Enabling Information Interoperability through Multi-domain Modeling

Georg Grossmann, Andreas Jordan, Rishi Muruganandha, Matt Selway, and Markus Stumptner

University of South Australia, Adelaide, Australia
{first.lastname}@unisa.edu.au

Abstract. Flexible integration of information systems with heterogeneous data structures and interfaces has been an important IT research goal for decades. It is a fundamental requirement for enterprise transformation that the business knowledge captured in form of data and business processes can be integrated and adapted within and across enterprise boundaries. In this paper we present results of a model-driven interoperability approach in the asset management domain. The approach builds on multi-domain modeling principles and has been applied in three large use cases over the last 5 years. We show how information interoperability and enterprise transformation can benefit from multi-domain modeling and how it fits together with a design science approach.

Keywords: Domain modeling languages, model-driven engineering, meta modeling, information transformation, interoperability.

1 Introduction

The lack of semantic interoperability between information systems remains a big challenge in computer science and costs industry billions of dollars each year. In 2002, the National Institute of Standards and Technology estimated that the costs for inadequate interoperability in the US Capital Facilities Industry to be USD 15.8 billion per year [10]. This number was considered conservative by a report on interoperability in the construction industry in 2007 [25] and a recent Fiatech report on advancing interoperability [9].

Semantic interoperability is often a fundamental requirement to accomplish an enterprise transformation successfully. Many of drivers for enterprise transformation described by Harmsen et al. [13] such as mergers, acquisitions, introduction of novel technologies, new business models or compliance to corporate rules and policies require that existing information systems are interoperable at the time of the transformation and remain interoperable in the future. As pointed out in one use case in [13], lack of integration between the IT departments was a root cause of the lack of synergy between different business units.

Despite considerable progress made in the past, information integration remains challenging. Although first interoperability efforts were codified in the 1980s with standards such as EDIFACT or STEP APIs, the modeling of application data in terms of record structures and relational database schemas remained

F. Harmsen and H.A. Proper (Eds.): PRET 2013, LNBIP 151, pp. 16–33, 2013.

"too flexible" [2]. On the one hand, such flexibility increased the modeling power and incorporated more semantics (object or ontology-based methods) but on the other, it resulted in higher complexity and an increased likelihood that the data structures of independently developed systems would be mutually inconsistent and non-interoperable. The problem of low-level encoding and simple heterogeneous data types on the syntax level has mainly been solved with the acceptance of XML as the standard interchange format within and across enterprise boundaries. Data and application integration is therefore an ever-present issue and capabilities in this space are forcefully advertised by major information architecture vendors. To be strict, much of the emphasis lies on data transformation and interchange rather than wholesale migration. Systems such as IBM Websphere or SAP Netweaver present an XML-syntax, SOA based interface to the world, but the structure and meaning of data still needs to be adjusted by developers ensuring that the right data structures are exchanged and transformed at the right time [2].

We propose a multi-domain modeling approach to address some of the problems in semantic interoperability. In the context of three industry use cases we apply a meta modeling language to specify a set of executable domain specific languages (DSLs); one language for each data structure and interface that is required to be connected to the enterprise landscape. In a following step the developed domain models are integrated in two ways: (1) Behavior integration is achieved by composing some of their elements in another domain language that orchestrates the execution, and (2) static data and interface integration is achieved through bi-directional model transformation which is specified by another domain specific language within the same framework. By using the same meta modeling language for all DSLs, we achieve an integration on the language level that simplifies the integration of models while maintaining the benefits of domain-specific modeling. Specifically, this served the purpose of simplifying the semantic integration and addressed particular requirements which are explained in more detail in the next section.

The remaining paper is structured according to the STARR template: Section 2 describes the initial *situation* of the use cases which comes from the asset management sector, Section 3 provides an overview of the *tasks* including the requirements and goals, Section 4 covers the model-driven *approach* we took, Section 5 discusses the positive *results* of the approach, and Section 6 includes some *reflections* on the use case.

2 Interoperability in the Asset Management Sector

Gregory and Matthew [11,19] highlighted that the integration and data management of information systems are among the key challenges in Engineering Asset Management (EAM). Starting from the organization, planning, and controlling the acquisition of assets to the use, monitoring, maintenance, and disposal of physical assets, EAM incorporates multiple disciplines to manage the whole lifecycle of physical assets representing a unique interoperability challenge. In order

to achieve an integrated EAM solution, information systems from different areas such as risk management, budget and costing estimation, condition monitoring, human resources, or facility management need to be integrated. In this section we discuss different aspects of integration that are relevant in the EAM domain. First we discuss Enterprise Application Integration (EAI) and Business-to-Business (B2) integration and then two dimensions of integration, horizontal and vertical integration. Lastly, we introduce three use cases to which we have applied our approach.

2.1 EAI and B2B Integration

The integration of software applications can be classified according to the enterprise boundaries: Enterprise Application Integration (EAI) and business-to-business integration (B2B integration). EAI is concerned with the integration of software applications within an enterprise and B2B integration is concerned with the exchange of electronic documents between organizations. Both share some commonalities [22,6,7]:

- Business processes are used for modeling the sequence of activity execution.
- Routing rules are applied for defining the data exchange between two systems.
- System interfaces provide the basis for data exchange.

However, EAI and B2B integration differ in their focus and requirements. EAI software provides the infrastructure to rapidly connect and interface between an organization's internal applications. B2B integration can be regarded as an extension of EAI by integrating organization's applications with the applications of its partners. The three use cases we are going to introduce in Section 2.3 cover both EAI and B2B integration situations.

The technical integration of systems is usually driven by a goal. Three possible goals on the technical level can be observed according to Eyal et al. [8,21]:

1. Systems with similar functionality may be merged: Merging systems with similar functionality is an important issue in preserving data quality. If duplicated information is distributed over several systems and an integration of those is not considered then there is a high risk that information becomes inconsistent over time. For example, duplicated data is changed in one system but not in the other.
2. Complementary systems may be composed to gain new functionality: Composing systems to gain new functionality, is the main reason for integrating existing systems in EAM. For example, a new decision support system is introduced that requires data from sensors and the ERP system. Without integration, prediction of asset health conditions cannot be achieved accurately.
3. Existing systems may be customized with new features: New functionality is introduced to the environment but in this case it affects only one system.

The main reason for customizations is the ease of integration with other systems. For example, data extracted from sensor readings are filtered first and then transferred to an ERP system. Customization may be implemented within systems as an extension or as a separate component that can be reused in combination with other systems.

An integration of two systems can be established in two dimensions depending on the relationship of the systems. The dimensions are referred to as horizontal and vertical integration known from organizational integration.

2.2 Horizontal and Vertical Integration

In order to achieve effective decision support the information needs to be integrated from different disciplines and integrated on a high level. These two goals refer to the integration in two dimensions: horizontal and vertical integration. The horizontal dimension integrates information from different disciplines whereas the vertical dimension integrates information within a discipline [12]:

Horizontal Integration: Horizontal integration incorporates different systems that provide complementary functionality required to reach a certain business goal. Horizontal integration can be established by implementing a distributed business process that orchestrates activities executed in different locations. It defines the specific order in which activities are executed to achieve a business goal.

Vertical Integration: Vertical integration handles the integration of systems within a certain domain, sometimes on different levels of abstraction. Depending on whether we are dealing with an EAI or B2B integration situation, the levels of abstraction and goal of the integration are different. In EAI, the systems in the same domain are usually on different levels of abstraction. For example, a system consists of sub-systems and a sub-system may consist again of sub-systems. Vertical integration in EAI allows to abstract information from a low level and lift it to a higher level that is appropriate for further processing and decision support. In this way unnecessary information is hidden and an overall view of the underlying data is accomplished [18,1]. An important feature of vertical integration is the direct access of information on different abstraction levels. A high level view has the advantage of identifying abnormal conditions quickly, but for finding the reasons why the condition is abnormal, an approach is required that allows to navigate to the sub-systems in order to localize the cause.

In B2B integration, the information systems in the vertical dimension also share the domain but are usually on the same level of abstraction. They are usually integrated by merging for a particular reason, for example, for comparing competing businesses in a market or for accessing them in a unified way.

Horizontal and vertical integration in EAI and B2B integration are illustrated in Figures 1 and 2. When we look at the order in which the dimensions are integrated, vertical integration is typically performed before horizontal integration.

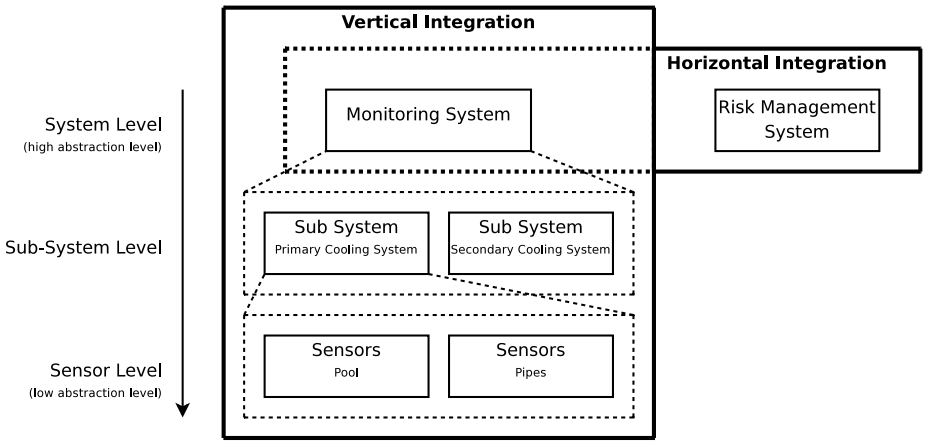

Fig. 1. Horizontal and vertical integration in EAI

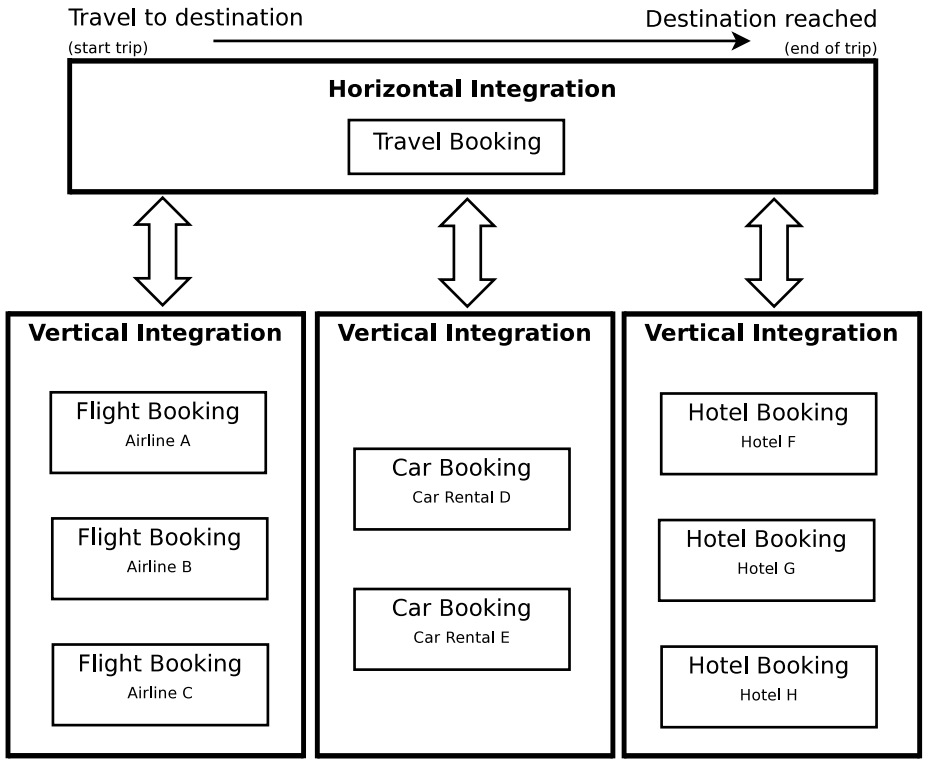

Fig. 2. Horizontal and vertical integration in B2B integration

The reason for this is that in the case of EAI systems, the more detailed data on a lower abstraction level needs to be aggregated first to a level where it can be integrated with other systems. In B2B integration, usually one or some systems are selected for integration on the horizontal level. In order to select a system, they must be vertically integrated first.

2.3 Use Cases

In this section we describe three use cases we have conducted in the asset management sector over the last 5 years. The first projects dealt with "plant monitoring and management" and has been conducted in collaboration with the CRC for Infrastructure Engineering Asset Management[1] (CIEAM) and the Australian Nuclear Science Technology Organization[2] (ANSTO). The second use case dealt with the automation of requirement engineering for interoperability and has been conducted with CIEAM and Mainpac Pty Ltd. [3]. The last use case investigated the digital handover of design documents to the operation and maintenance phase in the Oil & Gas industry and was conducted in collaboration with industry alliances and major CAD and software vendors in the Oil & Gas Interoperability area. In the context of the previous discussion on EAI/B2B and the integration dimensions, Use Case A covers horizontal and vertical integration in EAI, Use Case B covers horizontal integration in EAI and B2B and Use Case C covers the horizontal dimension in B2B integration.

Use Case A. Plant Monitoring and Management: In the first use case we considered a power plant management environment shown in Figure 3. It consists of five systems: The vertical dimension captured the "field data collection" which consisted of (1) an embedded sensor reading system, (2) a data filtering system, and (3) a field data collection system using personal digital assistants (PDAs). On the horizontal level, two systems were required to be integrated: (4) an enterprise resource planning system (ERP) and (5) a decision support system (DSS). The DSS was introduced later because it provided unique functionality that was required by the enterprise running the plant, functionality such as prediction of asset health and decision support in asset maintenance. Particular challenges in this use case were: how to interface with existing systems in a unified way, and how to deal with future changes that affect a new version of an existing software product or the introduction of new software to the enterprise landscape.

Use Case B. Capturing Interoperability Requirements: In the second use case, the problem was how to accurately capture and manage requirements for the interoperability of information systems within an enterprise (EAI) and between local and external systems at a business partner site (B2B integration). Requirements are negotiated in iterative sessions between an enterprise and its

[1] http://www.cieam.com
[2] http://www.ansto.com
[3] http://www.mainpac.com.au

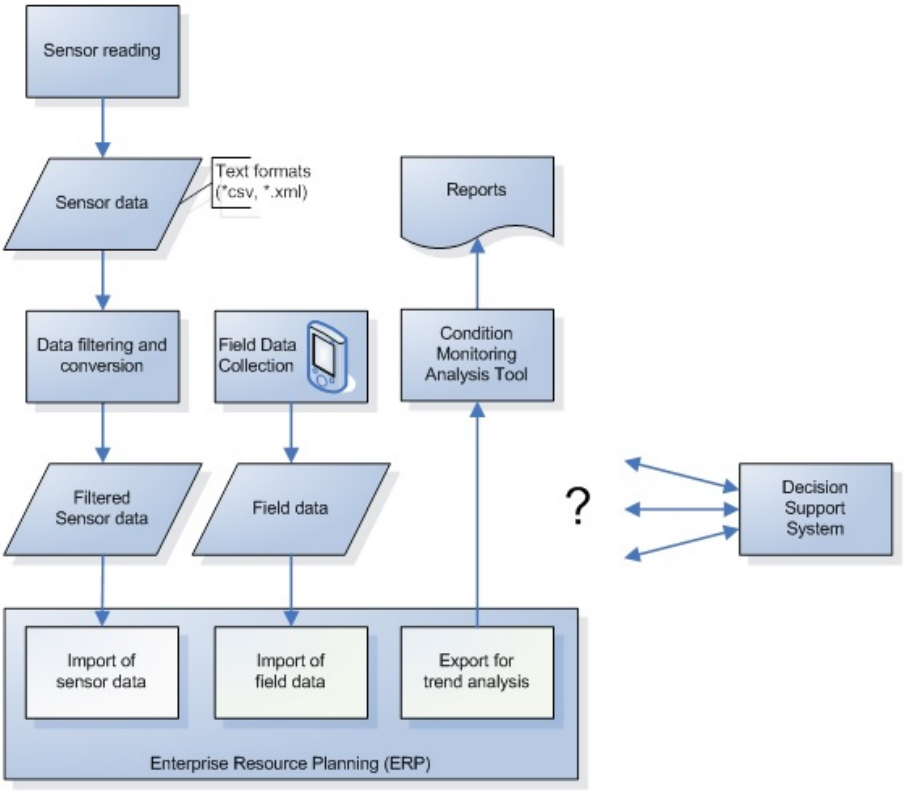

Fig. 3. Environment of Use Case A "Plant Monitoring and Management" [12]

partners and captured based on user input during meetings and on existing interfaces in the form of Web services definition files (WSDL) and C# code. From the requirements, a contract is formed in Microsoft Word documents and WSDL files and C# code are created which may be used as input for the next requirement capture cycle. Particular challenges in this use case were: how to capture requirements during interactive sessions with business partners, how to generate service contracts and interfaces automatically, and how to deal with changes and achieve consistency between interfaces, contracts, requirements and implemented code.

Use Case C. Standards-Based Interoperability in the Oil & Gas Industry: The third use case deals with the digital handover of documents from the design phase to the operation and maintenance phase. In the engineering space, interoperability is a major challenge in the information hand-over from one phase in the asset life-cycle to another [5]. To overcome the interoperability problem, considerable effort has been invested into the development of standards to serve as a *lingua franca* between computer systems. Two candidates of those standards are ISO 15926 and MIMOSA [20], which are currently applied

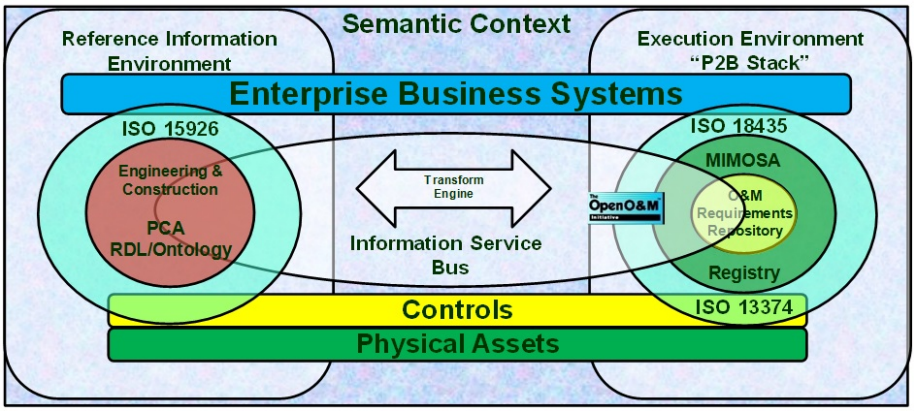

Fig. 4. Overview of Use Case C "Standards-Based Interoperability" (Figure provided by OpenO&M at ISA 2012 Exposition)

in the Oil & Gas Interoperability (OGI) Pilot in support of the ISO TC 184 OGI Technical Specification project. The particular challenge in this use case is how to map data in CAD tools into asset management tools using open standards and how to ensure that the mapping is correct according to the semantics of the underlying data model specifications [15]. An overview of the context is shown in Figure 4.

In the next section we discuss the common goals and objectives of the three use cases.

3 Requirements and Goals of a Sustainable Interoperability Solution

Although all three use cases cover a different type of integration and dimension, they share common requirements and goals. In this section we provide an overview of the key challenges and goals that have been defined in collaboration with the industry partners.

3.1 Requirements

Legacy Applications: Integrating systems that have been developed in the past remain one of the biggest challenges in interoperability. Because of limited human resources with necessary expertise the costs of changing and adapting legacy systems is usually so high that this option is often not considered as a solution. Alternatively, underlying data might be accessed directly, e.g., direct access of a relational database through SQL interface and bypassing the legacy application or developing a wrapper that maps an API of the legacy tool (if it exists) with state-of-the-art technologies.

Minimize Dependency on Existing Systems: Non-IT enterprises are often in the situation where they are locked into software support contracts, especially with large software vendors, and become dependent on particular systems that have been developed externally and dominate their tool landscape. Changing or adapting those systems usually requires high costs or is sometimes even impossible due to policy reasons. Software tools tend to be closed and only allow access through a limited API which becomes a challenge if no tools are acquired that need to access the same data. In some cases it might not only be necessary to access the data but also to feed data back into the systems, for example, a data quality tool performing data cleansing.

Comprehensive Integration: Since information can be integrated in different dimensions and within or across enterprise boundaries (see Section 2), a comprehensive solution is required that is able to support all different integration types in a single framework. Further to the already discussed integration types are the static and dynamic aspects of an information system that need to be considered, in particular *data integration* and *behavior-based integration* [23]. A fundamental requirement for a framework to handle data integration and behavior-based integration is the modeling language used. One of the first modeling approaches that combined behavior- and data modeling were Object/Behavior Diagrams [16]. More recently, the two modeling aspects received increased attention in artifact-centric business process modeling [4].

Security: An important issue in all three use cases, especially where critical infrastructure is involved, is security. Although integration always aims at a higher degree of automation where data is passed on automatically and information systems react automatically to events, it can be the case that data flow is limited because of security reasons. For example, supervisory control and data acquisition (SCADA) systems that control and monitor industrial processes in power plants are usually physically disconnected from the remaining enterprise systems for preventing the possibility of intruders accessing the system remotely and for restricting the access to critical infrastructure. The most prominent example for malicious software to exploit security holes in SCADA systems is the Stuxnet computer worm. Another security objective is that an integration solution should make use of the existing security infrastructure rather than try to bypass it, e.g., access data from an ERP system through the provided API rather than accessing the data directly.

Non-intrusive Solution: Today there exist many data integration solutions. About 60 tools are listed in a recent Gartner report on data integration tools published in October 2012 [24]. The drawback of a majority of those tools is that they require a lot of resources. They do not only require hardware and software resources but also human resources because people need to be trained to use, manage and maintain those tools. By introducing a new system to minimize the dependency, a new dependency might be created because of the complexity of the tool. Enterprises with limited resources are therefore looking for "non-intrusive solutions" that require a minimum impact on the existing infrastructure, a

minimum requirement on hardware and software resources and minimum effort in learning to use the tool.

Sustainable Solution: Sustainability is not only an important aspect in energy production and consumption but also in software development. Especially in the asset management sector where physical assets usually operate much longer than software systems, it is crucial to consider future changes and even the whole replacement of a software system. An interoperability solution should therefore support changes on different levels: (1) changes made to the software landscape, e.g., adding or removing whole systems and interfaces, (2) changes on existing systems due to upgrades or changes in the business strategy and (3) changes to the integration solution itself, e.g., how to transfer the integration logic from one system to another.

Verification and Validation: This objective focuses on the correctness of an interoperability solution and is divided into syntax and semantic correctness. Whereas syntax correctness is usually fully automated by verifying data according to provided specifications, the semantic correctness still remains challenging and often involves domain experts. As there does not exist a fully automated solution for a semantically correct integration, functionality should be provided that supports domain experts in verifying the data (e.g. through query functionality), visualization and back-tracking transformed data to its original source.

Performance: The last objective is the performance of an integration solution which is often an issue in real-time integration scenarios. For example, the instant persistence of data across systems. Since we did not deal with real-time data in the three use cases mentioned above, this objective was not critical. However, performance was an issue for the transformation of very large amount of data and the requirement was to execute it in minutes.

3.2 Goals

Based on the requirements listed above the following goals were defined:

- **Seamless, non-intrusive integration of required systems**: The main goal was establishing bi-directional data integration of existing systems with minimal impact on the existing landscape. This goal included overcoming heterogeneous data interfaces and structure, e.g., SAP RFC, relational databases, and ontology-like specifications of standards.
- **Open transformation**: Internal change of systems and adding new systems must be supported. The integration solution must be open so it can be extended for support of new systems and must be able to export its own integration logic to an open standard, e.g., Query-View-Transformation (QVT) or XSLT languages that can be imported into another transformation engine.
- **Centralized integration**: The solution should centralize integration logic in a single system and replace integration components in existing systems. This offers the advantage of a centralized management of the integration and leads to a more flexible landscape.

- **Cover all dimensions in EAI and B2B integration**: A solution needs to support horizontal and vertical integration within and across the boundaries of the enterprise.
- **Easy usability:** It should be usable by domain engineers who do not have the background knowledge of the underlying IT technology. The domain experts should be able to: (a) design, simulate and execute an integration solution, and (b) verify the integration with the help of visual functionality.

In the next section we describe how we addressed the above mentioned goals in a model-driven interoperability approach using multi-domain modeling.

4 Model-Driven Interoperability Approach

In this section we describe a model-driven approach we have applied to Use Cases A–C introduced in Section 2.3. The approach builds on multi-domain modeling languages that are semi-automatically generated and coherently form an interoperability framework to achieve the goals mentioned in the previous section. Before this approach is discussed in more detail, the methodology applied in the use cases is described from a Design Science Research perspective in which the modeling languages specify the central design artifacts.

4.1 Design Science Research

We first discuss how the approach relates to the design science research cycles [14]. Figure 5 shows the *relevance-*, *design-* and *rigor cycle* between environment, design science research and knowledge base. The figure is annotated with the information from the three use cases.

Environment: All three use cases came from the asset management sector. The first use case, *plant monitoring and management*, involved *engineers* responsible for the maintenance of power plant and *external software consultants* responsible for the installed enterprise resource planning systems that supported the engineers with their regular tasks. The information systems to-be integrated were commercial SCADA and ERP and a decision support systems built in-house. On the technical side, heterogeneous interfaces in form of plain CSV, Web service like interfaces and a relational database had to be bridged. The main problem was a non-existing interoperability solution and high costs in performing a manual data exchange. However, the opportunity existed to implement and deploy a state-of-the-art solution that automates the data exchange and to demonstrate a new way of designing interoperability that can easily be handled by engineers and software consultants without any knowledge of the underlying implementation.

Use Case B, *capturing interoperability requirements*, involved internal and external *software consultants* and *software developers*. They met in regular meetings and needed a design tool to interactively capture requirements for negotiating service contracts. The problem was that capturing the requirements was performed manually, producing a Word document, and partly re-created existing

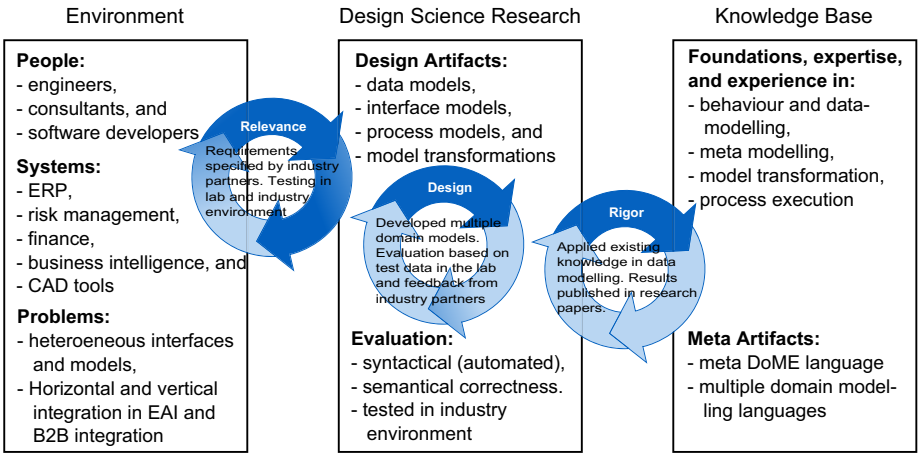

Fig. 5. Use cases in the context of Design Science Research according to the cycles by Hevner et al. [14]

interfaces rather than importing them automatically. The systems involved were a Business Intelligence software product with a relational database model in the background and various other systems, such as a finance system that can be accessed via Web service interface. There was an opportunity to introduce a model driven approach for human-computer interaction in capturing requirements interactively, re-engineering existing interfaces and generating documentation and code automatically.

Use Case C, *digital handover of design documents*, in the oil & gas sector involved CAD designers, engineers, owner & operators and large software vendors who built asset registries and enterprise resource planning systems for the operators. Systems considered in this use case were three different CAD software tools and one asset registry. On the technical side, the standards ISO 15926 and MIMOSA had to be used to specify the design and asset registry data, and an enterprise service bus specified by the OpenO&M Information Service Bus Model (ISBM) had to be used for the data exchange. The main challenge in this use case was identifying the overlaps and differences between ISO 15926 and MIMOSA and how the mapping could be specified and executed. This provided an opportunity to introduce a model-driven approach and model transformation techniques on a real-world scenario that involved complex standards.

Design Science Research: The design science research tasks consisted of developing design artifacts and processes as well as the evaluation. Artifacts were designed on two modeling levels: (1) On the meta model level a modeling language was constructed for each interface and data model and (2) on the model level artifacts were designed that represent a particular interface or data model. Further details on the artifacts and how they were designed and developed are explained in Section 4 below. The approach was evaluated in two stages, first within a test environment in the lab with a small data set and then a test environment

at the industry partner's site with a larger set of data taken from real-world use cases. The tests included syntax verification which was fully automated by the modeling framework and semantic verification which was supervised by domain experts using visual feedback provided by the modeling framework.

Knowledge Base: The knowledge required to solve the problems in the use cases came from the research team who designed and implemented a solution and domain experts who provided feedback to the research team in regular meetings for the semantic correctness of the integration. Specific technical knowledge and experience from four areas in computer- and information science contributed to the use cases: (1) behavior- and data-modeling, (2) meta modeling, (3) model transformation and (4) process execution and consistency rules for data and behavior modeling.

Relevance Cycle: The cycle between environment and design science research was conducted by the research team in collaboration with the domain experts in regular face-to-face and online meetings. In the beginning, the research team had to become familiar with the environment and industry partners and their domain experts provided requirements and goals. Field testing was performed either collaboratively or independently by the industry partner in multiple cycles. In all three use cases a first prototype was deployed after successful testing in the lab environment. This prototype allowed industry partners to test in their environment and provide feedback either through meetings or log files. While the research team improved the design and implementation, the industry partner could perform further tests and feedback was incorporated in a next cycle.

Design Cycle: The development of the design languages and design artifacts were mainly performed by the research team. The design (or modeling) languages were semi-automatically generated by querying meta data from the required interfaces and data models. These languages were used to model design artifacts for the execution of the integration task. Depending on the results of the field, the artifacts were modified or new types of artifacts were added on the meta model level.

Rigor Cycle: The rigor cycle between design science research and knowledge base was also mainly conducted by the research team. It included the application of existing research outcomes and knowledge into multiple software prototypes deployed to industry partners. Feedback and evaluation results were captured in experience reports and led to the improvement of software prototypes and new scientific results that were published in multiple publications.

In the following section we describe the model-driven approach in more detail.

4.2 Multi-domain Language Approach

We decided to apply a model-driven development (MDD) approach because of the well-known benefits that come with it [17]. In particular the following benefits were relevant for the use cases: (a) fast prototype development to demonstrate benefits to industry partners in a shorter time, (b) separation of concerns and

skills where design artifacts were used by domain experts to focus on the integration, and the research team focused on the implementation and execution of the artifacts, (c) bridging the gap between business (or engineers) and IT because IT systems are defined on a much higher-level using design artifacts, (d) results in software being less sensitive to changes, (e) design artifacts can be used for execution and for up-to-date documentation, and (f) platform independent modeling allowed to focus on the actual integration problem rather than implementation details.

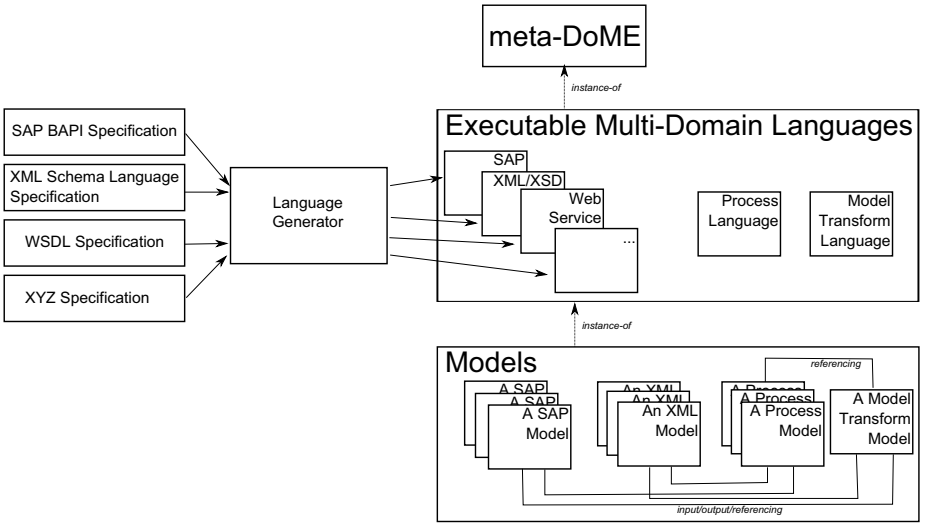

Fig. 6. Architecture overview of a multi-domain modeling approach for an interoperability solution in engineering asset management

One significant difference in our approach compared to existing MDD approaches in interoperability is the use of multi-domain modeling. Figure 6 shows an overview of the architecture. We created a domain modeling language with its own visual notation for each aspect of the framework: for each interface and data model specification, for the process-driven orchestration, and for the transformation of the data a separate modeling language was created. Through the use of multiple languages we maximized the benefits of MDD and allowed a domain expert in each aspect to focus on the problem with the help of a design tool. Each language has its own visual notation and constraints with which the domain experts are usually familiar. For example, an ERP system expert was able to verify an existing design in a fast and easy-to-understand way in our tool without knowing the tool and without the need to learn it beforehand.

Despite the advantage of this approach, we faced two challenges: First, how to develop a new modeling language for a model specification and second, how to integrate all languages and their models in a coherent way? For specifying a

modeling language we used the open source meta modeling tool DoME (Domain Modeling Environment). DoME comes with a visual meta modeling language called meta-DoME and a modeling tool generator similar to the commercially available MetaEdit+[4], and is available as a contributed package in the VisualWorks[5] development environment. We addressed the first challenge by a text-to-model (T2M) transformation in order to lift existing specifications to the model level. Each specification language that was applied for data models and interfaces in the environment, was mapped to meta-DoME, the meta modeling language in DoME. This enabled us to generate a modeling language in DoME from each model and interface which have been specified in a language that has previously been mapped to meta-DoME. For example, we developed a mapping for the XML Schema language and this allowed us to generate a modeling language for each XML Schema specification automatically.

Meeting the second challenge, integrating languages in a coherent way, was possible by using DoME. All languages were created with the same meta modeling language *meta-DoME* and therefore integrated in the same framework. An important feature in DoME is that elements from one language can be referenced in another language. We made use of this feature in a simple process language to integrate other language elements, e.g., model transformation tasks for the orchestration of different data sources. This combined behavior- and data integration in a single framework.

5 Outcomes

For each of the three use case we developed a prototype using the same meta modeling framework DoME. In all cases the prototypes focused on bridging syntactical and structural differences between data sources and provided the following features:

Interface: Support for the required interfaces mentioned above and additional interface for XML, Web service and relational database servers such as Microsoft SQL Server, Oracle, SQLite, and MySQL. The support for Web services enables the continuing support for SAP in the case that ANSTO decides to upgrade to the new SAP version called NetWeaver which is based on Web service technology.

Flexibility: The prototypes provided a generic business process modeling editor with a basic notation that allowed creating new integration solutions. A clear separation between orchestration and data transformation supports various scenarios and enhanced re-usability and flexibility.

Adaptability: The prototypes used existing communication technologies and interfaces and did not require additional resources. In can be seen as a light-weight approach compared to existing tools. All prototypes were deployed as a single executable file with additional configuration information installed by copy

[4] `http://www.metacase.com`
[5] `http://www.cincomsmalltalk.com`

and paste. The prototypes can be executed in two different ways during the design phase: (1) executing the whole integration process at once or (2) executing the process stepwise and observing data flow and transformation between various applications. After a prototype is deployed, it can be executed in batch mode which allows the scheduled execution on a server.

Usability of Application: All prototypes provided a visual editor that was easy to use and understand by domain experts. For example, the editor for orchestrating an integration came with a simple process language that consisted only of activities with data in- and output and data flows between them. Designing a business processes is supported by wizards which insert new activities into a process. For each software application type that had to be integrated one wizard was implemented and provided.

With the deployment and application of the developed prototypes in an industry environment we could reduced the manual steps from an average of +15 to 3 single steps in all scenarios. Furthermore all scenarios were supported by a single integration tool and could be combined in different ways, e.g., features that were developed for one use case could be deployed in combination with features developed for another use case.

6 Discussion and Conclusion

A state-of-the-start software environment in asset management must be able to accommodate a dynamic environment to allow the introduction of new software applications and their integration to cope with Enterprise Transformation. We have pointed out that integration needs to be considered as a separate component to optimize re-usability of integration knowledge and provide flexibility to the environment. We have proposed an integration architecture that supports horizontal and vertical integration and have demonstrated its application in three use cases that involved multiple industry partners. We have developed a light-weight integration solution and implemented prototypes which are currently used in industry. They fulfilled the requirements identified in the use cases which are (1) support for various data interfaces, (2) flexibility in building future integration solutions, (3) highly adaptable to the running environment, (4) ease of use for non-IT users, and (5) a significant performance improvement through the automation of manual steps.

Open challenges we have identified are matching heterogeneous data models and interfaces, coherent modeling of processes and data, and modeling events with business processes. There exist many matching techniques and tools for matching heterogeneous data structures [3] but in the three use cases, especially for matching heterogeneous standards as in Use Case C, existing approaches were not sufficient to identify possible matches. On the modeling side, there is still a lack of standardized languages for modeling processes and data flow in a coherent way. Artifact-centric business processes try to overcome this gap but so far no standard has emerged. Similar problems can be found for modeling events in the context of processes. The Business Process Model and Notation (BPMN)

language provides a set of events for modeling start, end and intermediate events in a business process but there is no guideline on how to use these events for orchestrating multiple systems in a correct way.

Future work includes addressing the challenges mentioned above, support for automated integration of service interfaces and the complex integration of engineering asset management standards. This includes the development of additional wizards which help to design and deploy integration solutions, the discovery and automated matching of services, and the complex mapping of data specified in standards such as MIMOSA and ISO 15926.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services – Concepts, Architecturse and Applications. Data Centric Systems and Applications (DCSA). Springer (2003)
2. Berger, S., Grossmann, G., Stumptner, M., Schrefl, M.: Metamodel-Based Information Integration at Industrial Scale. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part II. LNCS, vol. 6395, pp. 153–167. Springer, Heidelberg (2010)
3. Bernstein, P.A., Madhavan, J., Rahm, E.: Generic Schema Matching, Ten Years Later. In: Proceedings of the VLDB Endowment (PVLDB), vol. 4(11), pp. 695–701 (2011)
4. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
5. Braaksma, A.J.J(J.), Klingenberg, W(W.), van Exel, P.W.H.M(P.): A review of the use of asset information standards for collaboration in the process industry. Computers in Industry 62(3), 337–350 (2011)
6. Bussler, C.: B2B Integration. Springer (2003)
7. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M. (eds.): Process-Aware Information Systems. John Wiley & Sons (2005)
8. Eyal, A., Milo, T.: Integrating and customizing heterogeneous e-commerce applications. The VLDB Journal 10, 16–38 (2001)
9. Fiatech. Advancing Interoperability for the Capital Projects Industry: A Vision Paper. Technical report, Fiatech (February 2012)
10. Gallaher, M.P., O'Connor, A.C., Dettbarn Jr., J.L., Gilday, L.T.: Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry. Technical report, NIST (2004)
11. Gregory, N.: Excellence in Asset Management. In: Proc. of WCEAM, pp. 682–691. Coxmoor Publishing (2007)
12. Grossmann, G., Stumptner, M., Mayer, W., Barlow, M.: A Service Oriented Architecture for Data Integration in Asset Management. In: Proc. of WCEAM. Springer (2010)
13. Harmsen, F., Proper, H.A.E., Kok, N.: Informed Governance of Enterprise Transformations. In: Proper, E., Harmsen, F., Dietz, J.L.G. (eds.) PRET 2009. LNBIP, vol. 28, pp. 155–180. Springer, Heidelberg (2009)
14. Hevner, A., Chatterjee, S.: Design Science Research in Information Systems, ch. 2. Springer Science+Business Media, LLC (2010)

15. Jordan, A., Grossmann, G., Mayer, W., Selway, M., Stumptner, M.: On the Application of Software Modelling Principles on ISO 15926. In: MODELS Workshop on Modelling of the Physical World (MOTPW), Innsbruck, Austria (2012)
16. Kappel, G., Schrefl, M.: Object/Behavior Diagrams. In: Proceedings IEEE ICDE, pp. 530–539. IEEE Press (1991)
17. Kleppe, A., Warmer, J., Bast, W.: MDA explained - The Model Driven Architecture: Practice and Promise. Addison-Wesley (2003)
18. Kobayashi, T., Tamaki, M., Komoda, N.: Business process integration as a solution to the implementation of supply chain management systems. Information and Management 40(8), 769–780 (2003)
19. Mathew, J.: Engineering Asset Management – Trends, Drivers, Challenges and Advances. In: Proc. of WCEAM, pp. 59–74. Springer (2008)
20. MIMOSA. Open Systems Architecture for Enterprise Application Integration 3.2.3. Technical report, MIMOSA (2012)
21. Papazoglou, M.P., van den Heuvel, W.-J.: Business process development life cycle methodology. CACM 50(10), 79–85 (2007)
22. Pinkston, J.: The Ins and Outs of Integration - How EAI differs from B2B Integration. eAI Journal 8, 48–52 (2001)
23. Stumptner, M., Schrefl, M., Grossmann, G.: On the road to behavior-based integration. In: Proc. APCCM. CRPIT Series, vol. 31, pp. 15–22. ACS (2004)
24. Thoo, E., Friedman, T., Beyer, M.A.: Magic Quadrant for Data Integration Tools. Technical report, Gartner (2012)
25. Young, N., Jones, S.: SmartMarket Report: Interoperability in Construction Industry. Technical report, McGraw Hill, New York (2007)