Huysmans

University of Antwerp,
Department of Management Information Systems,
Prinsstraat 13, Antwerp, Belgium
philip.huysmans@ua.ac.be

Abstract. In volatile and customer-driven markets, the ability to innovate is a key success factor. It has been claimed that innovations need to be implemented at a steady pace to ensure business sustainability. However, the successful implementation of innovations is only poorly understood. As a result, many organizations and governments have difficulties stimulating and managing innovation. Several authors have proposed organizational modularity as a theoretical basis to better understand and manage innovation. Their main argument is that a modular structure enables parallel evolution of different organizational modules. Consequently, innovations can be implemented without being limited by implementation aspects of other organizational modules. Similarly, an imperfect modular structure will exhibit obstacles when implementing innovations. Such a modularity analysis has been applied by various authors on different levels of the organization, such as products, processes, departments, and supporting IT systems. Often, an enterprise architecture framework is used to model these different levels. However, these frameworks do not adequately support the modeling of modularity characteristics. In this paper, we present three case studies to demonstrate (1) how modular dependencies impact enterprise architecture projects, and (2) how modeling modular dependencies can be used to complement existing modeling approaches.

Keywords: Enterprise Architecture, Business Innovation and Software Evolution, Business-IT Alignment and Traceability.

1 Introduction

Contemporary organizations are faced with rapidly changing environments. As a result, innovations need to be introduced in the organization in order to remain competitive in these markets. The introduction of innovations often impacts many different aspects of the organization. For example, Barjis and Wamba describe the impact on organizations caused by the introduction of Radio Frequency Identification (RFID) technology [1]. Besides the adaptation of the business processes through, for example, Business Process Reengineering (BPR), it can be expected that other organizational artifacts need to be adapted as well.

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 70–90, 2012.

Data management systems need to able to handle "enormous" data volumes [1], hardware infrastructure and software applications need to be purchased to handle accurate tracking, and customer acceptance needs to be handled by privacy commissions and marketing departments.

Enterprise architecture projects are frequently initiated to guide the change process needed to implement such innovations [2]. Enterprise architecture frameworks help to identify the different artifacts to which changes need to be applied. In such frameworks, models are created from different perspectives or viewpoints. Complexity from other viewpoints is abstracted away to be able to focus on the relevant aspects of a specific viewpoint. While this approach aids understandability of different aspects of the organization, it can lead to unexpected results when artifacts from different viewpoints impact each other. This can be observed in the case of RFID, where changes to processes can be hindered by the data structures on which they operate. When innovative processes are designed using a BPR approach, implementation of the new processes can be problematic because, for example, data systems are unable to support the required data types. Possibly, the redesigned processes will then need to be altered in order to be supportable. Consequently, adapting processes can be impacted by the concrete data implementation, which is not completely visible in the models which are created in a process viewpoint. While aspects of this implementation are not visible in these models, they do impact the way these models can be used, and therefore need to be considered.

In this paper, we argue that *modular dependencies* can be used to explicitly identify such impacts. This approach can be applied complementary to existing enterprise architecture modeling. We introduce relevant literature for this approach concerning modularity and enterprise architecture (EA) frameworks in Section 2. In this section, we will argue that the issues represented by modular dependencies cannot be adequately represented in current EA frameworks. We then present three case studies in Section 3 to illustrate the application of modular dependencies in organizations which are faced with innovations which require changes in several viewpoints. In the first case study, we describe how dependencies between enterprise architecture layers limit the ability to react to market changes in a public broadcasting company. In the second case study, we elaborate on a previously published case study to show the importance of eliminating such dependencies in order to design a structure which allows the implementation of innovations. In the third case study, we demonstrate in more detail how an approach using modular dependencies can be applied during an enterprise architecture project. In Section 4, we discuss the goal of resolving modular dependencies, and how this paper contributes towards reaching that goal. Finally, we present our conclusions on these cases in Section 5.

2 Research Background

In this section, we introduce a necessary background of modularity and enterprise architecture research literature.

2.1 Modularity

Organizational modularity recently receives much attention in both research and practice. Campagnolo and Camuffo provide a literature overview of 125 management studies which use the modularity concept [3]. They define modularity as "an attribute of a complex system that advocates designing structures based on minimizing interdependence between modules and maximizing interdependence within them" [3]. They argue that organizational artifacts such as products, production systems, and organizational structures can be regarded as modular structures. A characteristic of a perfect modular structure is that it enables parallel evolution of different modules. Baldwin and Clark introduced modular operators through which this evolution can occur [4]. These operators describe how modules can be (1) split into new modules; (2) substituted for other modules; (3) excluded from a system; (4) added to a system; (5) organized in a new structure (modular inversion); and (6) applied in another system (ported). However, dependencies between the modules of such a structure limit the ability of the individual modules to perform these operators. Baldwin and Clark state that "[b]ecause of these dependencies, there will be consequences and ramifications of any choice" made during the design of the artifact [4]. A design choice for a given parameter can limit or affect the possible design choices concerning other parameters. In traditional modularity approaches (e.g., product modularity), dependencies between and within modules are visualized by Design Structure Matrices (DSM). In a DSM, an artifact is described by a set of design parameters. The matrix is then filled by checking for each parameter by which other parameters it is affected and which parameters are affected by it. The result is a map of dependencies that represent the detailed structure of the artifact. An example design structure matrix is shown in Figure 1. Dependencies are represented by an "x". The intersection of identical design options is marked with a ".". Consider the design dependency which is represented by the "x" in the intersection of the column of design option A2 and the row of design option A1. This signifies that design option A2 influences design option A1: the design decision for design option A1 will be dependent on the decision taken for design option A2. This dependency does not break the modular structure of the artifact, since design options A1 and A2 both belong to the same module. Now consider the dependency of design option B1 on design option A2. Since these design options belong to different modules, it can be concluded that these modules are directly dependent on each other. Therefore, this dependency does violate the modular structure. Indirect or chained dependencies can occur as well. While design option B2 does not seem to affect any design options of module A, it does affect design option B1. As we discussed before, design option B1 does affect design option A1. Therefore, a so-called chained dependency exists between design options B2 and A2.

Originally, DSMs were used to model modular products. In later publications, a DSM was also applied to model organizational departments [5]. This indicates that similar tools can be used for analyzing modularity on various levels. In this paper, we will focus on modularity on the organizational and software levels.

		Module A		Module B	
		Option A1	Option A2	Option B1	Option B2
Module A	Option A1	.	x		
	Option A2		.	x	
Module B	Option B1			.	x
	Option B2				.

Fig. 1. An example Design Structure Matrix

Organizational modularity focuses on other artifacts than product modularity within the same organization. Research on this level has been performed by, for example, Galunic and Eisenhardt [6]. Galunic and Eisenhardt consider organizational divisions as modular organizational building blocks. These divisions, which have independent decision power, cost structures and profit responsibility, are a combination of capabilities and charters. Charters represent the task, market and customer a division is concerned with. Charters need to be able to change as markets evolve. By dynamically attributing these charters to organizational divisions, a flexible organization is created which can adapt to changing market conditions. This kind of modularity therefore enables the flexibility required on a business level. On the software level, modularity is used to achieve a flexible structure as well. For example, Parnas argued that a modular decomposition in software systems should be made to isolate the impact of changes [7]. When the impact of a change remains within a module, changes can be applied to individual modules without requiring changes in other parts of the system. Mannaert et al. proved that the impact of such changes is an essential obstacle for achieving evolvability [8,9]. While modularity is applied to both the organizational and software level by various researchers, most research project focus on a single level. However, interactions between a modular approach on the organizational and the software level remain an important issue. Given the high dependence on IT systems, it is important that a change on the organizational level (e.g., changing chapters) can be handled by the supporting IT systems as well. Otherwise, the inability to change the IT systems will restrict the ability to change organizational modules. Indeed, modularity needs to be considered as a *relative* attribute of complex systems (such as organizations), meaning that within a single artifact, different levels of modularity can exist [10]. As discussed, modularity has already been applied separately on the level of products, processes and organizational structures.

2.2 Enterprise Architecture

Enterprise Architecture (EA) frameworks provide insight to the structure of organizational goals, divisions, and supporting IT systems. By specifying separated viewpoints on organizational artifacts, an overview is provided of specialized models created by different stakeholders [11]. These models are grouped in different layers. Currently, no consensus has been reached on a specific classification of layers: different classifications are used by different researchers or frameworks. Based on a review of 126 papers, Schöenherr claims that the following layers occur most often [12]:

- Strategy layer
- Business layer
- Functional layer
- Information layer
- Application layer
- Infrastructure layer

While we do not claim that this classification of layers is the most comprehensive or correct, we will follow this classification. The analysis made in this paper will apply to other classifications as well.

Current enterprise architecture frameworks do not explicitly focus on dependencies or interactions between different architectural layers. Most frameworks use a top-down perspective. They start by defining business goals and high-level artifacts to realize these goals (e.g., the organizational structure). Based on these artifacts, lower-level artifacts are defined which offer services to support the business level. For example, TOGAF suggests to use an iterative process consisting of eight phases to develop an enterprise architecture. Based on the business goals which are defined in the first phase, different architectures need to be developed in the following order: business architecture (second phase), information systems architecture (third phase), and technology architecture (fourth phase). These architecture correspond to the defined layers. The business architecture is defined on the business layer. The information systems architecture consist of both the functional and information layer. The technology architecture is addressed on the infrastructure layer. Such a top-down approach assumes that (a) business requirements can be developed without being constrained by the concrete implementation or operationalization, and (b) supporting services can be created straightforwardly based on business requirements. However, when implementation-focused approaches are described (e.g., Service-Oriented Architecture), the existing infrastructure is often an important restriction on the services which can be provided. Consequently, the focus of such approaches is often on subjects such as legacy integration. Approaches which explicitly integrate an implementation approach with business goals (e.g., SOMA) therefore use a middle-out approach [13]. Such an approach takes into account the capabilities of the supporting systems, and attempts to find the best solution for the business requirements.

Indeed, the use of additional methods illustrates that a top-down approach in enterprise architectures needs complementary approaches to deal with complex environments. In a literature review on enterprise architecture projects, Lucke et al. identify several issues which motivate this need [14]. First, *complexity* is referred to as an underestimated issue. Not only the complexity of the models themselves, but also the dependencies between the different layers remain problematic [15,16,17,18]. Second, *rapidly changing conditions* imply that a top-down specification of an enterprise-wide architecture can become out-dated before it is even implemented [19,20,21]. Third, a top-down specification of the architectural layers results in issues regarding *scoping of architectural descriptions*. Rather than being straightforward, the identification of organizational and technical services to support business requirements is often considered to be problematic [17,20,22].

These empirical observations do not come as a surprise. Wide-spread enterprise architecture frameworks acknowledge the issue of dependencies between architectural layers. Nevertheless, they do not address this issue. For example, the Zachman framework acknowledges that the different cells describe abstractions of the same underlying organization, and that *dependencies* between the cells have an impact [23,24]. At a minimum, the cells are considered to be related to every other cell in the same row [24]. If a change in the structure of one cell affects the structure of another cell, a dependency between these cells exists. Moreover, such dependencies can occur not only within a row, but also between rows. The framework however does not provide guidance on how to determine the possible impacts between models from different cells. It is however acknowledged that the challenge during designing a model "is to design each while understanding the integrity of all others to avoid being surprised by undesirable side effects appearing long after it is possible to contain them" [24, p.595], and that understanding and storing the dependencies would "constitute a very powerful capability for understanding the total impact of a change" [24, p.603].

Moreover, the Zachman framework also introduces *constraints*, which represent the limitations lower-level layers impose on higher-level layers. The models from each perspective (i.e., each row) have a different set of constraints they need to adhere to [24]. For example, the models in the scope row are subjected to usability constraints (e.g., utility of the artifact), while models in the technology row are subjected to constraints from the state of the art of the used implementation platform. These constraints are additive across the different perspectives: constraints of a lower row also limit the models from higher rows. For example, the technological constraints on the technology models (e.g., only webservices are allowed) will also impact the system model, which will need to structure the system using services. When a constraint in a lower row is inconsistent with a model defined in a higher row, "the designers who are responsible for the two rows must initiate a dialog to determine what must be changed and to ensure that no gap in expectations exists between the different perspectives" [24]. Consequently, it is argued that the issue of conflicting constraints can be handled by

communication. Based on the reported empirical observations discussed above, this solution does not seems satisfactory. Moreover, the implications of adhering to an additional set of constraints are not expected to add complexity. When discussing the model transformation between adjacent rows, it is claimed that "as each model is structurally changed through the successive application of additional constraints, the original purpose of the business is not so obscured that the business requirements are not recognizable in the end product" [24]. Huysmans presents similar observations with regard to other wide-spread enterprise architecture frameworks, such as TOGAF and ArchiMate [25]. Therefore, a complementary approach is required to represent the impact of lower-level layers on higher-level layers.

Our suggestion to use modularity theory for this goal is not isolated. Several authors link the explicit decomposition of viewpoints in enterprise architecture frameworks to the ability to independently change artifacts. This indicates how enterprise architecture frameworks can be linked to the modularization of organizations. For example, business modularity is considered to be the highest level of enterprise architecture maturity [26]. On this maturity level, the role of IT in an enterprise architecture is to "provide seamless linkages between business process modules" [26]. Such business process modules allow "strategic experiments that respond to changing market conditions" [26]. Indeed, a modular business process should enable the execution of the *modular operators* described above. Based on a practitioners survey, it seems that many business users indeed expect an enterprise architecture to enable their ability to change in response to market conditions [2]. A modularity perspective can aid to specifically focus on the issues specified by Lucke et al. [14], which have been discussed above: identifying modular dependencies reveals how complexity is introduced when modules are added; the parallel evolution of modules limits the impact of rapidly changing conditions; the explicit specification of modules aids scoping architectural descriptions.

Consequently, modularity or independence between enterprise architecture layers should be achieved by defining business components and standardized interfaces based on the artifacts modeled in the different layers. This shows that an explicit focus on the coupling of artifacts from different enterprise architecture layers is required to gain insight in the kind of changes which can be supported. In the case studies we performed, adequately dealing with this kind of coupling often seemed to be an important success factor. An approach which explicitly shows these impacts can therefore help to improve insight in the change process.

3 Case Study Observations

In this section, we illustrate the occurrence and impact of modular dependencies between artifacts from different levels in an enterprise architecture. For our research intentions, we believe that a *qualitative* research approach using case studies is required. In order to study modular dependencies, profound insight in the organization is required, which cannot be obtained by quantitative descriptions. A qualitative approach is wide-spread in organizational modularity

research [27,28,29,30,31]. Argyres even argues that "insights generated from case studies, in many cases, outstrip the insights that have been generated by earlier work" [32]. The presented case studies are part of a larger group of case studies which aim to apply insights from modularity to enterprise architectures. These case studies have been performed adhering to the case study methodology [33]. Given the ambiguity of the definition of enterprise architecture in both practice and academic literature, and the large scope of enterprise architecture frameworks, it is difficult to clearly distinguish between the research subject itself and the organizational environment. Therefore, we selected an exploratory case study approach, since it is well suited for research goals where the boundaries between phenomenon and context are not clearly defined [33]. In the various cases, we have used the key informant method to identify informants who were highly knowledgeable about and involved in the enterprise architecture projects. The primary mode of data collection consisted of face-to-face interviews. In preparation for these interviews, various documents (e.g., documentation and presentation materials, documents on the organizational structure) have been consulted to gain an initial understanding of the organization and the project itself. A case study protocol was crafted, including an initial set of questions. These questions concerned various topics (e.g., architecture definition, expected benefits and barriers), in order to obtain a thorough description of the project. Follow-up questions took place via e-mail. During the interview, additional sources of evidence were collected, such as articles and internal documentation. The interview was digitally recorded and transcribed for future reference.

3.1 Case Study 1: Public Broadcasting Company

In a first case study, we observe a public broadcasting company (PBC) which is faced with changing customer demands. We focus on the division which is responsible for broadcasting news journals. Traditionally, this organization offers radio and television transmissions, which follow a clearly defined schedule. The radio and television business units import news items from different sources, such as feeds from external agencies, or items made by reporters. Items can be imported using physical tapes, digital files or through satellite transfers. The items are then edited and transmitted in the form of a news journal. However, since the introduction of the internet, customers demand personalized and real-time access to transmissions. Therefore, the content of news journals needs to be approached differently. The PBC decided to create a dedicated business unit to create an online channel, next to the existing radio and television units. This new business unit could reuse content from both the radio and television units, and create dedicated online news items as well. Adding the online channel was considered to be a necessary strategic move in order to serve a changing market. For the PBC, the radio, television and online business units are therefore situated on the *business layer* of their enterprise architecture.

In modularity terminology, this can be considered to be a modular operator, i.e., adding a module. Adding this additional business unit posed serious problems due to the supporting structure of the PBC on the lower enterprise

architecture levels. Here, we will discuss one aspect of an impact of the supporting structure. For example, accessing existing radio and television items proved to be complex, since they were stored in specialized applications. For every import channel (tape, digital or satellite), different applications needed to be used. This is a direct result of the principles used to develop the application portfolio, which was located on the *application layer* in the enterprise architecture. As a general principle, the organization always selects best-in-class software solutions for specialized media editing. This principle ensured the most efficient editing process. However, this results in an application portfolio which is not well integrated. Employees with specialized competences are required to operate these software packages. Therefore, in order to reuse audio and video fragments for the online channel, employees with these competences needed to be made available for the online business unit. This resulted in a duplication of skill sets, which was not efficient. The inclusion of employees with the competence to use the specialized software packages was not foreseen when the new business unit was defined. However, the coupling between software packages on the application layer and employee competence on the business layer necessitates this inclusion. Put differently, the required competences defined on the business layer need to be adapted to account for a dependency on the application level. It should be noted that coupling on the application layer is solved by adapting the artifacts which are defined on the business layer. Consequently, the ability to take strategic decisions to serve an emerging market are impacted by decisions made on the application layer. While the principle to select best-in-class applications may be justified for this sector and the performance of the organization, the lack of integration which it causes and the restrictions it places on business flexibility need to be understood as well. When considered as a modular structure, this dependency can be represented using a DSM. The DSM is presented in Figure 2. It shows how design parameters from the applications, which are part of the application layer, affect the design parameters of the business unit, which is part of the business layer. The discussed dependency of the competence for the applications on the organizational chart is highlighted in grey.

Since the PBC was unable to motivate the costs of specialized employees for the new channel, a structural solution was proposed. The dependency on the required application capabilities forces the the organization to deal with concerns from the application layer on the business layer, i.e., the different handling of tape, digital and satellite items. This dependency was removed by developing an abstraction layer on top of the application layer, which provided only the functionality needed on the business layer. This abstraction layer was defined based on concepts known in the business layer: the basic entity on this abstraction layer was a news item. A news items entity abstracts from the method which was used to import the source material. Therefore, it served as a kind of interface to the specialized software applications. For every application, audio and video fragments can be extracted, together with the required meta-data, in a uniform way. The specific functionality of the editing programs is not available through this interface. In order to use the editing functionalities, specialized competences

		Business Unit				Application			
		Organizational chart	Charter	Media reuse rate	News item granularity	Competence	Platform	Storage medium	Essence data collections
Business Unit	Organization chart	.	x			x			
	Charter	x	.						
	Media reuse rate		x	.	x			x	
	News item granularity		x	x	.			x	x
Application	Competence					.	x		
	Platform						.	x	
	Storage medium						x	.	
	Essence data collections							x	.

Fig. 2. The DSM for PBC

remain necessary. However, the need for employees with these competences is now not imposed on the business layer, only on the application layer.

This first case shows that decisions taken on the application layer of enterprise architectures can make changes to the business layer more complex. By adequately encapsulating the complexity from the application layer, dependencies between the different layers can be removed, and necessary services can still be offered. As a result, decisions on the business layer only need to deal with the inherent complexity imposed by the business environment, instead of dealing with complexity originating from the supporting layers as well. In this case, the number of modules is low and the dependency which limits the business layer is clear. However, this case provides a clear illustration of the issue we address in this paper.

3.2 Case Study 2: Gas Flow Manager Company

In a second case study, we observe an enterprise architecture project at a gas flow manager company. Here, we focus on a possible approach to resolve modular dependencies. The company offers gas transport services to its customers based on a grid consisting of entry points, nodes, and pipelines connecting the nodes. The functional and information layer of the enterprise architecture had to be rebuilt after the company was separated from a gas trading company. The legislation concerning liberalization of the energy market demanded this separation, as the company had to offer its gas transport services to other gas trading companies as well. Prior to the architectural redesign, IT was generally considered to be a bottleneck during the implementation of changes by business users. The new

architecture needed to be able to respond better and more quickly to changing business requirements and had to be understandable for business users, so they could more realistically estimate the impact of the changes they requested. Therefore, it was decided to clearly align the functional architecture with a high-level enterprise architecture model on the business level. This model needed to be constructed in such a way that the *stable* operation and stakeholders of the organization are represented. If the subsequent changes required changes in the high-level model, it would not be useful as a basis for the functional architecture. The issue of rapidly changing conditions has indeed been addressed as an important enterprise architecture issue by Lucke et al. [14]. Put differently, business changes need to be attributed to the implementation of the model elements, not on the nature of the elements itself. The model which was designed is shown in Figure 3 as a Role-Activity Diagram (RAD). The modeled activities represent generic descriptions of the business-level construction of the organization (e.g., *Define commercial services*). An abstract RAD model describes how the organization works, without detailing how the specific activities are implemented [34, p. 234]. Internally executed activities are represented by gray boxes. Activities which require collaboration with external partners are represented using white boxes in the collaborating entities, which are connected by a solid line. The arrows represent the process flow. In order to achieve a well-aligned business and functional architecture, a separate application has been developed to support the scope of exactly one activity in this model.

In a previous publication, we focused on a repeatable and reproducible method to design such models [35], and the benefits which can be achieved with such an approach. That description was limited to the top layers of the enterprise

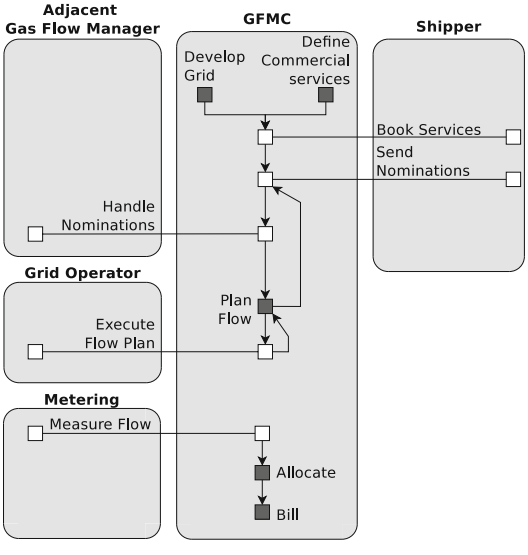


Fig. 3. Stable GFMC model

architecture, i.e., the business and functional layers. In this paper, we focus on the preconditions for applying this approach, which are situated on the functional and information layers. For this approach to work, the specification the functional layer may not be restricted by the information layer. Modular dependencies between artifacts from the information layer and artifacts from the functional layer can be used as an indication for such restrictions. Before the organizational split, applications had separate data sources, which impacted the specification of application scope. Put differently, dependencies existed between the functional and information layers. Because the semantic differences between entities in these data sources, it was not feasible to design applications which relied on data from different sources without creating complexity within the application. In the enterprise architecture maturity model, this indicates an architecture which is not correctly modularized [26]. For example, the entity *"customer"* existed in different data sources. In some sources, this entity referred to end consumers of the gas which is transported. In other sources, it contained the customers of the gas flow manager company (i.e., the gas traders). As a result, the scope of applications was defined based on the scope in the data sources. During the enterprise architecture project, a glossary was developed iteratively with business users from different areas to ensure consistent terminology across the organization. The glossary defines the entities, such as customers, and the relationships between each other. Based on this glossary, a database scheme was developed and imposed on the different data sources. Consequently, the dependency of the application scope on the data source scope was resolved. The coupling between the functional and information layer was thus removed. As a result, the modules on the functional layer could be designed to be well-aligned with the business layer, without being restricted by aspects from the information layer.

Moreover, the solution applied in this case follows the solution presented in modularity theory. In order to resolve modular dependencies, modularity theory proposes to specify an *architectural rule* [4]. Such a rule limits the design freedom for the implementation aspects of modules by prescribing a certain design choice. In this case, the glossary limits the allowed interpretation of, for example, the *"customer"* entity. As a result, no conversions need to be performed to ensure a correct use of data entities. Other solutions are possible to resolve dependencies as well. Currently, we focus on the ability of modular dependencies to identify coupling between modules of different architectural layers as a cause for restrictions on higher-level layers. In future research, we elaborate on different methods to deal with this kind of coupling. We elaborate on this approach in Section 4. However, the current case illustrates how a modularity approach can be complementary to enterprise architecture frameworks. While the different applications and data sources can be represented in an enterprise architecture, no indications of the coupling between the different viewpoints can be represented. When we consider the applications and data sources as modules, we can use a Design Structure Matrix (DSM) to represent aspects of the module implementation which affect each other.

3.3 Case Study 3: Data Usage in Governmental Processes

In the previous case studies, we illustrated the relevance of modular dependencies in enterprise architecture projects and how they could be resolved. We now apply the insights gained from the analysis described above to a case study in a governmental organization. The mission of the organization is to introduce and implement e-government solutions. To achieve this goal, it undertakes projects in the field of back-office reengineering, and tries to leverage these improvements by supporting projects with governmental partners. In this paper, we focus on a project that improves the way data from various sources is used in governmental processes. We will refer to this project as the Data Usage in Governmental Processes (DUGP) project. Similar to the description in Section 3.2, the structure of the data sources limits the design of governmental processes. Because of the political situation, different data sources are controlled by different governmental entities, which belong to governments on different levels (e.g., federal, regional, local level). As a result, different implementations exist for a large amount of design parameters. For example, the data delivery design parameter may be implemented by an online web interface, an FTP transfer, or through web services. Consider the partial Design Structure Matrix represented in Figure 4, which has been developed to describe the situation before the DUGP project. In this DSM, multiple design parameters are considered simultaneously. Compare this to the coupling in the GFMC enterprise architecture, where we focused on a single design parameter (i.e., data semantics). Dealing with multiple design parameters concurrently greatly increases the complexity of resolving dependencies.

The "x"-es with the grey background represent (1) the dependency of the data retrieval design parameter of the processes on the data delivery design parameter of the data sources, and (2) the dependency of the data syntax used in the processes on the data syntax used in the data sources. Consider the impact of these two design parameters in the following example. A process to request construction premiums requires personal data of the citizen requesting the premium (from data source A) as well as geographical data of the construction site (from data source B). Since the databases which contain the personal and geographical data are not integrated or standardized, various conversions between the implemented data parameters of these data sources may be necessary. Suppose that the information required from data source A needs to be obtained by invoking a single web service call, providing the address from the end user using four data fields (street name, street number, bus number, city name). In order to query geographical information in data source B, a request file containing two data fields (street name and street number, and ZIP code) has to be transferred using the FTP protocol. Since the construction premium process depends on both data sources, it needs to be able to communicate using two different versions of address data syntax, and two different versions of data retrieval technology. If the construction premium process needs to be changed, and an additional data source is required, the process owner needs to be aware of the data syntax and data retrieval method offered by the new data source. Moreover, when changes are made to these implementation aspects of data sources, additional versions

		Process				Data Source			
		Process Throughput	Data Retrieval	Data Syntax	Data Dictionary	Capacity	Data Delivery	Data Syntax	Data Dictionary
Process	Process Throughput	.	x			x			
	Data Retrieval	x	.	x		x	x		
	Data Syntax	x		.	x			x	x
	Data Dictionary				.			x	x
Data Source	Capacity					.	x	x	
	Data Delivery					x	.	x	
	Data Syntax						x	.	x
	Data Dictionary							x	.

Fig. 4. DSM before the DUGP project

need to be supported by the processes. The resulting complexity of these conversions is a barrier for the use of these data sources. Moreover, this example shows that different design parameters are intertwined in a certain implementation. As a result, it is hard to resolve these dependencies individually. The goal of the DUGP project is therefore to eliminate these dependencies at once in order to reduce the complexity of using data sources in governmental processes. However, the solution which is suggested by modularity theory, i.e., the definition of architectural rules, was not feasible because of the governmental structure. The different data sources are controlled by different governmental entities, who can decide independently on the implementation of design parameters. Declaring an architectural rule for a design parameter therefore requires an agreement between all governmental units responsible for a data source. However, since most units rely heavily on legacy systems to provide data services, changes to the implementation of design parameters are not easily realized. Therefore, it is difficult to reach such an agreement if an organization which can impose rules to these governmental units is not in place.

Consequently, the e-government organization developed a platform to consolidate data sources and provide uniform data access. Similar to the PBC case discussed in Section 3.1, an abstraction layer was developed to offer the required functionality without exposing the complexity of the layer offering these services. This abstraction layer provides services from the information layer to governmental processes, which are considered to be on the functional layer. The platform is based on two existing data sources from the federal government. Data from these data sources will be augmented with data available in data sources from other governments (e.g., geographical data, which is offered by regional governments).

The first data source which is used focuses on data concerning organizations. In this data source, data such as registration number, official addresses and legal statute of enterprises can be obtained. We will refer to this data source as the Data Source for Organizations (DSO). The Federal Public Service Economy is responsible for this data source. The second data source offers data concerning individuals. It refers to data such as employment and social status of citizens. We will refer to this data source as the Data Source for Individuals (DSI). This data source is governed by a separate organization created by the federal government. The platform will maintain this distinction, and offer its data services grouped in an Enhanced Data Source for Organizations (EDSO) and an Enhanced Data Source for Individuals (EDSI).

Since the EDSO relies on the DSO for its data, and the EDSI relies on the DSI, they need to consider the implementation of these data sources. Many implementations of design parameters are quite different. For example, the DSI has webservices available to query its data. As a result, these webservices can be used to develop webservices in the EDSI. These webservices are not directly offered in their original form. Instead, a facade pattern is used. This enables the creation of a uniform web service syntax throughout the platform. Otherwise, a dependency on the data syntax design parameter would be introduced. In contrast, the DSO has no webservices available. It is a mainframe which operates using batch requests. Therefore, a copy is made from the original DSO every night. This copy is then augmented with data from other governmental authorities, and used as a central database on which the services from the EDSO are provided. In order to simplify data access, the new platform provides three data delivery methods which will be available for all data sources: data repositories, an online application and webservices. Customized data repositories are large data files, which are copied to the process owner. After this initial data provision, automatic updates are transferred when data changes. These repositories are offered to enable process owners to incorporate the data from the new platform in their processes, without having to implement a webservices-based data access. Since many organizations are accustomed to using their own data sources in their processes, a customized data repository can be implemented without many changes in the processes. However, the unauthorized data sources which have been collected by the organizations themselves will then be replaced with authentic data. The online application allows for manual consultation of the data with a much smaller granularity: instead of a single large data file, only the result of a single query is returned. The same result can be obtained automatically through the use of webservices. Webservices offer the same data granularity, but can be implemented to automate processes.

From a modularity perspective, the platform can be considered as an additional module to eliminate dependencies between data and process modules. The DSM for the DUGP project is shown in Figure 5. When comparing DSM of the platform in Figure 5 with the DSM in Figure 4, we can conclude that some of these dependencies are indeed eliminated. Consider for example the data syntax and data delivery. We included an empty grey background to mark the previous

		Process				Platform				Data Source			
		Process Throughput	Data Retrieval	Data Syntax	Data Dictionary	Capacity	Data Delivery	Data Syntax	Data Dictionary	Capacity	Data Delivery	Data Syntax	Data Dictionary
Process	Process Throughput	.	x			x							
	Data Retrieval	x	.	x		x							
	Data Syntax	x		.	x				x				
	Data Dictionary				.				x				x
Platform	Capacity					.	x						
	Data Delivery					x	.	x					
	Data Syntax						x	.	x				
	Data Dictionary				x			x	.				
Data Source	Capacity									.	x	x	
	Data Delivery									x	.	x	
	Data Syntax										x	.	x
	Data Dictionary											x	.

Fig. 5. DSM of the DUGP project

existence of these dependencies. The syntax of webservices offered by EDSI is decoupled from the naming conventions of the DSI by using a facade pattern. As a result, naming conventions can be kept internally consistent with custom-built webservices for EDSO. The data syntax can be considered as an architectural rule which is maintained by the e-government organization. By adhering to this data syntax, process owners no longer need conversions between different data syntax versions. Another example is the data delivery design. In data sources from the platform, data can be provided through customized data repositories, an online application or webservices. Here, a single design option has not been selected, but the consistent offering of all data delivery techniques allows process owners to implement a single design for data delivery. Again, process owners no longer depend on the specific data delivery technique of the individual data sources. Consequently, it seems that the platform aids to decouple the process owners from the design decision of the data sources.

However, as stated by Baldwin and Clark, eliminating all dependencies in a modular structure is not a trivial task [4]. Consider the design option *process throughput* in the case of an automated process. Our respondents indicated that the number of processes which can be supported is limited by, amongst others, the capacity of the data delivery implementation. In Figure 5, this is visualized by the "x" where the column of the capacity of the platform intersects with the row of process throughput. A possible data delivery implementation in the

platform are webservices. As described above, webservices from the platform can be either custom-built by the e-government organization (e.g., webservices for EDSO), or can be part of a facade-pattern, calling underlying webservices (e.g., webservices for EDSI). The custom-built webservices operate on a local database. Consequently, their capacity is limited by the servers of the e-government organization itself. However, webservices which are part of the facade pattern are dependent on the capacity of the underlying services of DSI. Based on the implementation, issues with webservice capacity need to be discussed with the e-government organization or with the organization responsible for the original data source. The difference between the implementation of webservices in the platform is based on the available data delivery techniques from the original data sources. In Figure 5, this is visualized by the "x" where the column of the data delivery of a data source intersects with row of the capacity of the platform. It therefore seems that an unexpected dependency can be identified: when the platform is used, the process throughput design decision is impacted by the data delivery technique of the original date source. When the data delivery of DSO is changed (e.g., webservices become available) and used by the platform, process performance may be impacted. This is an example of a chained design dependency which propagates through the design structure matrix. Such dependencies are difficult to trace and to account for in change projects.

4 Towards Resolving Modular Dependencies

In this paper, we use design structure matrices to represent modular dependencies in order to understand which artifacts can be an obstacle during organizational changes. We already described how the organizations in these cases handled the identified issues. While modular dependencies can be resolved individually (as shown in Section 3.1), or by following solutions proposed in modularity (as shown in Section 3.2), it is clear no generalizable approach to dealing with modular dependencies has emerged yet. Because of the organizational context, traditional modularity solutions cannot always be applied. For example, political influences prevented the introduction of architectural rules in the DUGP case study in Section 3.3. It is clear that organizational modularity cannot always be approached similarly as in, for example, product modularity. Moreover, the wide variety of aspects which need to be considered increases the impact of chained dependencies, as shown in Section 3.3. Such chained dependencies greatly increase the complexity of a modularity analysis.

In the three case studies presented in this paper, it has been shown that by itself, the identification of modular dependencies allowed a deep understanding of phenomena concerning organizational changes. Interpretations are not limited to superficial models indicating a number of modules and their interfaces. Rather, modular dependencies have been shown to be far more complex at the organizational level than accounted for in current enterprise architecture frameworks. Indications from the cases are that they exist both in top-level models,

and in transformations or mappings to lower levels. However, the overall importance of modular dependencies is not yet completely clear at the organizational level: we do not know how many dependencies are present, and to which extent it is desirable to eliminate them. Nevertheless, the identification of modular dependencies by itself has already resulted in some practical applications. For example, a research project is currently being performed in which anticipated changes are defined and evaluated in the current enterprise architecture using scenarios. In software engineering, this is a well-known principle, and methodologies for analyzing the evolvability of software architectures such as SAAM are based on this. SAAM uses scenarios to analyze architectures with respect to achieving quality attributes, such as evolvability. The three important steps in SAAM are: (1) scenario and architecture description; (2) indirect scenario analysis; (3) scenario interaction. Direct scenarios are scenarios which are handled adequately (with regard to the required quality attributes) by the architecture without modifications. Anticipated changes are a way of describing these scenarios. Indirect scenarios are scenarios that cannot be handled by the analyzed architecture. Therefore, changes will have to be made to certain components. Scenario interaction occurs when different scenarios have to perform changes on the same components. Design structure matrices of relevant organizational modules help identify indirect scenarios and scenario interactions. Structured DSM walkthroughs with relevant stakeholders seem be an efficient means to achieve this. Similar approaches have already been used to test quality attributes of enterprise architectures [36]. However, these approaches lack a representation of modular dependencies.

Moreover, research efforts are currently ongoing on how to prevent modular dependencies. However, this is a very difficult task, and cannot be performed across all the levels of an enterprise architecture at once. Therefore, research focuses on different subdivisions of enterprise architectures. Currently, results are already published on the *software* and *process* levels. On the process level, different guidelines are proposed which are proven to prevent certain dependencies [37]. On the software level, such guidelines are available as well [9]. Moreover, software patterns which are free of so-called combinatorial effects (i.e., a more specific kind of modular dependencies) are already available [8]. The long-term goal of this research is to present organizational patterns which do not contain unknown modular dependencies. However, many different aspects have to be considered in such a pattern. It has already been described that organizational patterns should deal with (a) supporting technologies; (b) knowledge, skills and competences; (c) money and financial resources; (d) human resources, personnel and time; (e) infrastructure; and (f) other modules or information [38]. The approach for representing modular dependencies presented in this paper contributes to this approach by identifying problematic aspects in organizational modules.

5 Conclusions

In this paper, we explored a concrete application of modularity on the organizational level. We have shown that by modeling modular dependencies, in-

teractions between layers in enterprise architecture models can be represented. Such interactions are not explicitly focused on in enterprise architecture frameworks. Therefore, this approach is complementary to existing frameworks. These frameworks usually focus on the top-down specification of different viewpoints. However, these viewpoints can not be considered to be independent from each other in complex organizations. As a result, one needs to be aware of the impacts and restrictions imposed by lower-level layers during a top-down specification of enterprise architecture models. Therefore, research on this subject should be based on observations in real-life case studies instead of on theoretical examples. We focused on the restrictions of modeling artifacts on higher-level layers based on dependencies on the implementation of design parameters of lower-level artifacts. First, we demonstrated how this effect occurs in the PBC case study. Second, we discussed how the elimination of such dependencies can be a prerequisite in successful enterprise architecture projects. We illustrated this prerequisite in the context of the GFMC case study, which was published earlier. Moreover, we explored the applicability of dealing with modular dependencies as suggested by modularity literature. This solution implies the definition of architectural rules to limit the implementation possibilities of design parameters. Consequently, artifacts which are dependent on these design parameters can assume that a fixed implementation will always be supported. Third, we applied the insights from the observations in these case studies more concretely to the DUGP project. We showed that a DSM can be used to represent relevant issues for the enterprise architecture project as modular dependencies. This perspective allows to objectivate the issues which are resolved by the project. Moreover, a structured analysis can lead to the discovery of remaining issues after the project. Remaining issues can be unresolved dependencies, or newly introduced dependencies. This was illustrated by identifying the *capacity* design parameter as a chained dependency.

Future research needs to be conducted to gain insight on how modular dependencies on this level can be dealt with. In some cases, the definition of architectural rules seems appropriate. However, instead of choosing a single implementation option, it has already been observed that a consistent offering of multiple implementations can be required as well. Moreover, in some cases, imposing architectural rules does not seem feasible, because of the organizational structure. In the DUGP project, it was impossible to impose architectural rules to different organizational units. Therefore, an additional module was added. This approach resembles a bus pattern which is frequently used in the design of, for example, IT systems. However, the introduction of new dependencies using this bus pattern shows that a structured approach is required to adequately resolve modular dependencies. Therefore, we elaborated on future research, which will focus on a structured approach to resolve modular dependencies.

References

1. Barjis, J., Wamba, S.F.: Organizational and business impacts of rfid technology. Business Process Management Journal 16(6), 897–903 (2010)
2. Schekkerman, J.: Trends in enterprise architecture: How are organizations progressing? Technical report, Institute For Enterprise Architecture Developments (2005)
3. Campagnolo, D., Camuffo, A.: The concept of modularity in management studies: A literature review. International Journal of Management Reviews 12(3), 259–283 (2010)
4. Baldwin, C.Y., Clark, K.B.: Design Rules. MIT Press Books, vol. 1. The MIT Press (January 2000)
5. Baldwin, C.Y., Clark, K.B.: The value, costs and organizational consequences of modularity. Working Paper (May 2003)
6. Galunic, D.C., Eisenhardt, K.M.: Architectural innovation and modular corporate forms. The Academy of Management Journal 44(6), 1229–1249 (2001)
7. Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Communications of the ACM 15(12), 1053–1058 (1972)
8. Mannaert, H., Verelst, J., Ven, K.: The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability. Science of Computer Programming 76(12), 1210–1222 (2011)
9. Mannaert, H., Verelst, J., Ven, K.: Towards evolvable software architectures based on systems theoretic stability. Software: Practice and Experience 42(1), 89–116 (2011)
10. Simon, H.A.: The architecture of complexity. Proceedings of the American Philosophical Society 106(6), 467–482 (1962)
11. The Open Group: The open group architecture framework (togaf) version 9 (2009), http://www.opengroup.org/togaf/
12. Schöenherr, M.: Towards a Common Terminology in the Discipline of Enterprise Architecture. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 400–413. Springer, Heidelberg (2009)
13. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: Soma: a method for developing service-oriented solutions. IBM Syst. J. 47, 377–396 (2008)
14. Lucke, C., Krell, S., Lechner, U.: Critical issues in enterprise architecting - a literature review. In: Proceedings of AMCIS 2010 (2010)
15. Delic, K., Riley, J., Faihe, Y.: Architecting principles for self-managing enterprise it systems. In: Proceedings of the Third International Conference on Autonomic and Autonomous Systems (2007)
16. Lam, W.: Technical risk management on enterprise integration projects. Communications of the Association for Information Systems 13, 290–315 (2004)
17. Meilich, A.: System of systems (sos) engineering and architecture challenges in a net centric environment. In: Proceedings of the International Conference on System of Systems Engineering (2006)
18. Rhodes, D., Ross, A., Nightingale, D.: Architecting the system of systems enterprise: Enabling constructs and methods from the field of engineering systems. In: Proceedings of the 3rd Annual IEEE Systems Conference (2009)
19. Dreyfus, D.: Information system architecture: Toward a distributed cognition perspective. In: Proceedings of ICIS 2007 (2007)
20. Kaisler, S.H., Armour, F., Valivullah, M.: Enterprise architecting: Critical problems. In: Proceedings of the 38th Hawaii International Conference on System Sciences, vol. 8. IEEE Computer Society, Los Alamitos (2005)

21. Shah, H., Kourdi, M.: Frameworks for enterprise architecture. IT Professional 9(5), 36–41 (2007)
22. Armour, F., Kaisler, S., Getter, J., Pippin, D.: A uml-driven enterprise architecture case study. In: Hawaii International Conference on System Sciences, vol. 3, p. 72b (2003)
23. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. 26(3), 276–292 (1987)
24. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Syst. J. 31(3), 590–616 (1992)
25. Huysmans, P.: On the Feasibility of Normalized Enterprises: Applying Normalized Systems Theory to the High-Level Design of Enterprises. PhD thesis, University of Antwerp (2011)
26. Ross, J., Beath, C.M.: Sustainable it outsourcing success: Let enterprise architecture be your guide. MIS Quarterly Executive 5(4), 181–192 (2006)
27. Camuffo, A.: Rolling out a "world car": globalization, outsourcing and modularity in the auto industry. Korean Journal of Political Economy 2, 183–224 (2004)
28. Djelic, M.L., Ainamo, A.: The coevolution of new organizational forms in the fashion industry: A historical and comparative study of france, italy, and the united states. Organization Science 10(5), 622–637 (1999)
29. Miozzo, M., Grimshaw, D.: Modularity and innovation in knowledge-intensive business services: It outsourcing in germany and the uk. Research Policy 34(9), 1419–1439 (2005)
30. Salvador, F., Forza, C., Rungtusanatham, M.: Modularity, product variety, production volume, and component sourcing: theorizing beyond generic prescriptions. Journal of Operations Management 20(5), 549–575 (2002)
31. Thyssen, J., Israelsen, P., Jørgensen, B.: Activity-based costing as a method for assessing the economics of modularization–a case study and beyond. International Journal of Production Economics 103(1), 252–270 (2006)
32. Argyres, N.S.: The impact of information technology on coordination: Evidence from the b-2 "stealth" bomber. Organization Science 10(2), 162–180 (1999)
33. Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. Sage Publications, Newbury Park (2003)
34. Ould, M.: Business Process Management, a Rigorous Approach. The British Computer Society, Swindon (2005)
35. Huysmans, P., Ven, K., Verelst, J.: Designing for innovation: using enterprise ontology theory to improve business-it alignment. In: Proceedings of the 1st International Conference on IT-enabled Innovation in Enterprise (ICITIE 2010), pp. 177–186 (2010)
36. Johnson, P., Johansson, E., Sommestad, T., Ullberg, J.: A tool for enterprise architecture analysis. In: IEEE International Enterprise Distributed Object Computing Conference, pp. 142–154. IEEE Computer Society, Los Alamitos (2007)
37. Van Nuffel, D.: Towards Designing Modular and Evolvable Business Processes. PhD thesis, University of Antwerp (2011)
38. De Bruyn, P., Mannaert, H.: Towards applying normalized systems concepts to modularity and the systems engineering process. In: Proceedings of the Seventh International Conference on Systems, ICONS (2012)

Calculating the Application Criticality and Business Risk from Technology Obsolescence

Cameron Spence[1,*], Vaughan Michell[2], and Daniel Spence[3]

[1] Capgemini, 1 Forge End, Woking, Surrey, UK
cameron.spence@capgemini.com
[2] Informatics Research Centre, Henley Business School, University of Reading, UK
v.a.michell@reading.ac.uk
[3] Independent Consultant
dj.spence88@gmail.com

Abstract. The problem of technology obsolescence in information intensive businesses (software and hardware no longer being supported and replaced by improved and different solutions) and a cost constrained market can severely increase costs and operational, and ultimately reputation risk. Although many businesses recognise technological obsolescence, the pervasive nature of technology often means they have little information to identify the risk and location of pending obsolescence and little money to apply to the solution. This paper presents a low cost structured method to identify obsolete software and the risk of their obsolescence where the structure of a business and its supporting IT resources can be captured, modelled, analysed and the risk to the business of technology obsolescence identified to enable remedial action using qualified obsolescence information. The technique is based on a structured modelling approach using enterprise architecture models and a heatmap algorithm to highlight high risk obsolescent elements. The method has been tested and applied in practice in three consulting studies carried out by Capgemini involving four UK police forces. However the generic technique could be applied to any industry based on plans to improve it using ontology framework methods. This paper contains details of enterprise architecture meta-models and related modelling.

Keywords: Enterprise Architecture, Application Criticality, Obsolescence, Risk, Modelling.

1 Introduction

As the pace of technology introduction quickens and IS becomes more pervasive the rate of change of technology and the forced obsolescence has also increased [1]. This very pervasiveness increases the connectedness and reliance on specific technology which can quickly become obsolete [2]. This increases both the cost of maintaining existing and or replacing the technology [3]. Doing nothing is not possible due to risk

* Corresponding author.

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 91–112, 2012.

of loss of service provision and hence the management of obsolescent technology is becoming critical. The current economic focus on austerity has increased the need for better obsolescence awareness and management as businesses seek to consolidate and reduce their costs whilst maintaining their technology competitiveness.

Much of the existing literature regarding obsolescence has focused on its definition and relationship to specific business contexts. [4] has focused on evaluation of the performance and eventual failure of obsolescent infrastructure facilities (e.g. public works, sewers, pavements etc) and the need to establish reliable design service life metrics. [3] explores the obsolescence of electronic integrated circuit components such as DRAMs, via an adapted stages of growth model that includes obsolescence. Whelan [2] discusses the impact of obsolescence on stock management and the value and effective mathematical productivity of stock ranging from computers to industrial machinery and its relationship to computer usage. Feldman [5] looked into the problems of obsolescence with respect to the parts procurement lifecycle to improve algorithms for parts forecasting and management. [6] examined strategies for evolving existing software and included obsolescence as a factor along with quality, economic and data value in their metrics. Similarly [7] includes technical obsolescence as a factor in a similar paper on software productivity and reuse. Little work has been conducted to develop a practical methodologies approach to finding obsolete software and related components.

Obsolescence results in an inability to meet performance criteria [4], for example when the requirement has moved on, or the technology has been superseded. Our concern is with the former. We propose an approach to identifying and managing obsolescence and risk using a simplified enterprise architecture and heat map approach that can be scaled to different size companies. The approach has also been successfully trialled to identify and manage IS infrastructure obsolescence in a specific business case of a police service where low cost obsolescence risk assessment was required to support change decisions.

There are three main issues to address when considering the impact of technology obsolescence on a business: we need to define obsolescence and its impact factors; we need a way of identifying the other aspects of technology and services impacted by that obsolescence; and finally, we need a way of identifying the greatest risk to the business from those obsolete technologies and services. These three issues are addressed sequentially in the next section on approach.

2 Approach

2.1 Obsolescence and Impact

What Is Obsolescence?

As Sandborn [8] suggests definitions of software obsolescence for commercial off the shelf (COTS) products vary, as there is often a big gap between the end of the product sale date and the date of the end of the support for the product. The withdrawal of support however, may not lead to immediate loss or degradation, but acts as a trigger for the business to make the necessary decisions to preserve the capability provided

by the technology, subject to its criticality. For our purposes we will define obsolescence as the loss or impending loss of support for technology that reduces its ability to continue to function in the organisation [5]. Whilst the date of end of production of the technology and the date of end of support is reasonable to identify, the way in which the technology will be affected by degraded or no further support is more difficult to determine and will depend on the capabilities and resources of the organisation as well as the technology being considered [4].

The Impact of Obsolescence

Many organisations are running with a technology estate that is substantially obsolete. This causes problems of cost and risk. Increased costs are introduced by (a) the requirement for special support arrangements; (b) the extra development required to provide applications that can cope with obsolete technology (a notable case in point is the continued use of Internet Explorer 6 and Windows 2000 in the UK public sector); and (c) difficulty/high cost of obtaining support for very out-of-date technologies (both hardware and software).

Many factors may contribute to a high cost of ownership of an IT estate; it is often instructive to keep recursively asking the typical root cause analysis question "why" to determine root causes [9]. For example, high costs associated with running applications in data centres could be traced back to costs for the actual infrastructure (servers, switches etc.) and costs for the datacentres themselves (cooling, power, rent etc.) For the servers, then costs could be associated with both software and hardware – and the costs for both of these are likely to go up over time as the products on which they are based start to become obsolete and thus the subject of special (custom) support arrangements. Eventually the products involved will become unsupportable, which transforms an issue with high cost into an operational risk to the business. A root cause analysis in graphical format made this very apparent and understandable to the client:

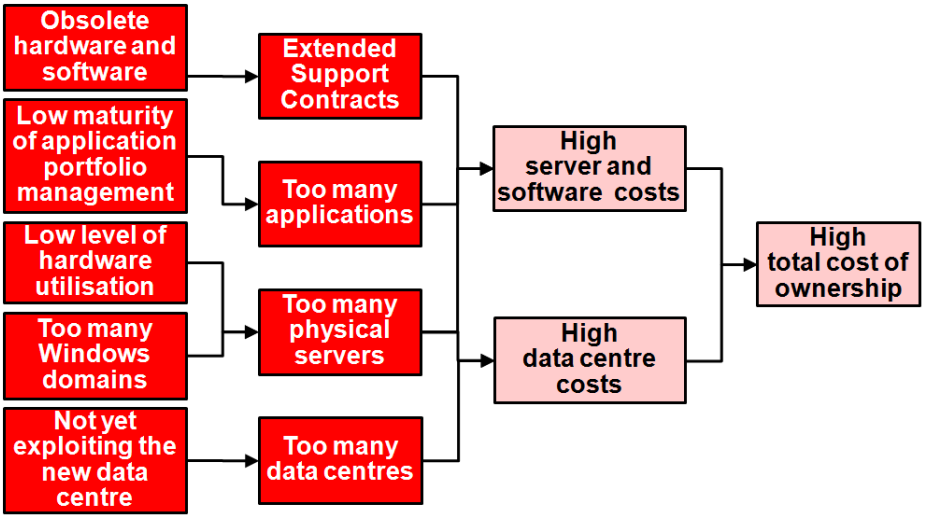


Fig. 1. TCO Root Cause Analysis

The increased costs and the lack of knowledge results in increased operational risk of service failure, especially where obsolete technology is a key part of the core service delivery capability of the business. This operational risk if not attended to can then result in reputational risk where business service performance is badly impacted by IS/IT failure. With no action there is a risk of degradation of business service capability through the inability to process and disseminate quality (correct, complete and timely) data and information through the enterprise. However, it is not enough simply to identify the technology components that are subject to obsolescence. We also need to understand how the technology supports the business and specifically the processes and services delivered by it. Critically we also need to quantify the importance of the relationship between in order to understand the impact of the obsolescence.

Open Source

Where a particular technology (typically software) is 'open source', this can lead to complications in the availability of support. The key question to consider however remains the same. If there is an issue with the technology, then who, if anyone, is obligated to address that issue. Some open source technology can be obtained from organisations who then offer support for that technology (for example, Red Hat offer support for their distribution of Linux). In this situation, arguable the situation with that open source technology is not really any different from a commercial product.

Where an open source technology has been adopted without a support contract as described above, then the organisation using that technology should then be asking the question: "what happens if we find a problem with it?". It might be that the organisation may have its own developers capable of rectifying the problem directly, perhaps in source code, in which case the organisation is support the technology directly, and the situation is the same as for any other technology (such as an application) developed and supported in-house.

If the technology is not supported externally or internally then one could argue that the technology is therefore *already* unsupported, and that as a consequence it should be seen as at risk.

Chain of Factors

A near obsolete IS component will be dependent on a cascade or chain of factors that determine the level of risk and impact of the obsolescence as seen in the four step chain in Figure 2.

To characterise the risk impact of the obsolescence we need to consider a number of factors. In the following discussion we will use the notation as follows:

Number of applications in total is denoted by **M**, thus

$$\exists \text{ applications } A_1, A_2, A_3, \ldots, A_{M-1}, A_M \tag{1}$$

Number of business services in total is denoted by **N**, thus

$$\exists \text{ business services } S_1, S_2, S_3, \ldots, S_{N-1}, S_N \tag{2}$$

We start the analysis by focusing on the business services that give context to the analysis of the supporting applications.

Criticality of the Business Service

We denote this by $\mathbf{CS_n}$ where $1 \leq n \leq N$.

This measure reflects how important a specific business service is in delivering value to a customer. Whilst many services provided by the business to the customer may be related as critical as they relate to the core business offering of the company others may be less critical. We need a measure of the criticality to the business that will reflect an impact on the potential obsolescence of the application/technology that supports it. This measure we will choose to make numeric, in the range 0 to 1, so that the most critical business services have the highest rating.

For example, in the policing sector, perhaps one of the most important services provide by the police is the ability to respond to an incident (perhaps a member of the public calling 999, for a life-or-death situation). This might be rated as having a criticality of 0.9 or 1.0 (or could be expressed as a percentage, i.e. 90% or 100%). Therefore, we have:

$$\forall \text{ n where } 1 \leq n \leq N,\ 0 \leq CSn \leq 1 \tag{3}$$

Clearly we would not expect to find any services that had a criticality value of 0 as that would be imply that the customers (internal or external) derived no value whatsoever from the provision of that service. For reasons that will become clear shortly, it is worth also viewing this as a simple one-dimensional matrix of business service criticality values:

$$\begin{bmatrix} CS_1 \\ CS_2 \\ \cdots \\ CS_{N-1} \\ CS_N \end{bmatrix}$$

Fig. 2. Matrix of business service criticality values

Criticality of the Application to the Business Service

This is a relative criticality value – it depends on which business service is being referred to, and so we will denote this by

$$\mathbf{CR_{m,n}} \text{ where } 1 \leq m \leq M \text{ and } 1 \leq n \leq N$$

In order to understand the impact of any obsolescence on the business we must understand how important the application/IS service is to the provision of specific services to the end customer of the business. For example if the service to the end customer is entirely provided by an IS application, and there is no possibility of the service continuing in the absence of the application (for example, by a manual workaround), then obviously the application is completely critical to the provision of that service.

Where a workaround is possible, but that might result in a degradation of service of, say, 60%, then we might choose to assign a criticality value of 60% (or 0.6) to the relationship between the service and the application. Generally speaking, the more important an application is to a business service, the higher value we would expect to see.

If the application is useful to the provision of the service, but only plays a minor role (a typical example would be email, which would be used by many business services), then a lower criticality value would be used, for example 0.2.

Clearly, not every application will be required by every business service, and so there will be many instances (ordered pairs of m and n) where $CA_{m,n}$ is 0 (meaning no correlation between the particular pair of application and business service).

Therefore, we have:

$$\forall \text{ m where } 1 \leq m \leq M \text{ and } \forall \text{ n where } 1 \leq n \leq N, \;\; 0 \leq CS_{m,n} \leq 1 \tag{4}$$

This can be viewed in a matrix representation, where each cell in the matrix represents the correlation between a business service (the columns) and the applications (the rows):

$$\begin{bmatrix} CR_{1,1} & CR_{1,2} & \dots & CR_{1,N-1} & CR_{1,N} \\ CR_{2,1} & CR_{2,2} & \dots & CR_{2,N-1} & CR_{2,N} \\ \dots & \dots & \dots & \dots & \dots \\ CR_{M-1,1} & CR_{M-1,2} & \dots & CR_{M-1,N-1} & CR_{M-1,N} \\ CR_{M,1} & CR_{M,2} & \dots & CR_{M,N-1} & CR_{M,N} \end{bmatrix}$$

Fig. 3. Matrix representation of application criticality values relative to the business services

It should be noted that this kind of mathematical treatment of the criticality of the applications in the context of the various business services that they support represents at this stage a theory as to how the method used thus far in practice may be further developed; the case studies discussed below have not in practice used this kind of algorithm (because the mathematics was developed afterwards), but have instead used a much simpler treatment that assigns a simple "High / Medium / Low" criticality rating to each application.

The combination of the criticality of a business service and the criticality of a particular application to that business service can be combined to give a view as to the overall criticality of the applications to the business. As a simple example, if application A_m is used in the provision of business services S_a and S_b, and no other business services, then we can see that

$$CR_{m,a} > 0 \quad \text{and} \quad CR_{m,b} > 0 \quad \text{and} \quad CR_{m,n} = 0 \; \forall \; n \neq a \text{ and } n \neq b \tag{5}$$

We can then calculate a contextual criticality value of application A_m from these two business services, which we will denote as $\mathbf{CA_m}$:

$$CA_m = CR_{m,a} \,.\, CS_a + CR_{m,b} \,.\, CS_b \tag{6}$$

For example:

> If business service S_a is deemed to have a criticality of 80%; and
> application A_m is deemed to be 70% critical to that service; and
> business service S_b is deemed to have a criticality of 30%; and
> application A_m is deemed to be 90% critical to that service;
>
> we would then calculate the criticality of application A_m to be:
>
> $$0.8 \text{ x } 0.7 + 0.3 \text{ x } 0.9 = 0.56 + 0.27 = 0.83$$

It can be seen from the above example that the highest contributions to the criticality value come from products where both multiplicands are close to unity; in other words, where a service is very important, and an application is very important to that service. This is the reason why we have selected this particular treatment.

This can be generalised to include all the business services that may (or may not) rely to an extent on a particular application, to produce the following definition for the criticality to the business of a particular application:

$$CA_m = \sum_{n=1}^{N} CR_{m,n} \,.\, CS_n \tag{7}$$

This can also be viewed as the product of the two matrices shown above, symbolically as correlation times service list equals service risk, in matrix form as:

$$\begin{bmatrix} CA_1 \\ CA_2 \\ \dots \\ CA_{M-1} \\ CA_M \end{bmatrix} = \begin{bmatrix} CR_{1,1} & CR_{1,2} & \dots & CR_{1,N-1} & CR_{1,N} \\ CR_{2,1} & CR_{2,2} & \dots & CR_{2,N-1} & CR_{2,N} \\ \dots & \dots & \dots & \dots & \dots \\ CR_{M-1,1} & CR_{M-1,2} & \dots & CR_{M-1,N-1} & CR_{M-1,N} \\ CR_{M,1} & CR_{M,2} & \dots & CR_{M,N-1} & CR_{M,N} \end{bmatrix} \begin{bmatrix} CS_1 \\ CS_2 \\ \dots \\ CS_{N-1} \\ CS_N \end{bmatrix}$$

Fig. 4. Refined application criticality calculated as the product of the application/service correlation values and the service criticality values

The product of these two matrices yields a list of the criticality to the business of each of the applications (A_1 has criticality CA_1, A_2 has criticality CA_2, and so on).

Thus, instead of what has been done to date in the studies mentioned below, where an absolute criticality value has been assigned to individual applications (based perhaps upon the perception of one individual), we have been able to produce a numerical value that sums the contribution made of that application across the whole set of services

provided by the business. This recognises the fact that although applications may be important in providing a service, they are not necessarily essential.

There are also other factors that could play a part in this, although further work is necessary to determine the best way of doing this:

Criticality of the Obsolescence, Co

Even if the obsolete technology is supported after obsolescence the service provided by it might be degraded, as it may not be able to provide the IS service performance of newer or competitor products. Hence we need a metric to define the level of degradation in some way, either as high/medium/low, or ideally in numeric form so that it can be added to the above mathematical treatment. The factors affecting risk can be divided into a number of areas:

- Increased risk of technology/application failure due to incompatibility and reliability issues due to obsolescence
- Lack of effectiveness e.g. due to lack of functionality compared with non obsolete technology (relates to effectiveness of service provision)
- Lack of problem solving and knowledge support due to withdrawing of supplier/manufacturer resources and skills necessary to provide them

Further work is required to determine a way in which this particular factor can be factored into an algorithm for business impact.

Proximity to Obsolescence, Po

The proximity of the technology component to obsolescence will depend on both the vendor support and maintenance to allow the product to function after obsolescence, although an upgrade path from a manufacturer or provider may allow the degradation to be reduced. We need factor in the degree of risk associated with the closeness to obsolescence. For example an announcement of pending replacement of a technology from a provider may be a starting point, with gradual reduction in support and perhaps eventual complete deletion of the product and its support.

Application Obsolescence Risk Factor, Af

We propose that the overall risk factor for the obsolescence of a specific application is therefore the combination of the relevant metrics Po, Co, and CA_m, possibly a product of them, although the method has not been developed sufficiently to indicate how best to do this.

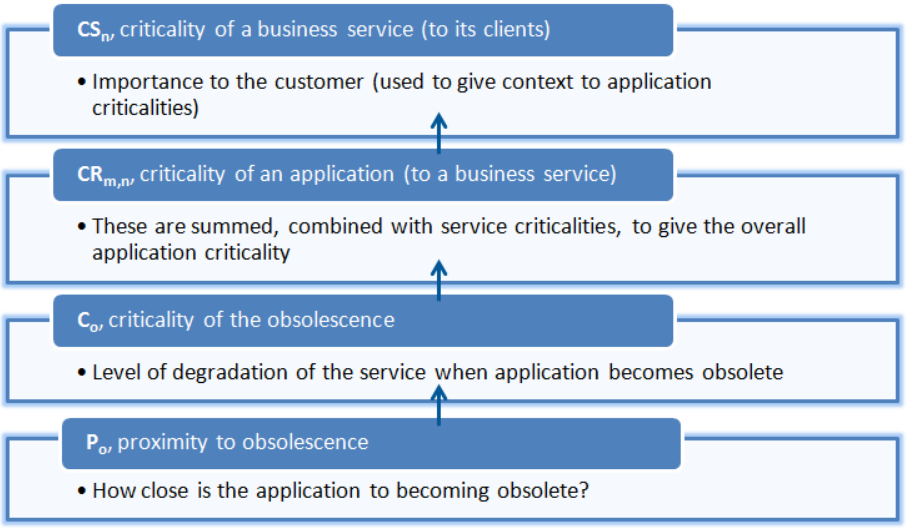


Fig. 5. Obsolescence Impact Factor Chain

2.2 Services and Technology Impacted

Identifying the Services and Technology Impacted
Information-intensive industries have very large amounts of IS/IT with attendant software and technology hardware and component risks. If the maturity of the architectural and operational processes have not kept track with the size of their IT estate, then there may be a lack of understanding of exactly what technology the organisation has, who is using it, and what its vulnerabilities are.

Understanding the IT Estate
The term "IT Estate" is generally used to refer to the complete portfolio of technology used within a business – including applications and infrastructure. If the knowledge of such an IT Estate is not encapsulated and maintained in some kind of 'living repository', then the lack of corporate knowledge can be exacerbated over time by changes to personnel within the organisation, so that key knowledge as to what exists and for what purpose, is lost as people's roles change. The federated nature of some large organisations can make this problem worse, because from the outset, no one part of the organisation ever has a complete picture of the business and technology architecture. The best that can be achieved in these circumstances is that individual parts of the organisation try to document solution architectures that capture their piece of the picture. A lack of knowledge of an Enterprise Architecture can make it hard to plan for the future, because without knowing the 'as-is' state, it is difficult to know what needs to change in order to achieve a 'to-be' state. Thus, it is possible to view the 'status quo' as a safer option, putting off the required upgrade and modernisation projects. Without knowledge of what applications are being used, by whom, and their underlying technologies, it is not possible to get a view as to the risks being posed to the business due to this obsolescence.

Case Studies

The aims of the first project, from the client's perspective, was (a) to understand the potential cost savings associated with rationalising the application platforms, and (b) understand the degree of risk associated with technology, and provide a roadmap for addressing it. Thus, the idea of heat-mapping the business risk is highly relevant because it provides part of the 'business case' for making the relevant upgrade / replacement projects to address the risks thus identified.

The aim of the second and third projects was to seek cost savings by identifying duplicate applications. However, the heatmaps were also relevant here, both in pointing out risks to the businesses, as well as assisting in the choice of applications to retain.

Enterprise Architecture Meta Model

It is necessary to gain an understanding of the business technology architecture estate in enough detail so that, obsolete technology types can be traced through to the applications relying upon them, and thence through to the business services and functions that in turn rely upon those applications. Many enterprise architecture models have been developed to make sense of business and technology components found across a variety of businesses [10] and aligning business services with IT capability [11]. Our specific issue requires a focused model that is easy for business and technical users to understand, but also shows dependencies between components. For this reason we have adapted and extended part of the TOGAF [12] content meta-model to build a set of artefacts and inter-relationships that are particularly relevant to our area of interest.

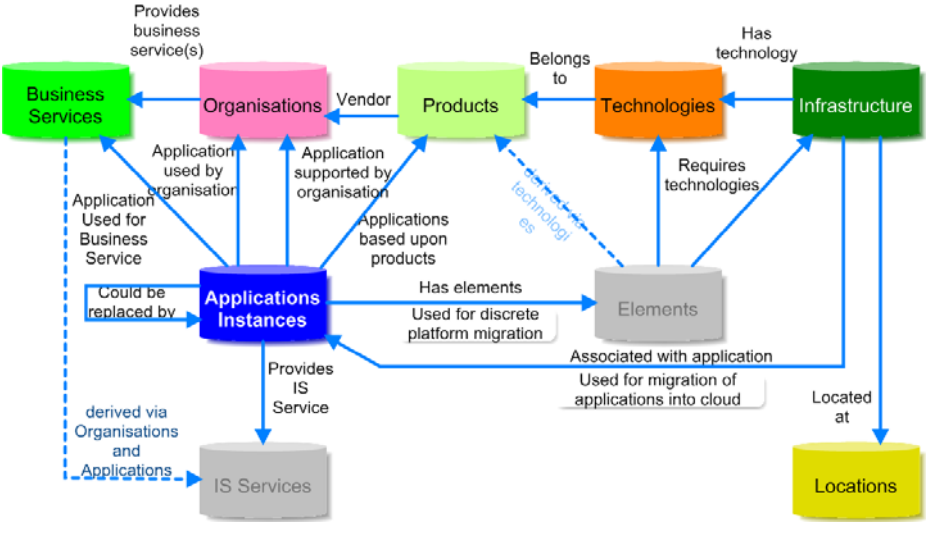


Fig. 6. Infrastructure-focused metamodel

Although standards exist for modelling Enterprise Architectures, it is frequently necessary to adapt those standards for different client situations. Reasons for this include: (a) the client prefers a particular standard (e.g. TOGAF or Zachman [13] or one of their own frameworks and (b) there are requirements for a particular engagement that demand a change to the standard model. In the first example discussed in this paper, the client was particularly interested in rationalising the application platforms rather than the applications themselves. Therefore, there was a heavy focus on infrastructure. The model used in the first project is illustrated below:

Figure 3 represents the actual model that was used to capture, analyse and report on various aspects of the IT portfolio for the first police force. With this particular client, two aspects of the model (shown in grey) were not used. For the other studies, heavy use was made of the IS Services component, critical for application de-duplication, and in the final study, an additional element was added for modelling the information stored within the applications (related to Master Data Management).

Parts of this model had previously been used with another client to carry out an analysis of their application portfolio with a view to rationalising that portfolio (removing duplication). This gave rise to the two elements focusing on Applications (or **Application Instances**) and **IS Services** (both taken from TOGAF). By definition, any two applications that are labelled as offering the same IS Services are duplicates; and one of the aims of that study was to aim for a 'minimum set' of applications that gave the full range of required IS Services (functionality).Part of the business case for rationalising applications is of course the cost of running those applications, and part of the cost of an application comes from the servers hosting that application. Servers are of course a particular kind of infrastructure, which is why the model includes **infrastructure**. This infrastructure resides in physical **locations**, which are important to know for a number of reasons, especially when part of the rationalisation design includes closing one or more data centres. This has no bearing on the question of obsolete technologies, but was critical to the analysis and creation of the rationalisation design.

The servers in the IT estate for any client will have installed on it a number of software **products**, for example applications, databases, middleware, operating systems, monitoring and so on. In addition, the servers themselves are of course products from a hardware manufacturer. Thus, the products need to include a list of all software and hardware in the IT estate. In practice, the terminology used to describe a particular product may be very different to the terminology used by auto-discovery software, which sometimes goes to the extent of looking at versions of libraries installed on the servers (for example, Dynamic Link Libraries, or DLL files, on Windows platforms). Examples of this are:

Table 1. Sample Product – Product Alias Mappings

Product	Product Alias
Windows Server 2003 Enterprise x64 (SP2)	Microsoft(R) Windows(R) Server 2003 Enterprise x64 Edition Version 5.2.3790 Build 3790 SP2
RDBMS 10g Release 2	Oracle Database Server 10.2.0.4.0
Solaris 10	SunOS 5.10

When looking up products on manufacturers' websites, the term in the left column needs to be used. However, when auto-discovery tools are run, the terms in the right column are those that are generated.

With this particular client, there were several thousand servers that needed to be matched up to products. Using this intermediate mapping meant that this could be done largely automatically. Once it had been calculated to which product a particular 'product alias' corresponded, then that mapping was automatically applied to all instances of that technology.

Another reason for the use of this intermediate layer was the fact that in many cases, there were multiple pieces of software that corresponded to the same product, which were 'discovered' separately. For example, two separate pieces of software were discovered ("Oracle Net Services (TNS) Listener 9.0.1" and "Oracle Database Server 9.0.1.0.0") that both corresponded to the same product ("RDBMS 9i Release 1").This intermediate mapping of **product aliases** provides the ability to cope with multiple synonyms and multiple pieces of technology that belong to a single product.

The intention of the **elements** artefact was to allow the modelling of major components of the application (e.g. web tiers, database tiers, business logic tiers, storage allocations) so that they could then be rationalised. This is not relevant to the obsolescence discussion, and was in fact only used (and renamed to **application components**) in the final study.

The final pieces of the model are both drawn from TOGAF. The **organisation** information allows us to represent the structure of an organisation, which is where the users of the applications reside. This is also useful for representing external organisations, for example the vendors of the products, or those involved in some way in supporting the applications.

The **business services** are the services provided by the business to its client. Clearly defining business services is necessary in order to be able to make a correlation between the services provided by the business and the IT that supports those services. Also the criticality of the service to the end client, will in turn affect the importance of the IT service and hence the impact and risk associated with the obsolescence of the technologies and components.

Populating the Model

The model was implemented using a particular modelling tool, MooD®. The functionality provided by this tool was critical to our ability to import, analyse and report on the data that was captured.

Many different information sources were used to populate this model, including but not limited to Active Directory, Tideway (auto-discovery software), a Configuration Management DataBase (CMDB) product and various spreadsheets populated manually.

The applications were populated using a combination of reports from the CMDB and spreadsheets provided by the client. The infrastructure was populated using a set of spreadsheets from various sources. The product aliases were populated using spreadsheets from the auto-discovery software. The products were populated partially manually, interpreting the product alias lists in the light of the team's knowledge of the marketplace, and partially using data from the CMDB.

The organisation was populated using information on the client's public web site.

The business services were populated from a generic UK police business service architecture published by the National Police Improvement Agency, called the "Policing Activities Glossary" [14]. This provided a hierarchical representation of the services provided by UK police forces, which we represented graphically in the modelling tool.

Tracing Obsolescence to Applications

Obsolescence as applied to technology (such as hardware and software products), using the definition previously offered, means "loss or impending loss of support for *hardware or software products* that reduces their ability to continue to function in the organisation". These products are produced by various organisations (vendors / manufacturers), who often specify a date beyond which support for their products will either cease, or become more restricted (and perhaps substantially more expensive). Some manufacturers specify "End of Support" (EOS) and "End of Extended Support" (EOES) dates for their products.

Fig. 7. Obsolescence from Products to Infrastructure

Starting from the products, therefore, and knowing that several products may relate (via the Product Alias intermediate layer) to a piece of Infrastructure (a server), then it is possible to say, for each piece of infrastructure, what is the earliest EOES date for any product that relates to that piece of infrastructure. For example, if that infrastructure was based upon a server model that had an EOES date of February 2013, but ran Windows 2000 that had an EOES date of 13th July 2010, then we can say that the earliest EOES date of these is the latter – so give the infrastructure as a whole, an EOES date of 13th July 2010. In other words, there is something about this piece of infrastructure that will be difficult and/or costly to support beyond that date.

The next step is to roll this up into the application layer:

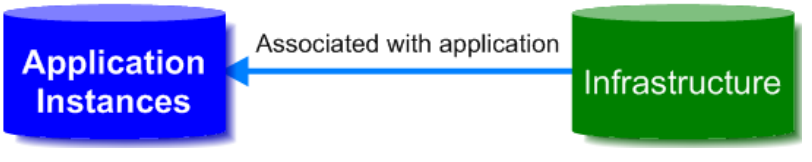


Fig. 8. Obsolescence from Infrastructure to Application Instances

Following the same approach, we can look at all the servers that are associated with a particular application (or instance of an application), and pick the earliest EOES date for each of these servers. In other words, we are saying that for a particular application, there is a particular piece of technology (software or hardware product) somewhere in the supporting infrastructure that will make application difficult and/or costly to support beyond that date.

2.3 Identifying Business Risk

An approach to identifying business risk from IT in general is discussed in [15]. This paper suggests a four-quadrant model for categorising risk, including notably the "avoid/prevent risks" which are viewed as being most critical because of the impact to the business should they be triggered, as well as the probability of them occurring. In this particular paper, we are focusing on the risk to the business from technology obsolescence that would fall within this particular quadrant. In other words, these are risks that a business can and should manage down to an acceptable level, and perhaps would if those business stakeholders were actually aware of their existence in the first place.

The use of 'heatmaps' is appropriate to demonstrate business risk, if backed up by proper evidence.

Algorithm for Generating a Risk Heatmap

The heat map, a colour-coded display of the intensity of a result has been used in various forms [16] to provide immediate visual understanding of multi-objective optimisation processes. It has been widely used in consultancy and problem solving as a means of highlighting critical obsolescence information in an easy to understand form [17] [18].

As discussed in the risk section, whilst the Obsolescence Impact Factor Chain helps to cover a range of appropriate factors, this can quickly become complex and costly and hence we adopt a simplified approach that assumes full support is provided by the IT service and focuses only on the criticality of the service and whether the technology is deemed obsolescent as defined by the client.

Continuing the algorithm started in the IT domain, it is possible to look at all the applications used to support a particular business service, at each of their EOES dates, which in turn are rolled up from infrastructure and products. We ask the question "are there any applications required by this service where the EOES date has already passed?". Where this is true, this obviously indicates that there is a degree of risk associated with the continued operation of that business service. The mapping of applications to business services can either be done via the organisation structure (i.e. for each organisational unit, determine which applications they use; and also which business services they provide) – used for our second study with the two police forces; or in the case of first study for a single police force, they were able to provide a direct correlation between applications and business services, as shown below.

However the mapping is done, the following algorithm was then used, which relies upon knowing the EOES dates for all the support applications, along with an indication of their criticality to the business (1 being the most important).

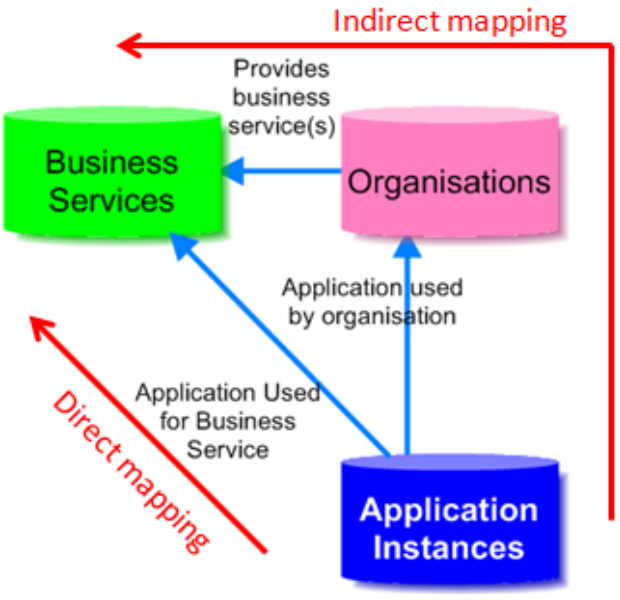


Fig. 9. Obsolescence from Applications to Business Services

```
FOR each business service
IF ∃ supporting applications of criticality 1 and EOES
date in the past
THEN
  SET risk to MAJOR RISK
ELSE
  IF ∃ supporting applications of criticality > 1 and
EOES date in the past
  THEN
    SET risk to SOME RISK
  ELSE
    IF we cannot associate any infrastructure with this
service
    THEN
      SET risk to UNKNOWN RISK
    ELSE
      SET risk to NOT AT RISK
    ENDIF
  ENDIF
ENDIF
```

Fig. 10. Algorithm for Calculating Business Risk in a Heatmap

It should be stressed here that the criticality of each application used in the above algorithm is a non-contextual, non-numeric one, and ignores the criticality of the application to all the business services that it supports. The contextual criticality values discussed previously represent new thinking carried out added after these particular case studies were carried out.

Like all of the algorithms in the model, this calculation was automated for all of the business services using the tool's ability to calculate and store intermediate results, including where necessary the ability to 'call out' to Excel. For example, the above algorithm looks like:

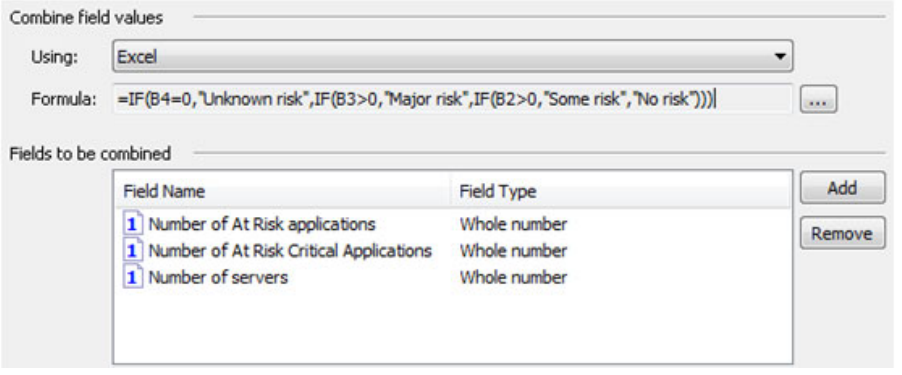


Fig. 11. Automated calculation of business risk using MooD

Risk Heatmap for a Police Force Requirement

By applying the above algorithm to each business service (from the PAG) in turn, it was possible to assign a risk value to each business service. The modelling tool was able to assign a colour to the business services dependent upon the risk value, and so the resulting heatmap looked like the following:

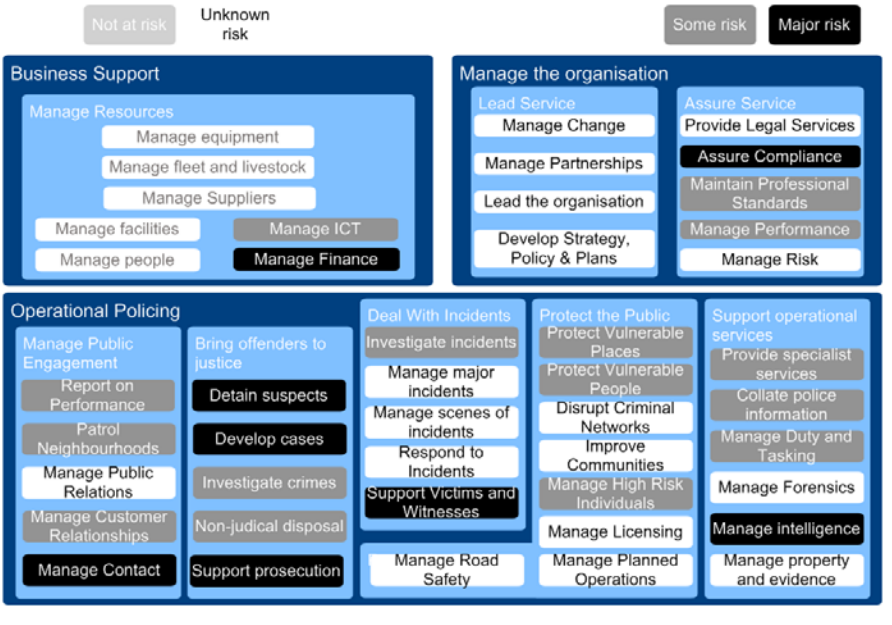


Fig. 12. Business Risk Heatmap (1)

This is for the first of the three studies: for a single police force, using a direct mapping from applications to business services. Every application was either at risk through some obsolete technology, or there was insufficient knowledge of the IT portfolio to determine whether there was any risk or not.

For the second study, an indirect mapping was used, via the organisation structure. The results obtained from this were:

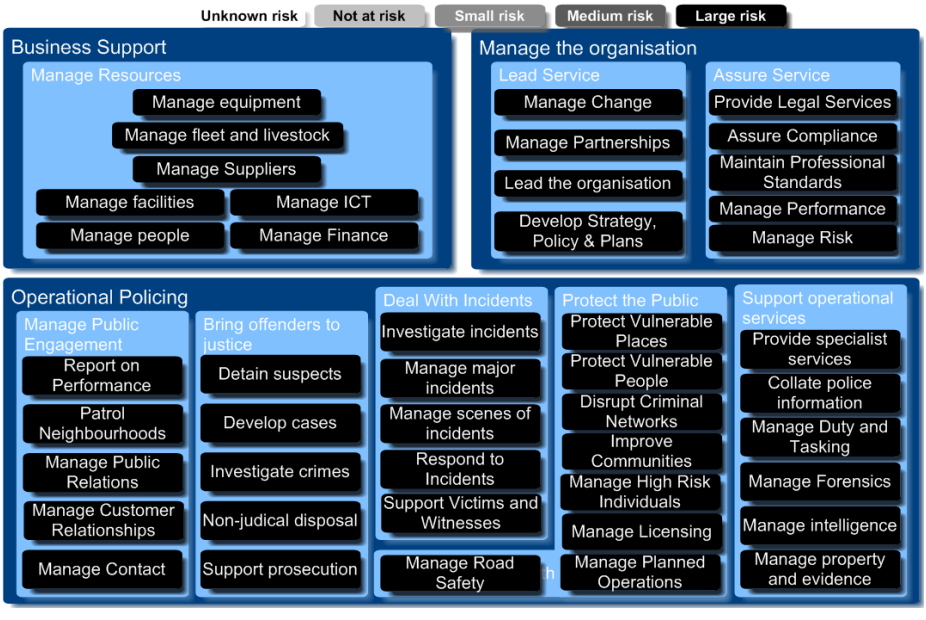


Fig. 13. Business Risk Heatmap (2)

This was caused by a "corporate application" – an application used, or at least available for use, by the whole of the organisation: in this case, a corporate gazetteer, considered to be a top-priority application. What the current model (i.e. the one used when these studies were carried out) does not take into account is the fact that an application may be essential to one business service but only useful to another (a concept that features in the mathematical treatment explored earlier in this paper, carried out after the studies had been completed).

To avoid the issue of a 'corporate application' causing the whole of a heatmap to snap to a single colour[1] in this fashion, for the third study we resorted to a direct mapping from applications to business services. The heatmap thus obtained is as follows:

[1] For the purposes of this paper, the colours originally used in the studies have been converted to grayscale; the original studies used red to denote large risk, green for no risk and other colours for intermediate levels of risk.

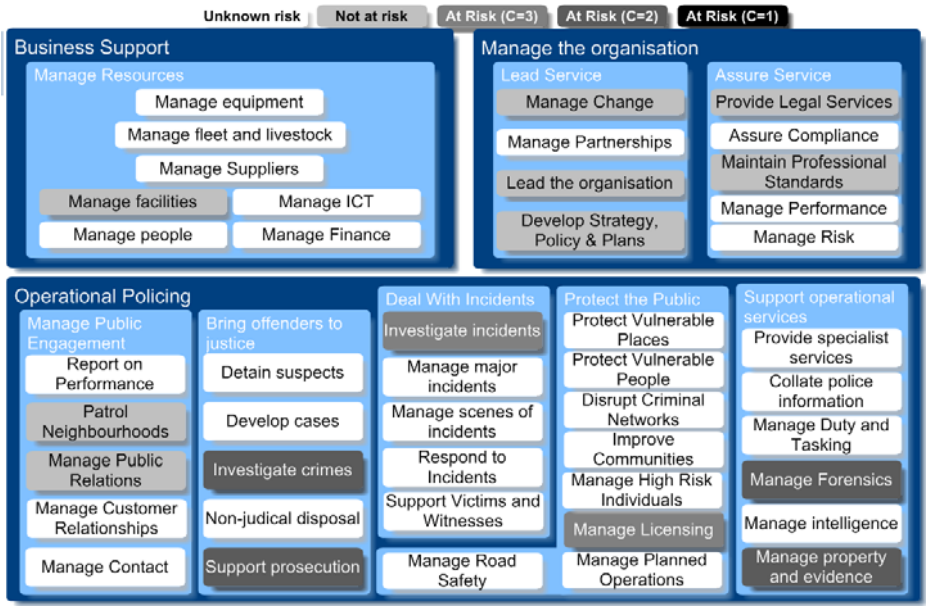


Fig. 14. Business Risk Heatmap (3)

The metamodel used in the third study was significantly enhanced from that used in previous studies, including the concepts of information (addressing Master Data Management) and also application tiers (components).

The 'white' or 'unknown' areas in the heatmap are as valuable as the 'shaded' or 'known' ones. This is because they highlight areas where the organisation does not have sufficient knowledge of their IT estate even to have a view as to the potential risk to their business.

The value of these models lies in their ability to convey a technical concept (out-of-date products) and the link through the Enterprise Architecture model to business stakeholders in terms that are easy to understand and completely non-technical. Thus, this is a useful tool in demonstrating the risk component of the business case for making the necessary changes to the IT portfolio, to remove this risk.

3 Lessons Learnt

3.1 Building the Model

In order to carry out this kind of analysis, two things are needed: firstly, the raw information, with consistent terminology and all the relationships between the various kinds of information; and secondly, some kind of toolset to enable the import, analysis and reporting of that information. For organisations with a relatively mature EA function, the first of these should not present a huge challenge, however in many

cases even getting the client to produce a single definitive list of applications is difficult. It is also difficult to see how this could be done without some kind of modelling and requirements capture tool. However, one approach we are exploring is the development of an ontology chart based on organisational semiotics principles.

The MEASUR [19] model approach offers a structured method to interview and model the ontology of an organisation and has been used in related work applying the techniques to enterprise architecture and consulting modelling frameworks [20]. Excel can handle two- or perhaps three-dimensional data with the help of pivot data; complex meta-models such as Figure 3 are probably beyond the ability of such tools to handle. A more robust technology is required, ideally layered over some kind of relational database to ensure the integrity of the data. Some candidate tools can be seen in [21].

3.2 What We Got Out of It

The teams involved in each of the projects felt that by the end, a good deal of evidence had been collected that gave a very strong business case to continue work with each respective client, to address the cost and risk issues identified so clearly by the work so far. In the second and third studies, the approach from the first study was readily re-usable, using the easy to understand heat-map, even though the meta-model was significantly different for subsequent studies. The resulting heat-map for the pair of police forces was all red (second study), due to a critical corporate application, used across the whole of the organisations that uses Oracle 8. This 'all-red' picture gave a very powerful and well-received message to the client about the urgency of the situation. The meta-model used for the second case study, built using lessons learnt from the first, omitted the infrastructure and technology catalogues, relating applications directly to underlying products. It also differed in that the linkage from applications to business services went via the organisation structure. This linkage was much simpler in the first study, going directly from applications to business services. Nevertheless, the heat-map was still able to be calculated in a similar fashion. The third study learnt from the second in that the linkage was made directly from applications to business services in order to avoid a problem with a corporate application presenting an unnecessarily pessimistic picture.

3.3 Illustration of Extending the Studies to Incorporate In-Context Application Criticality Values

The following discussion gives a brief example of how the above client data might be modified to use a more complex algorithm, described previously, that takes into account the correlation between applications and the services in which they are used.

For corporate applications (applications used in every business service), there is a likelihood that many such applications (for example, email, corporate intranet) are not absolutely key to many services, and as such may have a criticality correlation value of something low like 0.1. Given that the majority of the services will themselves have relatively low criticality values (let's assume an average of 0.2), then with a total

of 45 business services (in the policing sector), we would end up with something in the region of 0.1 x 0.2 x 45, which is close to 0.5.

By contrast, consider a specialised application such as command and control, which might support critical services (CS≈1) like 'respond to incidents' and perhaps two more. These kind of applications are likely to have high correlation values (very hard to run a police control room efficiently using bits of paper, although not totally impossible), and so the criticality is likely to be, say, 2 or 3 services, perhaps with a total CS value of 2, with applications that are very important to it (say, 0.8), yielding a criticality value of 1.6.

Thus, the combination of a highly critical service accompanied by an application that is highly critical to that service is more than capable (assuming we don't have too many business services defined!) of avoiding getting swamped, in terms of the magnitude of the result thus obtained, by a multitude of non-critical applications supporting non-critical services. However, the utility of this enhanced method would undoubtedly benefit from testing with a real client.

3.4 What the Clients Got Out of It

The main deliverable being sought by the clients, regarding obsolescence, was a view as to the motivation (business case) for making change. The use of the business heatmap, along with the TCO root cause analysis (shown above) and other financial information outside the scope of this paper, provided a clear business case at low cost. The approach and model are capable of being extended to accommodate increased technology and risk complexity if required.

4 Conclusions

Having used the approach successfully in three separate cases with very different meta-models, albeit only in a single industry sector, we have concluded that the approach is readily re-usable.

The library of product obsolescence data captured during the first engagement was very useful in terms of shortening the research required during the subsequent engagements to produce the obsolescence heat-maps.

5 Future Work

In retrospect, the terminology used for some of the artefacts in the meta-model need further work. In particular, the word 'technologies' is perhaps misleading in the way it is being used in this kind of analysis. Further research is required of existing architecture frameworks and methods to identify more commonly used and industry accepted terms that enable new clients to quickly come up to speed. As mentioned we are considering ontological analysis for specific industry terminologies to identify the relevant terms [22]. Also we intend to review and expand the obsolescence impact factor chain and investigate how this could be embedded in developing architecture

based consulting analysis frameworks, potentially using the Capgemini Integrated Architecture Framework or the BTS analysis framework [20] which has embedded structures for identifying the relationships between business services and IS services and their criticality.

In particular, more work is called for in terms of perhaps quantifying the extra factors in the obsolescence chain discussed earlier, and it would be helpful to test the enhanced method (mathematical treatment of contextual application criticalities) in real client scenarios.

Trademarks

References

1. Bulow, J.: An Economic Theory of Planned Obsolescence. The Quarterly Journal of Economics 101(4), 729–749 (1986)
2. Whelan, K.: Computers, Obsolescence, and Productivity. SSRN eLibrary (2000)
3. Solomon, R., Sandborn, P.A., Pecht, M.G.: Electronic part life cycle concepts and obsolescence forecasting. IEEE Transactions on Components and Packaging Technologies 23(4), 707–717 (2000)
4. Lemer, A.C.: Infrastructure Obsolescence and Design Service Life. Journal of Infrastructure Systems (December 1996)
5. Feldman, K., Sandborn, P.: Integrating technology obsolescence considerations into product design planning. In: Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2007 (DETC2007/DFMLC-35881) (2007)
6. Aversano, L., et al.: Supporting Decisions on the Adoption of Re-engineering Technologies. In: Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR 2004), Tampere, Finland (2004)
7. Boehm, B.: Managing software productivity and reuse. Computer 32(9), 111–113 (1999)
8. Sandborn, P.: Software Obsolescence - Complicating the Part and Technology Obsolescence Management Problem. IEEE Trans. on Components and Packaging Technologies 30(4), 886–888 (2007)
9. Ginn, D., Streibel, B., Varner, E.: The design for six sigma memory jogger: tools and methods for robust processes and products, 1st edn. Goal/QPC, Cop., Salem (2004)
10. Lagerstrom, R., et al.: A method for creating Enterprise Architecture metamodels - applied to systems modifiability analysis. International Journal of Computer Science and Applications 6(5), 98–120 (2009)
11. Strnadl, G.F.: Aligning Business and IT: The Process-Driven Architecture Model. In: The International Conference on Computer as a Tool, EUROCON 2005 (2005)
12. OpenGroup, The Open Group Architectural Framework (TOGAF 9) (2009)

13. Noran, O.: An analysis of the Zachman framework for enterprise architecture from the GERAM perspective. Annual Reviews in Control 27(2), 163–183 (2003)
14. NPIA. Police Activities Glossary (2011), http://pra.npia.police.uk/ (cited April 7, 2011)
15. Halliday, S., Badenhorst, K., von Solms, R.: A business approach to effective information technology risk analysis and management. Information Management & Computer Security 4(1), 19–31 (1996)
16. Wilkinson, L., Friendly, M.: The History of the Cluster Heat Map. The American Statistician 63(2), 179–184 (2009)
17. Miyake, M., Mune, Y., Himeno, K.: Strategic intellectual property portfolio management-Technology appraisal by using the technology heat map, in NRI Papers, Nomura Research Institute, p. 15 (2004)
18. Detre, J., et al.: Scorecarding and Heat Mapping: Tools and Concepts for Assessing Strategic Uncertainty. International Food and Agribusiness Management Review 9(1) (2006)
19. Stamper, R.: Social norms in requirements analysis: an outline of MEASUR. In: Requirements Engineering 1994, pp. 107–139. Academic Press Professional, Inc. (1994)
20. Liu, K., Sun, L., Jambari, D., Michell, V., Chong, S.: A Design of Business-Technology Alignment Consulting Framework. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 422–435. Springer, Heidelberg (2011)
21. Short, J. Wilson, C.: Gartner Assessment of Enterprise Architecture Tool Capabilities, Gartner (2011)
22. Guarino, N.: Formal Ontology and Information Systems. In: Proceedings of FOIS 1998, Trento, Italy. IOS Press, Amsterdam (1998)

A Method for Business Model Development

Lucas O. Meertens, Maria-Eugenia Iacob, and Lambert (Bart) J.M. Nieuwenhuis

University of Twente, PO Box 217, Enschede, The Netherlands
{l.o.meertens,m.e.iacob,l.j.m.nieuwenhuis}@utwente.nl

Abstract. Currently, business modelling is more an art, than a science, as no widely accepted method exist for the design and specification of business models. This could be an important reason why many IT innovation projects fail to be absorbed in a real life setting. We propose a structured method to create "as-is" business models in a repeatable manner. The method consists of the following steps: identify the involved roles, recognize relations among roles, specify the main activities, and quantify using realistic estimates of the model. The resulting business model reflects the current situation. This is the basis for further analysis of possible business cases, scenarios, and alternative innovations, which may enable successful projects to be implemented, instead of ending on a shelf after the pilot stage. We illustrate the proposed method by means of a case in the healthcare sector.

Keywords: Business modelling, modelling method, business models.

1 Introduction: Business Modelling Background

A business model is critical for any company, and especially for any e-business. Its importance has been recognized over the past few years by several authors that have created different business model frameworks aimed at identifying the main components of a business model (for example, Osterwalder [1]; for an overview, see Pateli & Giaglis [2], and Vermolen [3]). However, the state in which this field finds itself is one of "prescientific chaos" [4]: several competing schools of thought exist, and progress is limited because of a lack of cumulative progress. Because of this, no clear and unique semantics are agreed upon in the research related to business models. The very concept of "business model" is associated with many different definitions [3]. The components of such a business model differ significantly from one approach to another. Furthermore, to the best of our knowledge, no widely accepted methodological approaches exist in the literature for the design and specification of business models [3]. This is in contrast with well-established approaches, such as TOGAF [5], and Unified Process [6], which have emerged in the other, yet closely related, areas of enterprise architecture and information system design.

This lack of cohesion in the field clearly diminishes the added value of business models for organizations and makes business modelling an art, rather than a science. This, and not using business models altogether, are two of the most important reasons why many IT projects end after the pilot stage, and fail to be absorbed in a real life

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 113–129, 2012.

setting, which makes them unable to fulfil their apparent promise. Since the context of our research is the design and implementation of services in the healthcare sector, we particularly look at this issue in relation with healthcare IT projects. A majority of them fail in some sense, according to Kaplan and Harris-Salmone [7]. They recognize that, for systems to be successful, design methods must include organizational, behavioural, cognitive, and social factors. Also, a systematic review of cost effectiveness of telemedicine by Whitten et al. [8] concludes that "there is no good evidence that telemedicine is a cost effective means of delivering health care" (neither do they present evidence that it is not cost effective). While we do not go into any further detail whether or not telemedicine is cost effective, their review also shows that only a low ratio (55 out of 612) of studies present cost/benefit data. Even from this small amount, only a few did this according to the standards otherwise applied in medicine. This shows the lack of attention the financial aspect of innovations is getting.

In the case of telemedicine, previously published research by Broens et al. [9] indicates one of the reasons for the pilot-illness, namely that financial aspects and organizational aspects are considered only after the pilot phase.

While IT in healthcare is a special case, success of IT implementation projects in other sectors is not much higher. Several studies have reported failure rates between 40% and 84% [7] [10]. The CHAOS reports of the Standish Group are the most well-known ones these, especially as the report from 1994 reported the highest failure rates [11]. It reported that only 16.2% of projects were completed on time, on budget, and met user requirements, while 31.1% of projects failed outright.

This state of affairs motivates us to propose a method, which enables the development of business models in a structured and repeatable manner. Thus, the contribution of this paper is three-fold:

- A business model development method;
- A definition of the concept of business model and the identification of its core elements, captured by the deliverables of the method steps;
- An illustration of the method, by means of a case study from the healthcare domain.

The structure of the paper is as follows. Section 2 focuses on the discussion of the main concepts addressed in the paper, and positions our approach with respect to the existing design science and method engineering literature. In Section 3, we describe the steps of our business model development method. In Section 4, we demonstrate the method by means of a case study concerning the development of a quantitative business model of the elderly care in the Netherlands. Finally, we conclude our paper and give pointers to future work in Section 5.

2 Theoretical Background

A simple analysis of the two words “business model” already gives an idea of what a business model is about. On the one hand, there is “business”: the way a company does business or creates value. On the other hand, there is “model”: a conceptualization of something – in this case, of how a company does business.

We extend this common and simplistic interpretation of a business model as "the way a company earns money", into a broader and more general definition of the concept: "a simplified representation that accounts for the known and inferred properties of the business or industry as a whole, which may be used to study its characteristics further, for example, to support calculations, predictions, and business transformation."

The last part of the definition above, namely the indication of the possible uses of a business model is of particular importance in the context of this paper. The method we propose not only facilitates the development of such a design artefact – a business model – but also takes a business engineering perspective. Thus, its application will result in two (or more) business models: one that reflects the "as-is" situation of the business and one or more alternative "to-be" business models that represents possible modifications of the business as result of, for example, adoption of innovative technologies or more efficient business processes.

To the best of our knowledge, such a method does not exist yet for what we define as business models [3]. In the remainder of this section, we position our work in the context of design science, method engineering, and methodology-related contributions in the field of business modelling.

2.1 Design Science

A business modelling method can be seen as a design-science artefact. It is the process of creating a product, the business model. We use the seven guidelines of Hevner et al. [12] to frame how we use the methodology engineering approach from Kumar & Welke [13] to create our method.

The first guideline advises to design as an artefact. Design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation. As said, we produce a method.

The second guideline tackles relevance. The objective of design-science research is to develop technology-based solutions to important and relevant business problems. Viable business models lie at the heart of business problems. However, our solution is not yet technology-based. Partial automation of the method is left for future research.

The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods. We demonstrate the business modelling method using a case study.

Research contribution is the topic of the fourth guideline. Effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies. We provide a new artefact to use and study for the academic world. The methodology may be extended, improved, and specialized.

Guideline five expresses the scientific rigour: Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact. We aim to be rigorous through using the methodology engineering approach. Existing, proven methods are used as foundation and methods where applicable. Evaluation was handled in the third guideline.

The sixth guideline positions design as a search process. The search for an effective artefact requires using available means to reach desired ends while satisfying

laws in the problem environment. Whenever possible, we use available methods for each of the steps. Following the methodology engineering approach helps us to satisfy the laws for creating a new methodology.

The final guideline instructs us to communicate our research. Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences. This article is one of the outlets where we present our research.

2.2 Methodology Engineering

Methodologies serve as a guarantor to achieve a specific outcome. In our case, this outcome is a consistent and better-informed business model. We aim to understand (and improve) how business models are created. With this understanding, one can explain the way business models help solve problems. We provide a baseline methodology only, with a limited amount of concepts. Later, we can extend, improve, and tailor the methodology to specific situations or specific business model frameworks.

The business modelling method has both aspects from the methodology engineering viewpoint: representational and procedural [13]. The representational aspect explains what artefacts a business modeller looks at. The artefacts are the input and deliverables of steps in the method. The procedural aspect shows how these are created and used. This includes the activities in each step, tools or techniques, and the sequence of steps.

2.3 Business Modelling Related Work

Several contributions in the area of business modelling are related and relevant in the context of this research. Montilva and Barrios [14] recognize the idea that information system design should consider the enterprise context of these systems, and that it should be enhanced with business modelling elements. They propose three types of models, two of which we discuss as well, namely that of a business model (the "BMM product model"), and that of a process model that specifies the steps to be taken to produce the business model. However, a significant difference exists between these results and our research, caused by the very definition of the business model concept. Thus, Montilva and Barrios' business model concept is closer to that of an enterprise architecture model than to our understanding of the business model concept, both in terms of content and in level of detail. Montilva and Barrios' business model contains rather detailed specifications of elements such as goals, events, business rules and processes, business objects, and technologies, which are typically captured by enterprise modelling languages, such as ArchiMate [15]. Furthermore, the process model that Montilva and Barrios propose only focuses on the design of a business model with the sole purpose of serving as source of requirements for the future IS design.

Barrios and Nurcan [16] follow the same line of thinking in another paper, which focuses on the relationship between business models and enterprise information systems in a changing environment. Nevertheless, neither of the papers mentioned above addresses the issue of quantifying business models and using them to evaluate the business value of the future system by means of one or more business cases or cost/benefit analysis.

3 Defining the Business Modelling Method

We define five individual steps of business modelling, which the rest of this section elaborates. To describe each step, we use the following elements:

- inputs of the steps,
- activities to perform during the steps,
- possible techniques to use during the steps' activities, and
- deliverables resulting from the steps.

Each step in the proposed method requires specific methods, techniques, or tools that are suitable for realizing the deliverables. We will mention examples of those. However, others may also be useful and applicable, and it is not our aim to be exhaustive in this respect. Table 1 shows an overview of our method.

3.1 Create As-Is Model

As mentioned in the previous section, our business model development method takes a business engineering perspective. Thus, the first four steps of our method focus on creating a business model that reflects the current state of the business. Therefore, steps one through four create an as-is model.

Step 1: Identify Roles. Identifying the relevant parties (which we refer to as roles) involved in a business model should be done as systematically as possible. The aim is completeness in this case. The business modeller must carry out a stakeholder analysis, to identify all roles. The input to this step includes for example, documentation, domain literature, interviews, experience, and previous research. The output is a list of roles.

Table 1. Business Modelling Method.

Step	Inputs	Techniques or Tools	Deliverables
Identify Roles	Documentation, domain literature, interviews, experience, previous research	Stakeholder analysis [17]	Role list
Recognize Relations	Role list, Stakeholder map, value exchanges	e3-value [18]	Role-relation matrix
Specify Activities	Role-relation matrix, Role list, business process specifications	BPM methods, languages and tools	List of activities
Quantify Model	Process specifications, accounting systems and annual reports	Activity based costing	Total cost of the business "as-is"
Design Alternatives	As-is business model, Ideas for innovations and changes	Business modelling method (steps 1 to 4), Brainstorming	One or more alternative business models
Analyse Alternatives	Alternative business models	Sensitivity analysis, technology assessment, interpolation, best/worst case scenarios	Business case for each alternative

For an example stakeholder analysis method, we refer to Pouloudi & Whitley [17]. They provide an interpretive research method for stakeholder analysis aimed at inter-organizational systems, such as most systems where business modelling is useful. The method consists of the following steps:

1. Identify obvious groups of stakeholders.
2. Contact representatives from these groups.
3. (In-depth) interview them.
4. Revise stakeholder map.
5. Repeat steps two to four, until...

Pouloudi and Whitley do not list the fifth step, but mention that stakeholder analysis is a cumulative and iterative approach. This may cause the number of stakeholders to grow exponentially, and the question remains when to stop. Lack of resources may be the reason to stop the iterative process at some point. Closure would be good, but seems hard to achieve when the model is more complex. Probably, the modeller has to make an arbitrary decision. Nevertheless, one should choose stop criteria (a quantifiable measure of the stakeholder's relevance for the respective business model and a threshold for the measure) before starting the process [19].

"Revising the stakeholder map" (step four) could use extra explanation, which can be found in the description of the case Pouloudi and Whitley use to explain the method. The stakeholders gathered from interviews can be complemented with information found in the literature. The business modeller then refines the list of stakeholders by focussing, aggregating, and categorizing.

Step 2: Recognize Relations. The second step of our method aims to discover the relations among roles. The nature of these relations may vary substantially, but it always involves some interaction between the two roles, and may assume some exchange of value of some kind. Much of the work and results from the previous step can be reused for this as input. In theory, all roles could have relations with all other roles. However, in practice, most roles only have relations with a limited number of other roles. Usually, these relations are captured in a stakeholder map, which often follows a hub-and-spoke pattern, as the focus is on one of the roles. This pattern may be inherent to the approach used, for example if the scope is defined as a maximum distance from the focal role.

To specify all relations, we suggest the use of a role-relation matrix with all roles on both axes as technique. Of this matrix, the cells point out all possible relations among the roles. Each of the cells could hold one or more relations between two roles. Assuming that roles have a limited number of relations, the role-relation matrix will be partially empty. However, one can question for each empty cell whether a relation is missing or not.

Cells above and below the diagonal can represent the directional character of relations. Usually, relations have a providing and consuming part. The providing part goes in the upper half of the matrix, and the consuming part in the bottom half. This especially helps with constructions that are more complex, such as loops including more than two roles.

The output of this step is a set of relations.

Step 3: Specify Activities. For a first qualitative specification of the business model, the next step is to determine the main activities. Relations alone are not sufficient: the qualitative model consists of these main business activities (business processes) too. These activities originate from the relations identified in the previous step. Each of the relations in the role-relation matrix consists of at least one interaction between two roles, requiring activities by both roles. Besides work and results from the previous steps, existing process descriptions can be valuable input. Techniques from business process management may be used.

The output from these first three steps is a first qualitative business model, including roles, relations, and activities. It reveals what must happen for the business to function properly.

Step 4: Quantify the Model. Quantifying the business model helps us to see what is happening in more detail and compare innovations to the current situation. To turn the qualitative model into a quantitative model, numbers are needed. The numbers are cost and volume of activities (how often they occur). Together, these numbers form a complete view of the costs captured by the business model.

Several sources for costs and volumes are possible, such as accessing accounting systems or (annual) reports. The resulting quantitative business model shows the as-is situation.

3.2 Develop the To-Be Model

The as-is model, created in previous steps, is suitable for analysis of the current state only. However, from the as-is model, it is possible to derive alternatives. Such alternatives can be created to assess how reorganisations, innovations, or other changes influence the business. These are the to-be models.

Step 5: Design Alternatives. From here on, we aim to capture a future state of the business in a business model. To make predictions, the model may need further instantiations. Each instantiation is an alternative development that may happen (to-be). Using techniques such as brainstorming and generating scenarios, business modellers create alternative, qualitative, future business models. These alternatives are used to make predictions. Usually, such alternatives are built around a (technical) innovation. This may include allocating specific roles to various stakeholders. A base alternative, which only continues an existing trend without interventions, may help comparing the innovations. Next to the business model, ideas for innovations serve as input. The resulting alternative business models show future (to-be) possibilities.

Step 6: Analyse Alternatives. The final step for a business modeller is to analyse the alternative business models. Besides the qualitative business models, several sources of input are possible to quantify the alternatives. Applicable techniques include sensitivity analysis, technology assessment, interpolation, and using best/worst case scenarios. Each alternative can be tested against several scenarios, in which factors change that are not controllable. We can use the models to predict the impact. This step and the previous one can be repeated several times to achieve the best results. The final output is a business case (including expected loss or profit) for each alternative.

4 The U*Care Case: Demonstrating the Business Modelling Method

U*Care is a project aimed at developing an integrated (software) service platform for elderly care [20]. Due to the aging population and subsequently increasing costs, elderly care - and healthcare in general - is one of the areas where governments fund research. However, many projects never get further than pilot testing. Even if the pilot is successful, the report often ends on a shelf. By applying the business modelling method, we aim at avoiding this, and putting the U*Care platform into practice. Specifically, our goal is to show how the technological innovations built in the U*Care platform influence the business model for elderly care.

4.1 Identify Roles

The first step of the stakeholder analysis, leads to the identification of several groups of obvious stakeholders. The groups include all the project partners, as their participation in the project indicates their stake. Another group includes the main users of the platform: the clients and employees of the elderly care centre.

After identifying the obvious stakeholders, we contacted and interviewed representatives from all the project partners and several people in the care centre. These interviews did not explicitly focus on stakeholder analysis, but served as a general step in requirements engineering. Table 2 displays a partial list of identified stakeholders after steps two and three of Pouloudi and Whitley's method for stakeholder analysis have been performed [17].

Table 2. Partial list of stakeholders after step three of Pouloudi and Whitley's method for stakeholder analysis [17]

Clients	Care (& wellness) providers
Volunteer aid	Hospitals
Nurses	Elderly care centres
Doctors	Psychiatric healthcare
Administrative employees	Homecare
General practitioners	Technology providers
Federal government	User organizations
Local government	Insurance companies

The fourth step includes a search for stakeholders in the literature. Besides identifying the extra stakeholders, the literature mentioned the important issue that some actors in the list are individual players, while other actors are organizations or other forms of aggregations (groups). Consequently, overlap can occur in the list of actors.

The final action of the first iteration is not a trivial one. Refining the stakeholder list requires interpretation from the researcher. Different stakeholder theories (for example, from E. J. Emanuel & L. L. Emanuel [21], J. Robertson & S. Robertson [22], and Wolper [23]) act as tools to minimize subjectivity.

The long list of identified stakeholders is not practical to continue with and has much overlap. Therefore, we grouped the stakeholders into a limited set of roles, shown in Table 3. This set of high-level roles is an interpretive choice. The small set helps to keep the rest of case clear instead of overcrowded. The larger set is kept in mind for the to-be situation to find potential "snail darters": stakeholders with only a small chance of upsetting a plan for the worse, but with huge results if they do [24]. The small set of stakeholders was subject to prioritization based on Mitchell et al [25]. While the prioritization is subjective, it shows that all roles in the list are important.

4.2 Recognize Relations

The current situation consists of five categories of interacting roles. Table 3 shows them on both axes. The cells show relations between the roles. While the care provider has relations with all the other roles, it is not a clear hub-and-spoke pattern. Several of the other roles have relations outside the care provider.

Table 3. Role-relation matrix

Consumer Provider	Care consumers	Care providers	Technology providers	Government	Insurers
Care consumers	X	Pay for care		Pay for AWBZ Pay for WMO	Pay for insurance
Care providers	Provide ZVW care Provide WMO care Provide AWBZ care	X	Pay for (use of) technology or service	Provide care to citizens	Provide care to insured
Technology providers		Provide technology or service	X		
Government	Provide AWBZ insurance Provide WMO insurance	Pay for WMO care to citizens		X	Pay for AWBZ care to citizens
Insurers	Provide insurance Refund AWBZ and ZVW care			Ensure AWBZ care for citizens	X

The relations show that a main goal of the business is to provide care to the care consumer. The insurers and government handle much of the payment. Other (regulating) roles of the government remain out of scope, as the case is complex enough as it is.

The insurers handle most of the payments. The patient has to pay the care provider after receiving care. The patient can then declare the costs to the insurance company, which refunds the patient. The patient pays a premium to the insurance company. According to the Dutch Healthcare Insurance Act (Zorgverzekeringswet, ZVW), every citizen has to have basic care insurance (ZVW). For "uninsurable care" (including most home healthcare, similar to USA Medicare), the Dutch government set up a social insurance fund, termed General Exceptional Medical Expenses Act (Algemene Wet Bijzondere Ziektekosten, AWBZ). All employees and their employers contribute towards this fund. The AWBZ is similar to the regular insurance companies, except for collecting the premium. The premium is paid through taxation by the government, which outsources most of the further actions to insurers. A similar system is set up for wellness homecare, such as cleaning. This is the Social Support Act (Wet Maatschappelijke Ondersteuning, WMO). In contrast to the AWBZ, the government takes care of all WMO actions itself, through its municipalities.

Several issues exist, which we do not handle in detail here. For example, it is inherent to insurance that not all people who pay premium are also (currently) care consumers.

4.3 Specify Activities

Most of the relations between the roles in Table 3 are described using verbs. This signals that they are (part of) behaviour. Any relation not beginning with a verb is a candidate for rephrasing or being split into smaller parts.

Besides the relations, we focus on AWBZ to identify the main activities of the care providers. "Providing care" has four top-level activities: personal care, nursing, guidance/assistance, and accommodation. Each of these activities consists of many detailed activities. Table 5 provides an example of a further refinement and specification of the personal care activities. We obtained these activities from documents made available by the government for reimbursement purposes [26]. Fig. 1 shows the qualitative model for AWBZ care in the Netherlands, as described above.

4.4 Quantify the Model

As we are interested in the actual healthcare and not so much in the insurance business, we zoom into the care provided by the care providers to the care consumers, as Fig. 1 highlights. We scope this further to the AWBZ care that a home for the elderly provides. This is mainly personal care, and accommodation. Accommodation has two components, similar to those you would find in a hotel: food-related and living quarters. Personal care consists of more activities, which Table 5 shows.

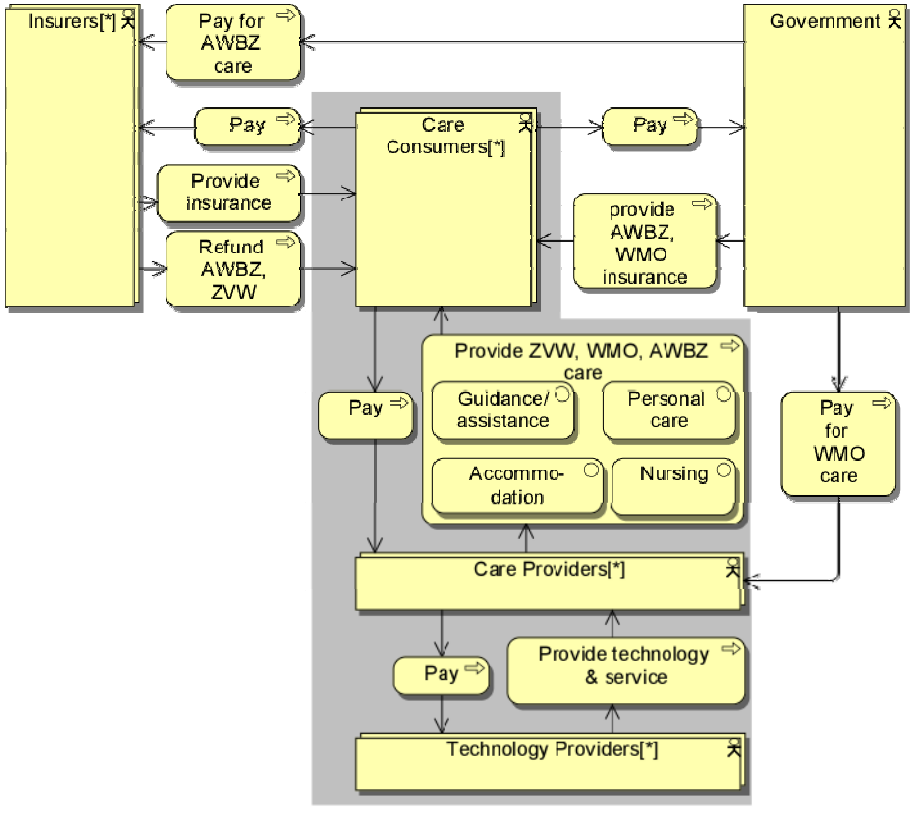


Fig. 1. A model for healthcare in the Netherlands, including actors, relations, and activities

For the U*Care case, we use the numbers of one department of a home for the elderly from Orbis Medisch en Zorgconcern. For confidentiality reasons, we have manipulated the numbers. However, they still represent such a department. The department houses 63 people, with an average care indication of "four" for the AWBZ care. This means that the care provider gets approximately €100 per person per day. Therefore, the annual revenues of this department are approximately € 2.3 million (= 63 people x 365 days x €100).

The total costs, which can be related directly to this department, are approximately €1.8 million. This includes personal care, accommodation (both food-related and living quarters), as well as management. Table 4 shows these costs per component. The difference, of €0.5 million between the revenues and the total costs, comes from costs that cannot be related directly to the department. It includes costs incurred by the overarching organization, such as cost of capital and other overhead costs.

Indications of volume (times a day, and minutes spend), which the government uses for reimbursement purposes, provide a further step to quantifying the model. With this information, we can assign costs to each of these activities, which the

Table 4. Costs for a department in a home for the elderly (x €1,000)

Food-related	250
Living quarters	510
Management	100
Personal care	910
Total costs	1,770

caregivers perform. We focus on this, as it is the largest part of the costs (95% of the personal care costs arise from human resources), and this is the area on which innovations can have the greatest influence.

The caregivers in this department combined work for approximately 30 FTE (Full Time Equivalent, which is 36 hours per week in the Dutch healthcare). So a total of approximately 154 hours can be spent per day (= 30 FTE x 36 hours / 7 days per week). The last column in Table 5 is the amount of hours caregivers spend on each activity per day. Adding up the hours in this last column leads to approximately 145 hours. This leaves 9 hours per day for things not included in Table 5, such as administration and changing rooms.

Table 5. Personal care activities according to [26], extended with the amount of elderly in need of each activity, leads to the total amount of time spend on each activity daily

Activity	Actions	Time in minutes	Frequency per day	Elderly in need	Hours per day
Washing	Whole body	20	1x	21	7
	Parts of body	10	1x	21	3,5
Dressing	(Un)dress completely	15	2x	21	10,5
	Undress partially	10	1x	21	3.5
	Dress partially	10	1x	21	3.5
	Put on compression stockings	10	1x	21	3.5
	Take off compression stockings	7	1x	21	2.45
Getting in and out of bed	Help getting out of bed	10	1x	21	3.5
	Help getting into bed	10	1x	21	3.5
	Help with afternoon rest (for example, get onto the couch)	10	1x	21	3.5
	Help with afternoon rest (for example, get off the couch)	10	1x	21	3.5

Table 5. (*Continued*)

Eating and drinking	Help with eating cold meals (excluding drinking)	10	2x	10	3.33
	Help with eating warm meal (excluding drinking)	15	1x	21	5.25
	Help with drinking	10	6x	10	10
Change position sitting/lying		20	3x	6	6
Going to toilet or changing incontinence material		15	4x	10	10
Support excretion	Stoma	10-20	4x	10	10
	Catheter	10	4x	10	6.7
	CAPD/CCPD	30	4x	3	6
Tube feeding		20	2x	3	2
Medication	Present medicine	5	3x	48	12
	Administer medicine (oral)	5	3x	15	3.75
	Apply medical patch	5	2x	10	1.7
	Administer eye, ear, or nose drops. Administer medicine (non-oral)	10	2x	6	2
	Nebulise medicine	20	1x	3	1
Personal care for teeth, hair, nails, and skin	Care for teeth	5	2x	21	3.5
	Care for hair	5	1x	31	2.6
	Care for nails	5	1x (per week)	31	0.4
	Inspect skin	10	1x	10	1.7
	Care for skin	10	1x	6	1
Attaching and removing prosthetic limb	Attaching limb	15	1x	6	1.5
	Removing limb	15	1x	6	1.5
Teaching and supervising personal care activities	Teaching the above activities	30 per week	As above	10	0.7
	Supervise to ensure quality of self-care	30 per week	Spread over week	63	4.5

An average hour of care costs approximately €15 (=€910,000 x 95% / 30 FTE / 52 weeks per year / 36 hours per week). Together with the hours spent per day, we can now calculate the costs of each activity. For example, the most expensive activity is presenting medicines. A total of 12 hours is spend on this each day, therefore the costs per day are approximately €180 (= 12 hours x €15 per hour). This is approximately 8% of the total costs of personal care each day.

The same calculations can be made for the other activities and for the other costs of the home for the elderly, such as accommodation. It results in a complete quantitative business model of the current situation. For this case, we do not go into further detail.

4.5 Design Alternatives

To come up with alternatives, we conducted interviews, held workshops, and constructed several scenarios for the U*Care project. Each of the scenarios features one or more innovations for a home for the elderly [27][28].

For this case, we focus on the scenario that includes the introduction of an electronic medicine dispenser. This innovative dispenser can present pre-packaged medicine to elderly on the right time and in the right dose. Using sound and light signals, it attracts the attention of the elderly. Besides this, it registers when the medicine is taken. Optionally, it can notify a caregiver if the medicine is not taken on time.

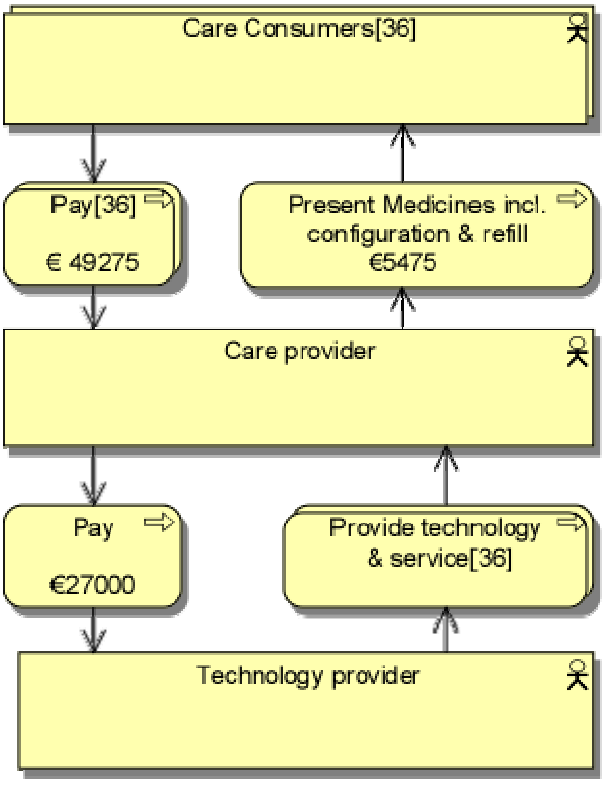


Fig. 2. Alternative model for presenting medicine with an electronic medicine dispenser

The expectation of the scenario designers is that the introduction of the electronic medicine dispenser will decrease the costs of care. To achieve this, it reduces the time caregivers spend on presenting medicine, as elderly can get the medicines from the dispenser instead.

Of course, the introduction of the dispenser also brings along new costs. The dispenser has to be obtained, maintained, configured, and refilled. The dispenser will be leased from and maintained by a technology provider. The caregivers get to do extra activities in the form of configuring and refilling the dispenser for the elderly. Fig. 2 shows a model of the new situation.

4.6 Analyse Alternatives

From a management perspective, the dispenser should only be introduced if the benefits exceed the costs. For this case, we only include monetary benefits and costs, as these can be quantified in a straightforward fashion. In contrast, potential benefits, such as quality of care, are hard to quantify. We assume the introduction of the electronic medicine dispenser does not result in a change in the quality of care.

The benefits arise from a reduction of the time spend on presenting medicine to the elderly. It is estimated that 36 (75%) of the elderly that currently get their medicines presented can make use of the dispenser. Therefore, the dispenser reduces the time spend on presenting medicine by 9 hours per day (= 12 hours x 75%). This amounts to €135 saved each day (= 9 hours x €15 per hour) or €49,275 per year.

The costs of the innovation come in two forms. First, a fee paid to the technology provider for leasing and maintaining the dispenser. Second, time spend by the caregivers on the extra activities of configuring and refilling the dispenser. The fee for the technology provider is €750 per year per dispenser. This amounts to €27,000 per year (=€750 x 36 dispensers). The time spend on the extra activities is estimated to be on average about an hour per day. We base this on configuring and refilling of the dispenser once a week, which takes twice the time that presenting the medicine normally takes. Therefore, the costs of this time is approximately €5,475 (=€15 x 365 days). The total costs for introducing the electronic medicine dispenser are approximately €32,475 (=€27,000 + €5,475).

As the (monetary) costs of the introduction (€32,475) are less than the benefits (€49,275), the business case seems to be positive (by €16,800). Therefore, we can recommend introducing the electronic medicine dispenser.

4.7 Evaluating the Case

The presented case shows how the business modelling method results in a quantitative business model of the current situation, as well as the target situation.

The case first provides a high-level model of the elderly healthcare business in the Netherlands. To assess the particular innovation, we went into depth on only a small area, a single department of a home for the elderly. For other innovations, maintaining a higher-level view may be necessary.

The case is simplified and it also contains estimations. For example, we simplified the case by leaving out actors, such as the pharmacist, and start-up costs, such as training costs for the dispenser. Estimations include numbers that were not available, such as the amount of elderly that need an activity, or exact times spend on them.

5 Conclusions: A Future for Business Modelling

This paper makes three contributions. Primarily, we created a business model development method. Secondarily, we defined the concept of business model and identified its core elements, captured by the deliverables of the method steps. Finally, we demonstrated the method by means of a case study from the healthcare domain.

The business modelling method provides a way to create business models. Innovators can apply the steps to create business cases for their ideas systematically. This helps them to show the viability and get things implemented.

We provide a new design-science artefact to use and study for the academic world. As business modelling has several goals, conducting only the first few steps may be enough. For example, if your goal is to achieve insight in the current state only, the last two steps are not useful.

The business modelling method may be extended. Situational method engineering seems suitable for this [29]. For example, for information system development, it is interesting to research if steps towards enterprise architecture can be made from business models. This can be seen as a higher-level form of, or preceding step to, the BMM proposed by Montilva and Barrios [14]. On the other side, a step could be added before identifying roles. Other domains may require different improvements.

In addition, the steps in the method can be further specified. The steps can be detailed further. One of the ways to do this is to tailor the techniques at each of the steps of this method. In the future, new tools and techniques may help provide partial automation.

Acknowledgements. This work is part of the IOP GenCom U-CARE project, which the Dutch Ministry of Economic Affairs sponsors under contract IGC0816. Especially, we thank Orbis Medisch en Zorgconcern for their contributions.

References

1. Osterwalder, A.: The Business Model Ontology - a proposition in a design science approach (2004), http://www.hec.unil.ch/aosterwa/PhD/Osterwalder_PhD_BM_Ontology.pdf
2. Pateli, A.G., Giaglis, G.M.: A research framework for analysing eBusiness models. Eur. J. Inf. Syst. 13, 302–314 (2004)
3. Vermolen, R.: Reflecting on IS Business Model Research: Current Gaps and Future Directions. In: Proceedings of the 13th Twente Student Conference on IT, University of Twente, Enschede, Netherlands (2010)
4. Kuhn, T.S.: The Structure of Scientific Revolutions. University of Chicago Press (1970)
5. The Open Group: TOGAF Version 9. Van Haren Publishing (2009)
6. Scott, K.: The unified process explained (2002)
7. Kaplan, B., Harris-Salamone, K.D.: Health IT Success and Failure: Recommendations from Literature and an AMIA Workshop. Journal of the American Medical Informatics Association 16, 291–299 (2009)
8. Whitten, P.S., Mair, F.S., Haycox, A., May, C.R., Williams, T.L., Hellmich, S.: Systematic review of cost effectiveness studies of telemedicine interventions. British Medical Journal 324, 1434 (2002)

9. Broens, T.H.F., Huis in't Veld, R.M.H.A., Vollenbroek-Hutten, M.M.R., Hermens, H.J., van Halteren, A.T., Nieuwenhuis, L.J.M.: Determinants of successful telemedicine implementations: a literature study. J. Telemed. Telecare 13, 303–309 (2007)
10. McManus, J., Wood-Harper, T.: Understanding the sources of information systems project failure. Management Services 51, 38–43 (2007)
11. Johnson, J.: The CHAOS Report. The Standish Group International, Inc. (1994)
12. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28, 75–105 (2004)
13. Kumar, K., Welke, R.J.: Methodology Engineering: a proposal for situation-specific methodology construction. In: Challenges and Strategies for Research in Systems Development, pp. 257–269. Wiley, Chichester (1992)
14. Montilva, J.C., Barrios, J.A.: BMM: A Business Modeling Method for Information Systems Development. CLEI Electronic Journal 7 (2004)
15. Iacob, M., Jonkers, H., Lankhorst, M., Proper, H.: ArchiMate 1.0 Specification. Van Haren Publishing, Zaltbommel (2009)
16. Barrios, J., Nurcan, S.: Model Driven Architectures for Enterprise Information Systems. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 3–19. Springer, Heidelberg (2004)
17. Pouloudi, A., Whitley, E.A.: Stakeholder identification in inter-organizational systems: gaining insights for drug use management systems. European Journal of Information Systems 6, 1–14 (1997)
18. Gordijn, J.: Value-based Requirements Engineering: Exploring Innovative e-Commerce Ideas (2002)
19. Pouloudi, A.: Stakeholder Analysis for Interorganisational Information Systems in Healthcare (1998)
20. U*Care Project: U*Care, http://www.utwente.nl/ewi/ucare/
21. Emanuel, E.J., Emanuel, L.L.: What is Accountability in Health Care? Ann. Intern. Med. 124, 229–239 (1996)
22. Robertson, J., Robertson, S.: Volere: Requirements specification template. Technical Report Edition 6.1, Atlantic Systems Guild (2000)
23. Wolper, L.F.: Health care administration: planning, implementing, and managing organized delivery systems. Jones & Bartlett Publishers (2004)
24. Mason, R.O., Mitroff, I.I.: Challenging strategic planning assumptions: theory, cases, and techniques. Wiley (1981)
25. Mitchell, R.K., Agle, B.R., Wood, D.J.: Toward a theory of stakeholder identification and salience: Defining the principle of who and what really counts. Academy of Management Review, 853–886 (1997)
26. Ministry of Health, Welfare and Sport: Beleidsregels indicatiestelling AWBZ 2009 (2008)
27. van't Klooster, J.W., Beijnum, B.J.V., Pawar, P., Sikkel, K., Meertens, L., Hermens, H.: Virtual communities for elderly healthcare: user-based requirements elicitation. International Journal of Networking and Virtual Organisations 9, 214 (2011)
28. Zarifi Eslami, M., Zarghami, A., Van Sinderen, M.: Service Tailoring: Towards Personalized homecare Systems. In: Proceedings of the 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC). SciTePress, Athens (2010)
29. Henderson-Sellers, B., Ralyté, J.: Situational Method Engineering: State-of-the-Art Review. Journal of Universal Computer Science 16, 424–478 (2010)

Administrations as Instruments for Dealing with Organizational Complexity

Coen Suurmond

RBK Group, Keulenstraat 18,
7418 ET Deventer
csuurmond@rbk.nl

Abstract. The concept of administration can contribute much to both the quality and the maintainability of information systems. Although the practical connotation of the concept is traditionally primarily in the financial area, an established definition of the administration gives a good foundation for the application of the concept in other areas. It has every possibility to comply with both organisational responsibilities and software structures. Decentralised, small, local administrations, located close to the business processes, can contribute to both process quality and to the maintainability of information systems. It supports lower and middle management in their jobs, and reverses the trend to centralisation and to the erosion of responsibilities of staff carrying out the work.

Keywords: Business Modelling, Software Architecture, Administrations.

1 Introduction

The development, implementation and operational use of information systems within enterprises face certain persistent problems. These problems arise from misconceptions about the nature and role of information systems within an enterprise. I refer here firstly to the misconception that the role of the computer-based information systems is to cover all of the information supply within an enterprise. And I refer here secondly to the misconception that it is in the nature of an information system to represent reality "as is".

These misconceptions shape the analysis and the design of IT systems mostly as background assumptions, and go undiscussed. The nature and role of information outside of the IT systems are often underexposed in IT projects, and such information is considered peripheral to these systems. Once an IT system is in use, the representations in the system are often considered as leading. E.g., a sales order is no longer seen as an agreement between a buyer and a seller; a sales order is defined by its representation in the system.

The result of these misconceptions is that IT systems often do not function properly in practice. Unforeseen limitations are imposed on business processes and on the commercial and logistic opportunities of the enterprise, as a consequence of the use of IT systems. Limitations that might cost money internally and might cost trading opportunities externally.

B. Shishkov (Ed.): BMSD 2011, LNBIP 109, pp. 130–146, 2012.

This can produce various reactions that aim to absorb these unforeseen negative consequences. First, the system can be expanded, to remove, to evade or to circumvent the limitations. Second, practice might create its own way of working and its own auxiliary systems for certain processes, alongside the IT systems that were meant to do this. Third, practice might learn to live with the limitations.

At the basis of any solution for the problems at issue must be the awareness that it is the task of computer-based information systems to facilitate business processes. These systems thus have an instrumental nature and a serving role. They cannot be the Archimedean point from which business processes are controlled. Neither should these systems determine how the business must be done.

To meet these demands IT systems should be open in three meanings of the word: (1) they have to be able to collaborate with other systems outside of their own functional areas, (2) they have to be able to collaborate with other systems within their own functional areas, when these systems have a lower-level execution task or a higher-level controlling task, and (3) they have to be able to deal with events within their functional area that (temporarily) circumvent the system.

If we want to achieve this openness, then I think it is useful, if not unavoidable, to exploit the concept of "administration" as a core concept in the design of information systems. The administration concept analyses the administrative processes in separation of other processes, its operations are based on clearly defined services and it implies clearly demarcated organisational responsibilities. Although the concept originates in the financial domain, it is very well applicable outside of the financial field; indeed, the concept should hold for all information that is used communally in an organisation.

2 Route Planning as a Metaphor

2.1 A Car Navigation System

A car navigation system is an example of an information system that gives its user proper and up-to-date information, leaving him the freedom to take his own decisions. Such a system indicates the route to be followed, taking into account the current position, possible routes and traffic intensity on the different routes. The driver is informed about the current possibilities, and can combine this information with his own experience and wishes. He can deviate from the route whenever he wants to do so. The navigation system always works from the actual situation; what happened before and what motivates the choices of the driver are completely irrelevant. The information system is supportive, based on the actual situation, independent of earlier plans or events, and does not force decisions upon the user. The only thing that seems to be lacking is the presentation of alternative routes simultaneously.

2.2 Offline Route Planning

To describe a route by step-by-step directions is quite a different story. See for example the directions given by the British AA for the trip from Corrie on Arran to Bridgend on Islay:

Direction	Miles
Start out on unnamed road	0.00
Turn left onto the A481	0.07
Turn right	8.78
Continue by vehicle ferry (Lochranza – Claonaig)	8.81
Turn right	13.47
Turn left onto the B8001	13.57
Turn right onto the A83 (signposted Glasgow)	18.63
Turn left (signposted Islay ferry)	19.00
Continue by vehicle ferry (Port Askaig – Kennacraig)	19.25
Turn left onto the A846	48.82
Arrive on the A846	56.66

Section time 6:16, Total time 6:16

Such a description loses a lot of its value when the driver is forced off the indicated route. Apart from that, these specific directions do not take into account that the driver should take either the ferry to Port Askaig (as described), or the ferry to Port Ellen (not described), depending on the time. And checking the miles travelled against the odometer of the car won't work either, because of the two ferries involved.

The information system is directive, based on a prefixed sequence of events, dependent on earlier plans or events, and forces decisions upon the user.

2.3 Organisation and Information

Starreveld wrote in the early sixties about an information supply for making judgements ex ante (for decisions in business processes) and ex post (for the accountability of business processes) [1]. He observes that on lower levels of the organisation there is primarily a need for all kinds of operational data that is required for the correct and efficient execution of the tasks, accompanied by a need for the information necessary to make professional assessments. According to Starreveld, the main questions for determining the information needs for an employee are:

- Which decisions are to be made in the involved business processes, and what information does he need in order to make these decisions?
- What operational data does he need in order to prepare and fulfil his tasks?

A few years later Robert Anthony writes: "Several authors state that the aim of control is to assure that the results of operations conform as closely as possible to plans. We emphasise that such a concept of control is basically inconsistent with the concept used in this study. To the extent that middle management can make decisions

that are better than those implied in the plans, top management wishes it to do so. And the middle management can in fact make better decisions under certain circumstances; to deny this possibility is implicitly to assume that top management is either clairvoyant, or omniscient, or both, which is not so." [2].

Especially in production organisations there is an increasing tendency to emphasise a cycle of planning and control where plans are made higher-up in the organisation and executed lower-down. Information from lower-down to higher-up is reduced to reporting back on predefined tasks. This tendency is reflected in ERP systems with their modules for planning and control. The shop floor is given production orders to be executed, and can only report back on those production orders. Registration of unplanned activities is difficult or even impossible, even if some problem on the shop floor made those unplanned activities necessary. Professional judgement in the fulfilment of jobs is eroded.

Current ERP systems are comparable to the directions given by the AA. The individual steps are determined in detail in advance of the execution of a process, and the employees involved in the execution of the process get pushed information about the steps to be taken. This way of working is sensitive to disturbances by deviations, the responsibilities are centralised and employees who perform the tasks are regarded as just cogs in the machine. The approach described by Starreveld and Anthony assumes a situation in which employees at every hierarchical level of the organisation have tasks and responsibilities which are not frustrated by centralised and bureaucratic information systems. They can get the information they need whenever they request it, and they can make the decisions within their domain. As a model this is more comparable to a driver assisted by a navigation system.

3 Example: Planning of Packaging of Perishable Consumer Products

Consider a department for the packaging / labelling of perishable consumer products. The products are collected several times each day for transport to a central warehouse where the customer orders are picked and distributed. The lead time of the products from ordering to delivery is 1 to 3 days, depending on the commercial agreements about guaranteed shelf life of the products. The planning / execution cycle consists of the following steps:

- Finished-Product-Planning: based on the current warehouse stocks and sales forecasts the central organisation creates a volume plan for the packaging department (volume per product per production date). This volume plan is created in advance of the production week and then adjusted based on the available data in advance of the production day.
- Site Planning: The planner on the location converts the volume plan into production orders for each line, in processing sequence. The criteria used are product characteristics and delivery schedules.

- Production Management: The department supervisor prepares the processing of the production orders and checks whether the orders can be started and processed, and he releases the orders for processing.
- Foreman: On each line the foreman sees to the processing itself and the data collection involved.

The responsibilities in this planning / execution cycle are as follows:

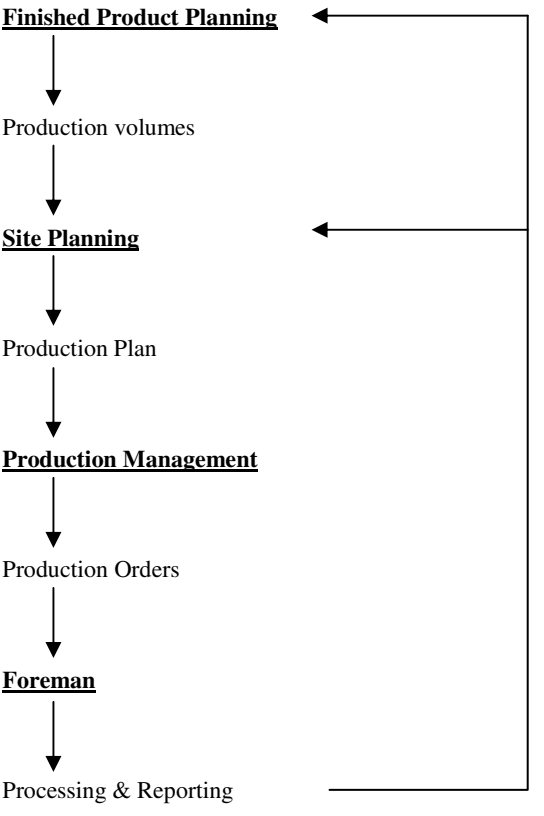
- The planning in the central organisation is responsible for the service level and for the loss incurred by products exceeding their sell-by date.
- The planning on the location is responsible for the realisation of the required daily volume, the timely departure of the shipments with the right contents and minimising the loss of run time of the lines because of change overs and cleaning.
- The department supervisor is responsible for the efficiency (use of resources per unit of product) and process quality.
- The foreman is responsible for the efficiency on his line and for the processing of the packaging orders.

In a normal course of business, with only insignificant deviations within the actual sales developments and the actual progress of the business processes, the picture outlined above will give an adequate view of the flow of the business processes. Each link in the planning chain translates the information of the preceding step into more detailed information for the next step and everybody sees to his own job. Processes and responsibilities are clearly demarcated.

However, significant disturbances will occur regularly. Examples are deviations in sales, rejection of products, machine malfunctions, late supply of raw materials, shortage of required resources. The question is how these larger and smaller disturbances are dealt with. There is a trend of increasingly putting it on the plate of the planners, partly because every deviation is viewed as a planning problem, partly because adjusted or new orders are necessarily created via the planning system in order to be able to produce and report.

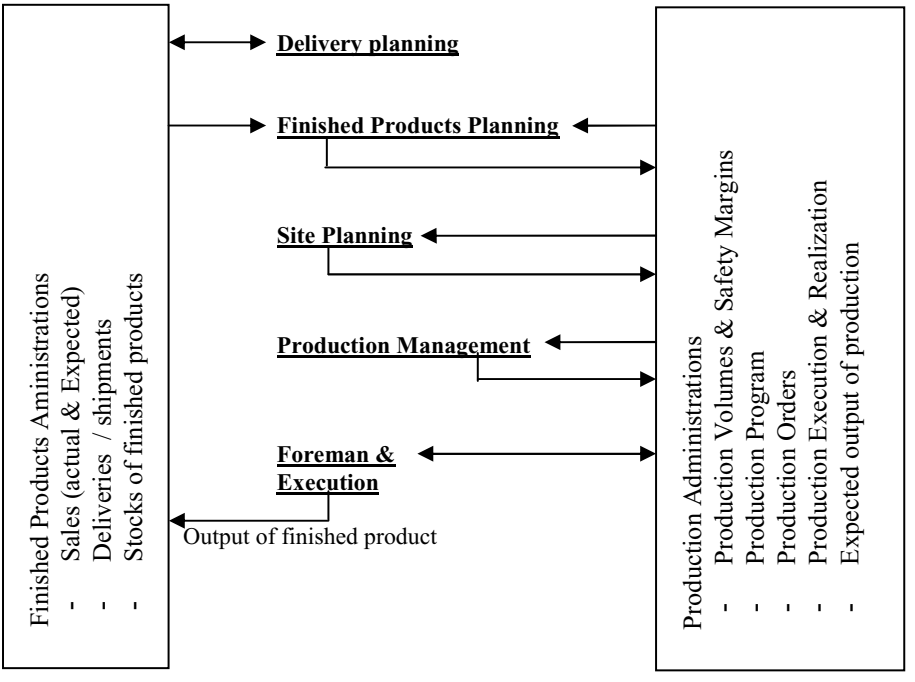
The consequences of this trend of putting every adjustment to the planner are twofold: erosion of the responsibilities of the department supervisor, who is increasingly less able to make decisions, and an increasing need for detailed real-time information from the primary processes to the planner.

The planner controls the production processes in detail and the department supervisors and forepersons become his puppets. At the same time department supervisors and forepersons lack the information to make adjustment for deviations, an area that has traditionally been their strength and part of their added value in the organisation.



Compare this to an alternative way of organising the information, a way that is aimed at allowing the employees to fulfil their responsibilities. The volume planner indicates what the target volume per product is, what the required minimum is to avoid endangering the service level and what the allowed maximum is to avoid the risk of destruction. The target volume is determined through the normal planning procedure and adjusted only in case of significant deviations. The minimum and maximum are updated regularly based on actual production and sales numbers. In his line planning the site planner can already take the given tolerances into account and determine the length of his production runs on that basis. Here, the site planner states with which shipment the different products should be transported to the warehouse at the latest (default is the last lorry, earlier only if the stock position of the warehouse demands so). At the moment of preparation for the processing of the production orders the department supervisor will base himself on the production orders of the location planner, but he will also take the tolerances of the volume planning into account. In this way he can optimise the efficiency and the use of raw materials, while limiting the number of batches of raw materials (because of traceability issues). And during the production day he can accommodate for small deviations in production speed by shortening or lengthening some production runs. The information regarding his decisions about the use of raw materials and the consequences for the length of the

production runs and the expected output of finished product is available to both the location planner and to the volume planner. In case of larger deviations it is firstly the department supervisor that sees whether he can accommodate them given the production orders, tolerances and departure times of the lorries. It becomes an issue for the site and volume planners only if this is not possible. Judging the costs of being unable to deliver the product is the responsibility of the volume planner, judging the extra production and transport costs is the responsibility of the site planner.



The finished product planning determines the volumes to be produced, and regularly updates the safety margins. The planning regularly receives updated information about the expected output of production, as well as updated information about the demand for finished products.

The site planning, production management and foreman are free to fine tune the production quantities, taking into account the safety margins and the current situation in production. They are required to use information from outside the computer systems (by email or by phone) that is relevant to their decisions. They are responsible for up to date feedback to the production administration about the expected output of production.

In the first planning model every layer in the organisation does its job in isolation and communicates only the standardised output to the next layer. But in the second model every layer makes up to date information available to allow every other layer to assess the situation and its margins; thus allowing every layer to observe its responsibilities.

4 Criteria

4.1 Introduction

The quality of an information system within an organisation can be derived from the contribution of the information system to the quality of the business processes (in terms of effectiveness and efficiency) and the manageability of the information system itself in terms of speed and costs of problem solving and costs of adjusting to requested changes. The contribution to the quality of the business processes presupposes that the employees and systems in the business processes get their information correctly, in time and complete. This requires that the collecting, checking, processing and storing of the information also happen correctly, in time and at acceptable costs. The contribution to the process quality is directly dependent on the data quality. Because of this, the responsibility for the data quality should lie with people who are knowledgeable about the processes that produce the information.

4.2 Quality of Business Processes

The quality of the business processes should be determined in terms of effectiveness and efficiency. Effectiveness is the degree to which the intended result is achieved and efficiency is the consumption of resources to achieve the intended result. For the efficiency of the business processes it is necessary that the information is supplied in the right form so that the user does not have to waste time by (avoidable) additional processing of information. The maxims for having a conversation formulated by the British language philosopher Grice present a good practical guideline for the way in which information should be presented to the user [3]:

- Maxim of Quality — Be truthful and reliable
- Maxim of Quantity — Be informative
- Maxim of Relation — Be relevant
- Maxim of Manner — Be brief and orderly

For the effectiveness of the business processes it is firstly necessary for the information to be correct, timely and complete, and secondly that the user knows what the presented information represents and thus is able to relate this information to information from other sources. Detailed real time information from production processes and from sales processes have little value when decisions about the volume and the sequence of production orders are to be made. Here, what is needed is aggregated information about any significant deviations that put the service level at risk. In order to interpret the aggregated data, definitions are important. Suppose a production company has divided the day into a number of production blocks of 2 hours. If a production block represents a fixed amount of time then the volume within the block will vary. However, if a production block represents a normative production volume then the finishing time will be variable while the volumes are fixed. In the first case the production planning will have to relocate volumes between production blocks regularly during a production day, but will have little reassigning of people to do.

In the second case this is reversed: little relocation of volumes, but regularly reassigning people. In this second case it is easier to relate production volumes to distribution and sales.

4.3 Maintainability

The maintainability of an information system can be determined by the speed and costs of solving operational problems in the information supply, as well as the speed and costs of adjusting the information supply to changing circumstances.

In the management and maintenance of information systems we are dealing with several aspects: technical management, application management and a form of management that I will here call the supervision of information. The technical management can be more or less centralised as with other forms of technical management; criterion here is the availability of the infrastructure and as such this is not relevant here. Application management, which includes the configuration of the systems, should be located close to the processes. The configuration of the systems should be tuned to the operational processes. An application manager must be able to translate the needs of the operational processes to the capabilities of the application and to formulate the requirements for any changes to the applications. This requires proximity to the process. Sometimes the distance between application management and operational process can be overcome through a middle layer of key-users.

The third category of management, the supervision of information, is primarily operational management like the other two forms, coupled with a responsibility for continuous process improvement (in terms of effectiveness and efficiency). The operational responsibility of information supervision is ensuring the continuous availability of timely, correct and complete information from the operational processes in the right form. Attention please: the responsibility is not for the availability of information to the processes, but for the availability of information from the processes. It is a responsibility to ensure that other processes have the right information at their disposal, and to inform other processes of possible problems in the information available.

The role of information supervisor is in general not a full time job, but a role that should be assigned to an employee within the department. The final responsibility always lies with the department supervisor and not somewhere far away in a central body of the organisation. As a production supervisor bears responsibility for the quality of his physical product, so does the information supervisor bear the responsibility for the quality of information.

Organisations often do have an information manager function. This is a professional whose job it is to link the business processes and the automated systems. His role is mainly in setting up and configuring the systems and not so much in the operational responsibility for the information within the systems. When adjustments are made to the information supply the insights of the information manager about the structures and about the general process knowledge will be combined with the insight and experience of the information supervisors to arrive at good solutions that work well in practice.

5 Concepts

5.1 The Concept of Administration

Starreveld arrives in his original work in the early 60s about information processes in organisations at this definition of administration: "The systematic collection, recording, processing and supplying of information for purposes of the managing and functioning of a household and for purposes of the accountability thereof" [1]. The American Accounting Association defines accounting as follows: "the process of identifying, measuring and communicating economic information to permit informed judgements and decisions by users of the information" [4]. The definition of the AICPA in 1961: "Accounting is the art of recording, classifying and summarising in a significant manner and in terms of money, transactions and events which are, in part at least, of a financial character, and interpreting the result thereof" [5]. The value of these definitions is that they sketch a clear and normative view of the role of administration in the business processes.

From the FASB, part 2: "The purpose of this Statement is to examine the characteristics that make accounting information useful. …All financial reporting is concerned in varying degrees with decision making (though decision makers also use information obtained from other sources). …The usefulness of information must be evaluated in relation to the purposes to be served, and the objectives of financial reporting are focused on the use of accounting information in decision making ... Even objectives that are oriented more towards stewardship are concerned with decisions. Stewardship deals with the efficiency, effectiveness, and integrity of the steward. To say that stewardship reporting is an aspect of accounting's decision making role is simply to say that its purpose is to guide actions that may need to be taken in relation to the steward or in relation to the activity that is being monitored." [6].

It is clear that the term administration traditionally has a strong connection to the financial management and to internal and external financial reporting. At the same time it becomes clear from the definition of Starreveld that this financial aspect is not an essential element of the definitions given. The essential elements of the definitions are: (1) the systematic nature of the collecting, processing and making available of data, (2) the separation of the collecting, processing and making available of data on the one side and the interpretation of the data on the other side, (3) the use for operational decisions, and (4) the use for internal and external reporting, analysis and accountability.

Regarding the systematic nature I would like to draw attention to an element that is only found explicitly in the definition of the AICPA: classifying and summarising. Boisot has made an analysis of the nature of information and he defines three aspects of information [7]. One aspect concerns the extent to which the information has been codified, a second aspect concerns the degree of abstraction, and the third aspect concerns the degree of diffusion. In an administration system information will have to be codified, the users request information at different levels of abstraction and an administration system is a mechanism for diffusion both within and outside of the organisation.

5.2 Administration and Organisation

Mintzberg defines the structure of an organisation as "the sum total of the ways in which divides its labour into distinct tasks and then achieves coordination among

them" [8]. He distinguishes five different kinds of coordination mechanisms, namely: (1) mutual adjustment, (2) direct supervision, (3) standardisation of work, (4) standardisation of outputs, and (5) standardisation of skills. From an organisational point of view an administration department can be considered both as a specialisation of labour and as a coordination mechanism.

Administration departments are often specialised in the processing of either financial data, employee data or production data. The justification of such departments in an organisation is in the required competencies of the employees, as well as in the sensitivity of data. The departments have to meet a variety of requirements from a variety of stakeholders. They have to meet both the internal requirements of the operational and management processes, and they have to meet the requirements of external stakeholders. The head of the department is responsible for the quality of the information supplied.

Administrative processes might also perform coordinating roles in an organisation. When the meaning and use of administrative data are clearly defined, and when the administration processes are specified as well, this will in effect be a coordination mechanism by standardisation of output and by standardisation of skills. The required skills are related to the ways and means of collecting and verifying the data in the operational processes, and (equally important) to the interpretation of the information supplied. The creation of big administrative departments may lead to autonomous administrative worlds with their own language, separated from the operational and managerial business processes. The advantages of specialisation are partially counteracted by the disadvantages of compartmentalisation, especially when the administrative officers are insufficiently aware of the supporting role of their jobs. As an old joke says: the difference between a terrorist and an accountant is that the terrorist might be susceptible to reason.

Directing information flows through an administration department or through an administration system also implies formalising the language used. The language used within an organisation is a mixture of everyday speech, jargon, and forms of more or less formalised language. Different departments can have different interpretations of the same concept. If an order is delivered to a customer in a lorry with a trailer, there is one shipment. If two lorries arrive together for the same delivery, does this involve one or two deliveries, and one or two shipments? If the customer demands that an order is delivered as a whole, then the answer is clear for the commercial department: in both cases one shipment is made. For the freight documents it is clear as well: in the first case one shipment is made with the accompanying freight documents and in the second case two shipments are made, each with their own documents. For the receiving DC of the customer it is also clear: in both cases two deliveries are made that must be docked and unloaded separately.

In practice people within an organisation use different kinds of sign systems simultaneously. The everyday use of language is fairly free and unconfined, even within the context of an organisation. For commercial and financial transactions the language used is more formalised and ultimately is grounded in written law and case law. The use of automated systems is another form of formalisation of sign use. Part of it is formatting (type and size of the fields), part of it is predefined categorisation

(tick the correct box) and part of it is capturing some part of the organisational reality in IT artefacts.

To implement an administration system implies the advance creation of conventions governing the terminology, relations and meaning. In this sense it is a formalisation of the use of language. It can also be considered as a coordination mechanism in the organisation: standardisation of meaning.

The concept of administration in an organisation can be summarised as a way of organising information about a group of organisational phenomena directly linked to the business processes involved. Each administration has a durable kernel determined by the logic of the processes involved, in combination with a more pliable shell of interfaces for categorisation of inputs and interpretation of outputs. Small, dedicated and decentralised administrations, directly linked to the business processes, could prove to be a highly flexible contribution to reliable and maintainable information systems.

5.3 Administration and Software Engineering

Like organisations, software engineering has its methods for managing complexity. The multi-tier model, the client/server model and the service oriented architecture model are three examples of the principle of 'separation of concerns'. Earlier forms are the concept of structured programming, employing units (Pascal) or modules (Modulo) and later on object-oriented programming.

The mentioned mechanisms in software engineering are each directly concerned with the structure of the software as such. They do not or only tangentially concern themselves with how the software is used. In the last few decennia the discussion is increasingly about architecture. At first it was about the architecture of software, then about the architecture of information systems and finally about the architecture of the enterprise.

Within the software engineering as such Taylor states “By architecture we mean the set of principle design decisions made about a system; it is a characterisation of the essence and essentials of the application” [9]. The architecture consists of a number of semi-autonomous parts and the connections between them (static structure). Further there are the specific communication mechanisms between the parts (dynamic structure). Both for the static and for the dynamic structure the architect makes use of a repertoire of standard patterns. This way of working has first been charted by the architect Christopher Alexander [10] and later on has spread widely within software engineering.

However, that similar mechanisms for managing complexity (either called organisation or architecture) have been developed both in organisations and in software engineering does not mean that the mechanisms of both worlds should be considered equal. In the case of software we are dealing with a strictly formal and determined system, whereas in the case of an organisation we are dealing with a system that is essentially open and social.

The Reference Model for Open Distributed Processing (RM-ODP) describes a model for the collaboration of interconnected autonomous, heterogeneous information processing systems. ”The objective of ODP standardization is the development of

standards that allow the benefits of distributing information processing services to be realized in an environment of heterogeneous IT resources and multiple organizational domains. These standards address constraints on system specification and the provision of a system infrastructure that accommodate difficulties inherent in the design and programming of distributed systems." [11]. The concept of an administrative subsystem should qualify as a fine example of the heterogeneous IT resources mentioned above. At the same time we must realise that RM-ODP offers a technical solution for automated information processing. It does not address questions about the human part of information processing, or the organisational questions about tasks and responsibilities.

The modern developments in software engineering, coupled with architectural ideas such as those expressed in RM-ODP are a solid basis for representing administrative subsystems in software. Such a separated administrative subsystem is accessed exclusively through well-defined interfaces, which are clear from a software perspective and the separate terms of which can be mapped clearly to definitions and meaning in the business processes. Such separated administrative subsystems can meet the demands of openness in structure and in management mentioned earlier. Ideally, the administration system records just what actually is the case, regardless of any plans or intentions. When someone takes stock from the warehouse, this should be registered. The IT system should not forbid registration because of some rule or constraint in the software. It happened and it is relevant to the representation of the stock in the IT system, so it must be recorded in the administrative subsystem. Either the physical removal is prohibited, or it must be captured.

In the same vein, employees and systems should be able to retrieve the information they need for their tasks (within the boundaries of authorisation) from the administrative subsystems involved and make their relevant facts available to the administrative subsystems. These processes should be based on a pull model for information retrieved and a push model for the information produced, all according to the organisational tasks and responsibilities.

The alignment of organisational structure and software architecture can be a great help in improving both data quality and maintainability. Decentralised, local responsibility for the information in administrative subsystems can contribute to data quality and the semi-autonomy of the subsystems in the software can contribute to maintainability. The separation of process logic in the software functionality and of meaning in the software interfaces (both to humans and to other parts of the software) should improve the value of the information for the different kinds of use in the organisation.

5.4 The REA Model

The REA Model, for which the foundations were laid around 1980, results from the administrative practice and it is therefore interesting to compare this model with the model presented here. The original model has been described by McCarthy in his article: "The REA Accounting Model, a Generalized Framework for Accounting Systems in a Shared Data Environment" [12]. A later development has been described

in the book "Accounting, Information Technology and Business Solutions" by Hollander, Denna and Cherrington [13]. Hruby has recently used the model as a basis for his "Model-Driven Design Using Business Patterns" [14].

The acronym REA (REAL with Hollander) has been derived from the basic concepts of Resource-Event-Agent (and Location with Hollander). The basic idea of McCarthy is "This paper proposes a generalized framework designed to be used in a shared data environment where both accountants and non-accountants are interested in maintaining information about the same set of phenomena.". The basis for the model is in the concepts of economic resources, economic events, and economic agents. Resources are nearly equal to assets, events here deal by definition with mutations of resources, and agents are the participants in the events and those who bear the responsibility for the events. Positive in this approach is that the economic functioning is used as a basis in the analysis of an enterprise and that the responsibility of an agent is named explicitly.

Hollander et al. [12] develop a concept of financial administration in their work which is based on a real-time event-driven architecture for the Business. This architecture falls into category 9 (Semantic Theory of the Firm) of a categorisation of information systems by McCarthy, David and Summer which is characterised as follows: "These systems enable designers to explicitly represent an enterprise's full economic plan for creating customer value, including full information disclosure about all of the resources, agents, and locations of all events of interest. These systems, if implemented completely, could enable organizations to trace 100% of their economic processes from product inception to the final sale to the customer."

To this architecture the following attributes are then ascribed (page 112):

- The architecture is based on business events (business activities) rather than information customer views
- The architecture supports business process simplification and change
- The architecture integrates all business data
- The architecture integrates information processes and real-time controls.

In this approach many valuable elements can be found, especially decoupling the processing of information from the specific views of certain users and attempting to fit of the administration to the nature of the business processes. Further, the model of financial administration lets events be recorded and has events as the basis for balance mutations. Because of this everyone who needs to know about the state of affairs can retrieve the required information from the administration. Hollander acknowledges that the events must be recorded immediately so that the administration can present an up-to-date view of the state of affairs.

In some essential respects, however, the model of Hollander is lacking. The model attempts to cover every resource, event and agent that is relevant to the business, but then focuses on one aspect in "business value" and thus purely on the financial administration. From the generalised concept of administration this has two shortcomings. Firstly, there are essential events which do not imply a balance mutation. Think of a promise to a customer to deliver by a set date. This promise should be present in the sales administration, but the REA model does not accommodate this. Secondly, to other administrations it is not the financial aspect that

is central as they deal with material events and balances. The administration of raw materials, work in progress and finished products deals with mutations and balances of amounts and weights. The administration of a technical service is mainly concerned with things like the operating hours of the installations and the history of past malfunctions. Of course, they also have financial aspects, but while service levels and operational reliability can be expressed in financial terms, they cannot be measured financially. Discussions about intangible assets in accounting literature indicate that valuation bases are not always clear cut and sometimes even arbitrary. This is at odds with the criterion in the model presented here that an administration should show things as they are and should as far as possible be free of subjective elements.

A second criticism is the encompassing nature ("The architecture integrates all business data") under the heading "Semantic Theory of the Firm". When companywide unity of meaning is an illusion and when sites and departments each use their own language (which of course also must include common elements), having one central administration presents a big risk (the language of money is more universal however and that is what Hollander focuses on). While the later work of Hruby [14] tries to deal with commitments, it is even a more rigid attempt to model based design. Conceptually it is weak, for example when it states that “The Sale event means transfer of ownership of a product from the enterprise to the customer”. In my view, that is a delivery. A sale event would be a mutually binding commitment between supplier and customer to deliver some product or service, at some time, for some price.

To conclude: although the model presents valuable insights in the nature of an administration, its focus on financial aspects, its pretension of having an encompassing nature and its disregard of meaning issues prevent the model from providing the good reference for administrations that is searched for here.

6 Key Issues in the Deisgn of Administrative Systems

6.1 Organisational Issues

Each administrative process has to be clearly grounded in the organisation. It should be clear who is responsible for the data. Administration processes should be located as closely as possible to the operational processes involved, to ensure short communication lines. Those responsible for the administration processes should have a clear understanding of both the operational processes and of the administration processes so that they are able to solve any problems occurring in the collecting, processing or interpreting of data. They should actively monitor the use of the administration processes and indicate when they should be adjusted because of changing practices. This final point specifically concerns tracking change of meaning, either abrupt or slowly evolving. Consider for example the concept "customer" when an organisation first encounters the difference between the entity that places an order, the entity to which they should be delivered and the entity that pays for the delivered goods.

6.2 Modelling Issues

The domain of each administrative subsystem is clearly defined, together with the meaning of the main concepts. The rules for determining the identity of the individual administrative entities are explicit. The way in which both systems and human users refer to the separate administrative entities is defined and tested for practical applicability. Categories and range of values of the attributes of the entities are defined in advance.

The domain is determined by the question of what is it concerned with and of what information is requested. Essentially this is the same question as the one asked about objects in OO-thinking. An object is an identifiable unit with its own identity. However, objects are no "ready-mades". They need to be carefully defined. See the problem above of what constitutes one shipment.

Another issue is how involved parties can identify the administrative entities. Systems use unique references, which must connect uniquely to physical or conventional reality. This can happen by physical identification such as barcodes or chips. It can also happen by conventional identification such as GTIN numbers for products or GLN numbers for addresses and locations, managed by the international organisation. Human users have to know what they are dealing with as well. Either the references used are fit for both machine and human recognition, or different references are used for machines and for human users.

6.3 Technical Issues

The administration processes can be supported by one or multiple IT subsystems with the responsibility for their proper functioning lying with those responsible for the involved administration processes (and not with a central IT department somewhere far away in the organisation). Besides IT systems the administration processes can be supported by locally developed solutions, in Excel for example, and by systematic storage of forms, receipts, and lists. What medium is used to store data is less important; that the data are collected and made available according to the agreed procedures and in a correct, timely and complete manner is essential.

The interfaces of the IT subsystems supporting the administration processes are explicit and complete. There is no tacit meaning and there are no hidden side effects. From the defined interfaces and from the defined technical implementation of the IT subsystem its behaviour can be fully explained. The technical interfaces of the subsystem can be directly mapped to the organisational aspects of the administration processes. The subsystems can deal with the key characteristics as mentioned in RM-ODP 10746-1, paragraph 6.1: remoteness, concurrency, lack of global state, partial failures, asynchrony, heterogeneity, autonomy, evolution and mobility.

7 Conclusion

In the introduction two misconceptions were identified as a source of many problems in the use of IT systems, namely the idea that an information system would coincide with the IT systems used and the idea that the information in IT systems would represent the reality "as is". The need for open systems to facilitate business processes instead of systems designed from closed and encompassing models was discussed.

It was then analysed that the administration concept as defined by Starreveld should provide a good basis for open systems. Well-designed administrative subsystems that provide their services to the organisation and to applications independently of one another should make the systems more manageable. A condition for this is that the independent administrative subsystems are well embedded into the organisation, as closely as possible to the operational processes. Modern software architectures like RM-ODP should provide a good technical basis for this.

The design of an administrative subsystem for a specified area starts with abstracting and modelling it. This should be accompanied by a thorough analysis of the process logic of the area, to arrive at an adequate and practical choice of entities and references. This should also be accompanied by a certain formalisation and standardisation of the use of language.

When the people involved are familiar with the ins and outs of the chosen model they will be able to work well with the model in the interaction with the administrative subsystem while retaining the freedom of interpretation of data from the subsystem because they know what is not represented and because they are able to combine the data with data from other sources autonomously.

In conclusion: adequate administration systems function semi real time, provide crucial services to the business processes, and are driven by employees that have a thorough practical knowledge of these processes.

References

1. Starreveld, R.W.: Leer van de administratieve organisatie. N. Samsom NV, Alphen aan den Rijn, 2nd edn. (1963)
2. Anthony, R.: Planning and Control Systems. Graduate School of Business Administration Harvard University, Boston (1965)
3. Grice, P.: Logic and Conversation. In: Grice, P. (ed.) Studies in the Way of Words, pp. 22–40. Harvard University Press, Cambridge (1991)
4. Tutor2U, http://tutor2u.net/business/accounts/intro_accounting.htm
5. Glautier, M.W.E., Underdown, B.: Accounting, Theory and Practice, 7th edn. FT Prentice Hall, Harlow (2001)
6. FASB, http://www.fasb.org/home
7. Boisot, M.: Knowledge Assets. Oxford University Press, Oxford (1998)
8. Mintzberg, H.: The Structuring of Organizations. Prentice Hall Inc., Englewood Cliffs (1979)
9. Taylor, R.N., Medvidovic, N., Dashovy, E.M.: Software Architecture. John Wiley and Sons, Hoboken (2010)
10. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Center for Environmental Structure, Berkely (1977)
11. RM ODP, Information technology – Open Distributed Processing-Reference Model: Overview. ISO/TEC 10746-1. ISO/TEC, Genève (1998)
12. McCarthy, W.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review, 554–578 (1982)
13. Hollander, A.S., Denna, E., Cherrington, O.J.: Accounting Information Technology and Business Solutions. Irwin McGraw-Hill, Boston (1999)
14. Hruby, P.: Model-Driven Design Using Business Patterns. Springer, Berlin (2006)

Author Index