Frank, Ulrich; Strecker, Stefan

Research Report

Beyond ERP systems: An outline of self-referential enterprise systems. Requirements, conceptual foundation and design options

ICB-Research Report, No. 31

Provided in Cooperation with:
University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB)

Suggested Citation: Frank, Ulrich; Strecker, Stefan (2009) : Beyond ERP systems: An outline of self-referential enterprise systems. Requirements, conceptual foundation and design options, ICB-Research Report, No. 31, Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB), Essen

This Version is available at:
https://hdl.handle.net/10419/58166

ICB
Institut für Informatik und Wirtschaftsinformatik

Ulrich Frank

Stefan Strecker

Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems

ICB-RESEARCH REPORT

Requirements, Conceptual Foundation and Design Options

UNIVERSITÄT DUISBURG ESSEN

ICB-Research Report No. 31

April 2009

Die Forschungsberichte des Instituts für Informatik und Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The ICB Research Reports comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Authors' Address:

Ulrich Frank
Stefan Strecker

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Universitätsstr. 9
D-45141 Essen

ulrich.frank@uni-due.de
stefan.strecker@uni-due.de

Contact:

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Universitätsstr. 9
45141 Essen

Tel.: 0201-183-4041
Fax: 0201-183-4011
Email: icb@uni-duisburg-essen.de

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)

Abstract

While it is widely agreed that current enterprise resource planning (ERP) systems cannot evolve much further, it is the question how future enterprise systems will look like. To approach an answer to this question, this report provides at first an analysis of essential high level requirements and their implications. Against this background, a conception of future enterprise systems referred to as self-referential enterprise systems (SRES) is presented. It is characterized by an elaborate conceptual foundation, which is based on multi-perspective enterprise models. These models enable enterprise software to not only refer to their conceptual foundation, but also to the concepts that characterize the context they are embedded in. In order to promote cross-organizational integration and high-level reuse, it is suggested to develop reference models, which provide design blueprints for an entire range of organizations. While the development of reference models and the realization of corresponding SRES is a very attractive research vision, it overextends the resources of single research institutions by far. It also requires involving prospective users. To cope with these challenges, the paper suggests a conception of collaborative development of open reference models. It concludes with the description of the recently launched "Open Model" initiative that is aimed at promoting the joint development of large conceptual models which could serve as a foundation of future SRES.

Keywords: integrated enterprise systems, conceptual foundation, multi-perspective enterprise models, self-referential systems, open systems, open models

Table of Contents

Figures

1 Introduction

In the past decades, enterprise resource planning (ERP) systems have determined the shape of large corporations' information systems. With an original focus on manufacturing and production planning, current systems meanwhile cover many other business functions including financials, human resources, and customer relations (Klaus et al. 2000; Markus and Tanis 2000). Available systems have also evolved economically and matured technically over the years. Users, consultants, and developers have become increasingly familiar with ERP systems and their siblings, e.g., customer relationship management (CRM) and supply chain management (SCM) systems, which, in turn, positively affects their overall economics. At the same time, both hardware and software technology has evolved to add to the reliability of present day systems.

Despite the constant evolution of ERP systems, awareness of their key issues has increasingly been raised. In fact, the limitations and shortcomings of ERP systems have been a recurring theme in the on-going discussion on yet the next-generation of enterprise-wide information systems for quite some time now. Both academics and IT professional point at insufficient support of business functions and processes (Markus et al. 2000), increased complexity of business processes (Shang and Seddon 2007), increased overall system complexity (Rettig 2007), deficient standard processes (Davenport 1998), unsatisfactory support of supply chain management and other forms of inter-organizational collaboration (Davenport 2000), among others. While these shortcomings point at potential requirements for future enterprise systems, their persistent recurrence suggests, on the one hand, that current ERP systems do not provide a suitable foundation for enterprise-wide information systems in the long run, and, on the other hand, that alternative conceptions for future developments have not yet been convincingly demonstrated.

Unfortunately, much of the on-going discussion remains at the level of business requirements and does not appreciate the technical intricacies of enterprise-wide information systems. At the same time, systems vendors propose new technologies as yet another panacea to solve the mentioned and further shortcomings. Whether enterprise application integration (EAI), componentization, or, more recently, service-oriented architectures (SOA), these proposed solutions emphasize technical concepts for developing software systems and do not adequately account for the organization as surrounding action systems in which enterprise systems are embedded (Sprott 2000; Fan et al. 2000). In particular, neither technology is based on (high-level business) concepts commonly used in enterprises such as invoice, customer, or product. Moreover, each technology addresses only specific issues with current ERP systems: EAI focuses on ex-post integration through middleware and data warehousing. Hence, metaphorically speaking, it is aimed at hiding the mess without removing it. SOA promotes the vision of "orchestrating" process-oriented information systems by selecting appropriate services, which are provided through standardized interfaces. It does not,

however, account, e.g., for data integrity across services. Although these technologies deserve consideration as candidate solutions for future software infrastructures, their use does not provide a comprehensive conceptual foundation critical to future enterprise systems.

Ultimately, the discussion on future enterprise systems has led to a confusing diversity of purported requirements, suggested solution proposals, and subsequent promises. It is, thus, difficult – not only for the interested user – to shed light on the status quo let alone assess future developments (see also Keyser 2007). The situation is further complicated by – sometimes subtle – commercial interests and hidden agendas of participating discussants (Carr 2003; Rettig 2007). It is, therefore, hardly possible to derive the shape of future enterprise systems from the current state of discussion alone.

In this respect, our motivation is twofold: Firstly, we observe demand pull and technology push forces with respect to the design of future enterprise systems that are largely driven by systems vendors, systems users, and consultants. IS research is taking a rather passive role in this respect, observing and interpreting rather than designing and contributing. Secondly, the recurring discussion on yet the next generation of enterprise systems in academic and industry literature treats proposed solutions largely apart from each other in an eclectic fashion; not to mention the predominant focus on the usual three-letter abbreviations (Møller 2005). The discussion is apparently missing an unifying view as it is provided by architectural and conceptual considerations.

Against this background, our contribution is aimed at outlining a conception of future enterprise-wide information systems. In order to distinguish our conception clearly from existing enterprise systems, we refer to them as self-referential enterprise systems (SRES). We start from high-level requirements for SRES (Sec. 2). We then show how multi-perspective enterprise models provide a comprehensive conceptual foundation for SRES (Sec. 3). The realization of such a foundation poses several challenges to both IS research and practice. In Sec. 4, we present an approach to collaboratively build a comprehensive system of "open models" of the enterprise, which does not only address current barriers impeding research contributions to the development of future enterprise systems, but also serves as a model for purposeful cooperation of (IS) research and practice. In the final section, we report on an initiative aimed at realizing such an approach.

2 In a nutshell: Requirements for Self-Referential Enterprise Systems

In this section, we focus on generic requirements we regard as essential for the design of SRES. We will show that these high level requirements cover many of the more specific ones discussed in literature (e.g., Davenport 2000).

2.1 Integration

The claim for integrating different business applications is characteristic for ERP. It promises to reduce redundancy and friction, hence to foster efficiency and integrity. To further emphasize and structure the claim for integration, we suggest the following generic requirement:

Generic Requirement #1: SRES should feature a **high level of multi-dimensional integration.**

Description: In order to describe this requirement and to derive its implications, we first need a more elaborate conception of integration than the one suggested by its colloquial meaning – "unification into a whole". Within the context of information systems, integration is mainly a linguistic conception, i.e., it is accomplished through language and communication respectively. Integrating two components requires them to communicate, either directly or through some kind of mediator. Communication in turn implies the existence of common concepts that define the semantics of the linguistic artifacts that are subject of communication relationships. In other words: Integration requires the existence of a *common semantic reference system*. Examples for such reference systems are data types or database schemas. The level of integration can be conceptualized by referring to the semantics of the common concepts (Frank 1994, p. 30): The higher the level of semantics these concepts include, the higher the level of integration they allow for. Note that this concept of semantics corresponds to the concept of information content: The more possible interpretations are excluded by a concept, the higher its semantics. For example, the concept "Customer" is of higher semantics than the concept "String" (data type). A high level of integration offers definite benefits. The higher the level of semantics, the more focused and efficient information exchange will be. At the same time, a high level of semantics reduces the effort that is required for reconstructing the meaning of a message; hence, it also reduces the threat a message imposes to integrity.

Refinement: To further refine the requirement, we differentiate basic dimensions and relevant scopes of integration. *Static integration* is accomplished through shared static structures. A typical example would be a common database schema used by a number of applications. *Functional integration* is aimed at linking applications by providing them with common functions, e.g., a common function library. It requires static integration to allow for common interfaces. Function libraries that are accessible throughout an entire – potentially distributed – platform are a typical example for functional integration. While these libraries would allow for cross-application functional integration, integration is usually restricted to those functions that are provided by vendors of operating systems or popular office systems. Usually, these generic functions offer only limited semantics. Object-oriented integration is a combination of static and functional integration. Finally, *dynamic integration* is aimed at synchronizing contributions of different applications to support a certain (business) process. It is accomplished through common event types. Only if an event that is produced by a software component corresponds to a common event type, another component (e.g., a workflow man-

agement system) can interpret it adequately and trigger the appropriate function – which presupposes functional integration.

While integration is often restricted to system integration, it can also be applied to a wider scope. *Organizational integration* denotes the integration of an information system with its surrounding action system. It includes overcoming the notorious cultural chasm between IT experts and business people. This kind of integration requires concepts that are shared by the organizational universe of discourse *and* the information system. If, for instance, the conception of business process types that are supported by an SRES is different from those that are actually implemented in a company, this lack of common concepts would promote friction. Also, if the semantics of common concepts is rather poor, it is likely that the potential of an information system cannot be exploited properly. Take, for instance, a concept like "file": It is often used as a generic concept to cover various types of documents. While these document types are characterized by a specific meaning within an organization, their (formal) semantics is reduced to an amorphous, flat concept. It is evident that such a semantic reduction does not only compromise convenience and efficiency of use, but is also a threat to an information system's integrity. Instead, a higher level of organizational integrity would be characterized by a correspondence of terms used in the organizational universe of discourse and the concepts implemented in the information system on a higher level of semantics. This includes static aspects, i.e., concepts that represent certain classes of artifacts or objects, functional aspects, i.e., concepts that represent tasks and/or functions, and dynamic aspects, i.e., concepts that serve to describe the execution of processes. In the case of cross-organizational integration of information systems, it is required to develop common concepts for those parts of the involved information systems and organizational processes that are subject to cross-organizational cooperation.

Challenges: The development of common semantic reference systems requires carefully analyzing the universe of discourse in the targeted companies to first identify those concepts – static, functional and dynamic – that are suited as a foundation for an SRES. In a next step, it is necessary to analyze similarities in order to develop *abstractions*, which characterize commonalities of a class of concepts. This process implies a remarkable intellectual challenge. Usually, it will *not* be sufficient to focus solely on concepts that are part of an existing organizational universe of discourse. Instead, the introduction of an SRES may require changing work objects and related patterns of action. Therefore, the required abstraction includes transcending observable organizational reality and reconstructing the resulting linguistic representation using appropriate languages. Since cross-organizational processes may include companies from various industries, reconstructing common concepts implies additional challenges. The more semantics a concept incorporates, i.e., the more specific it is the higher the chance that it cannot be (re-)used for integrating further systems, e.g., software components or information systems in other industries. This conflict is well known from philosophy of science: the higher the information content (semantics) of a proposition, the more likely it will be falsified (Popper 1982, p. 77). To cope with this problem, meta-level concepts can

be introduced that allow for defining specific concepts. Hence, any component that has access to the specification can reconstruct the intended semantics. For example, the semantic variance of a term such as "product" may not allow for specifying one unified concept. In this case, a meta model that provides concepts for defining product types (such as "feature name", "feature type" etc.) would still allow for expressing elaborate product semantics (for an example see Frank 2002a).

2.2 Flexibility

Considering the variety of requirements and their change over time, it seems compelling to demand flexibility:

Requirement #2: SRES should support **safe and convenient flexibility**.

Description: In the context of SRES, flexibility is a conception that incorporates three aspects. Firstly, it refers to the scope of additional requirements a system can be adapted to. Secondly, it refers to the effort it takes to perform adaptations. Thirdly, flexibility includes the (implicit) claim for safety, since software adaptations tend to jeopardize system integrity. Flexibility pertains to all dimensions of an SRES: the adaptation of static structures, of functions and of processes. The different aspects of flexibility are not independent. The highest level of adaptability would be to open the entire system code for manipulation. Apparently, this approach to flexibility would neither be convenient nor safe. More convenient approaches to adaptation, e.g. the modification of certain features in a class schema, imply the risk of producing side effects. The manipulation of tables that is sometimes used for the configuration of ERP systems is certainly not particularly convenient. It may be safe in a formal sense (depending on the incorporated integrity constraints) – however, due to their rather cryptic representation, tables are a likely source of human errors. Against this background, we can express a more concrete demand for flexibility. An SRES should allow for adaptations within the scope of possible future changes. While this demand does not imply the ability of predicting the future, it suggests a thorough analysis of the respective domain that is aimed at identifying those parts that are likely to be invariant and others that may be subject to changes in respond of certain future scenarios. Convenient modifications require a representation of the relevant aspects of a system in a comprehensible manner, i.e., by concepts that correspond to domain-level terminology. Graphical depictions may further foster comprehensibility, because they often allow for a clearer representation of complex interrelations.. For adaptations to be safe, the resulting changes should be clearly specified and mediated to those who perform the changes. Side effects are to be avoided.

Refinement: In software engineering, modularization is a common approach to foster maintainability: A system is composed of modules, which provide interfaces to hide their internal implementation. If the system needs to be adapted to changing requirements, those modules that are affected would be replaced or changed without affecting the rest of the system. However, it remains the question, how to design appropriate modules. The pivotal prerequi-

site for convenient and safe adaptations is reliable knowledge about the scope of possible future changes. Based on this knowledge, catering for flexibility requires developing abstractions, which represent essential common features of a set of possible concrete realizations. These abstractions need to fulfill certain requirements. Firstly, there should be a well-defined procedure for creating concrete realizations. Secondly, adding or changing concrete realizations should not affect the abstraction. Thirdly, the abstraction as such should be stable, i.e., whenever the abstraction requires modification, this should not result in any conflicts with existing concrete realizations. A well-know means of abstraction is generalization: Adding specialized concepts does not affect the general concept. Changes to the general concept are effective and make sense within the special cases as well. Other examples are polymorphism or encapsulation. The concept of an "aspect" (Lopes et al. 1997) provides a weaker abstraction. It serves to mark those parts of a system that belong to a specific concern. The notion of an aspect, however, lacks a precise conceptual meaning. Meta modeling is a powerful abstraction mechanism to support flexibility. It is not restricted to a set of domain-specific concepts, which may by possibly further specialized. Instead, it is based on a language that allows to specify specific concepts for a variety of domains. This is accomplished by instantiating concepts from meta concepts. If, for example, a meta model contains the meta concept "Process", it could be instantiated in business process types such as "Order processing" or "complaint management" - an adaptation that would not be possible through specialization. However, understanding meta models and instantiating concepts is a challenging task that is probably not regarded as convenient by many. It also implies significant software engineering challenges, which are – among other things –related to the fact the prevalent software architectures do not include a level of abstraction that would correspond to meta models (for a discussion see, e.g. Frank 2002b).

Challenges: Predicting possible future variance is demanding and costly. This is also the case for developing and implementing powerful corresponding abstractions. Therefore, it is required to find an appropriate balance between a high level of flexibility and the effort it takes to accomplish it – for instance, by focusing on change that is most likely to happen.

2.3 Transparency

SRES as well as enterprises themselves are complex systems. It can be expected that this complexity will grow further. Hence, managing complexity is a major prerequisite for successful action.

Requirement #3: SRES should efficiently contribute to **transparency**.

Description: An SRES is an invisible artifact. Linguistic representations are our only chance to develop an appropriate understanding. In order to foster transparency, these representations should be restricted to those aspects that are significant for specific purposes and users. Taking into account that professional perspectives of those who deal with SRES vary to a great

extent, this implies to provide representations of multiple perspectives. Graphical representations should be used where they foster transparency.

Refinement: It is an expression of sophisticated system design when a system includes the schema it is instantiated from. In order to foster transparency, the conception of a schema needs to be taken one step further: Firstly, schemas should cover all relevant aspects of an SRES (and not just its database). Secondly, they should represent concepts known and understood in the intended application domain. Thirdly, there should be a mapping to prepare the content of a schema for the needs of relevant analysis scenarios and user groups. This would include graphical representations, e.g., of business processes. In addition to that, this conceptual type level should be integrated with the instance level in a transparent way. If, for instance, a user first studies a business process model, he should then be able to navigate to its corresponding instances.

Challenges: While there is no doubt that transparency requires multiple abstractions of an SRES, it is difficult to design abstractions that are most appropriate for certain purposes and/or users. Groups of users are heterogeneous. Also, individual learning processes may change preferences for representations of the system.

2.4 Support for Analysis

The design of ERP systems was originally aimed at supporting business operations. The data collected on this level can be aggregated according to the need of specific analysis scenarios. This approach to support managerial analysis is often addressed by additional systems, such as analytic information systems, e.g., data warehouse systems, online analytical processing (OLAP) and business intelligence. However, this kind of ex post analysis is limited in two ways. On the one hand is depends on the availability and quality of data that were designed for different purposes. On the other hand, it does not account for those analysis methods that are beyond advanced data analysis approaches, namely those that require a higher level of data semantics. .

Requirement #4: SRES should support decision makers with elaborate **analysis methods**.

Description: In general, a method consists of a linguistic structure and a corresponding process model. In the case of methods to support managerial decision making, a method requires an elaborate terminology, which allows for addressing managerial problems on an appropriate level of abstraction as well as an adequate, adaptable process that guides the problem-oriented use of the terminology. With respect to the reduction of complexity and the need for communicating with different stakeholders, methods that make use of graphical representations promise particular benefits.

Refinement: SRES should include tools that guide the analysis of managerial problems on various levels of detail and complexity. For this purpose, these tools should represent analysis methods in general, modeling methods in particular. A modeling method consists of (at

least) a (domain-specific) modeling language and a corresponding process model. With respect to the variety of managerial problems, the tools should allow for the configuration of methods, e.g. by adaption of a process model.

2.5 Reuse

Overflowing costs are a major concern with corporate information systems in general and with ERP systems in particular. Hence, taking into account that future SRES should be more powerful – and complex – than today's systems, their economics becomes a major issue.

Requirement #5: SRES should take advantage of **software reuse**.

Description: There are many aspects that influence the costs generated by an SRES. We shall, however, focus on one generic aspect that is most promising: The reuse of software artifacts allows for dramatic cost reduction through economics of scale. The benefit of reusing a software component depends on its semantics: The more semantics it includes (the more specific it is), the higher the benefit of using it. If one can reuse an accounting module, the benefit is certainly higher than that provided by a sort function. At the same time, however, the range of reuse is compromised by semantics: The more specific a component, the less likely it can be used in further systems. Therefore, it is required to develop an appropriate balance between reuse benefit and economics of scale, or – in other words – to find the appropriate level of semantics.

Refinement: The primary objective for accomplishing reuse that combines high benefit and low cost is to identify or construct commonalities that are shared by a large number of enterprises. In other words, there is need for abstractions (again: static, functional, dynamic) that are suited to fit many enterprises. On the one hand, this requires analyzing many organizations. This is comparable to the typical approach in scientific research – not to focus on single occurrences only, but to analyze essential features of an entire class of instances. On the other hand, it is required to distinguish between those parts of the system that are invariant within the group of intended users and other parts that may need individual adaptation. Hence, this requirement corresponds to the demand for flexibility. It can be assumed that current costs of ERP systems are not only reflecting a lack of reuse, but also the market power some large vendors have managed to create. Hence, more competition could serve as a remedy. To foster competition, an SRES should be open. In a technical sense, this means that all relevant interfaces are published and that modules are interchangeable. In a conceptual sense, it means to publish the conceptual foundation of a system.

Challenges: Beyond conceptual and technological prerequisites of reuse, its actual realization depends on political processes, e.g., the specification of standards, and the marketing strategies of large vendors. From a scientific point of view, these aspects are hard to influence.

The overview of essential requirements shows that they are not independent. Fig. 9 on page 26 illustrates relations between these requirements. These essential requirements have in

common that they all imply to carefully design conceptual abstractions. In the following, we will outline the cornerstones of a conceptual foundation of future SRES that account for the shown generic requirements.

3 Cornerstones of a Conceptual Foundation for Self-Referential Enterprise Systems

A basic idea of the conception of SRES proposed in our approach is the integration of enterprise software with a corresponding enterprise model. Therefore, we will first outline a possible architecture for enterprise models and then show how it can be integrated with enterprise information systems. The elements of the proposed conception will be reflected against the requirements discussed above.

3.1 Multi-Perspective Enterprise Models

Proposing multi-perspective enterprise models as a conceptual foundation addresses all requirements mentioned in Sec. 2. An enterprise model integrates conceptual models of an information system, e.g., object models or state diagrams, with those of the surrounding action system, e.g., value chain diagrams or business process models. The various model types provide views on an enterprise that foster communication across different groups of stakeholders. At the same time, they allow for refinements that correspond to specific analysis and design tasks. ARIS (Scheer 1992), SOM (Ferstl and Sinz 1998) and MEMO (Frank 1994; Frank 2002c) are examples of enterprise modeling approaches. The term "multi-perspective" is to emphasize that an enterprise model should include models that correspond to the professional perspectives of those who are involved in designing, implementing, using, and evaluation information systems. While conceptual models focus on reconstructing domain level concepts, they need to account for restrictions imposed by design or implementation level languages, e.g., programming languages. Otherwise their support for system design would be compromised. To cope with this challenge, enterprise models require semi-formal languages, which allow for a convenient mapping to implementation-level languages. Fig. 1 illustrates the structure of an enterprise model. It is based on the framework and the modeling languages provided by MEMO. The framework differentiates two basic perspectives on an enterprise: *strategy* and *organization* (or *operations*). The strategy perspective includes models that emphasize a top management view, e.g., strategy nets (Frank and Lange 2007), value chains (Frank and Lange 2004), models of strategic resources, or scenarios of strategic decision making. The operational or organizational perspective includes, among other things, business process models, organizational charts and information models such as class diagrams. The domain-specific concepts represented in an enterprise model foster a high level of systems integration.

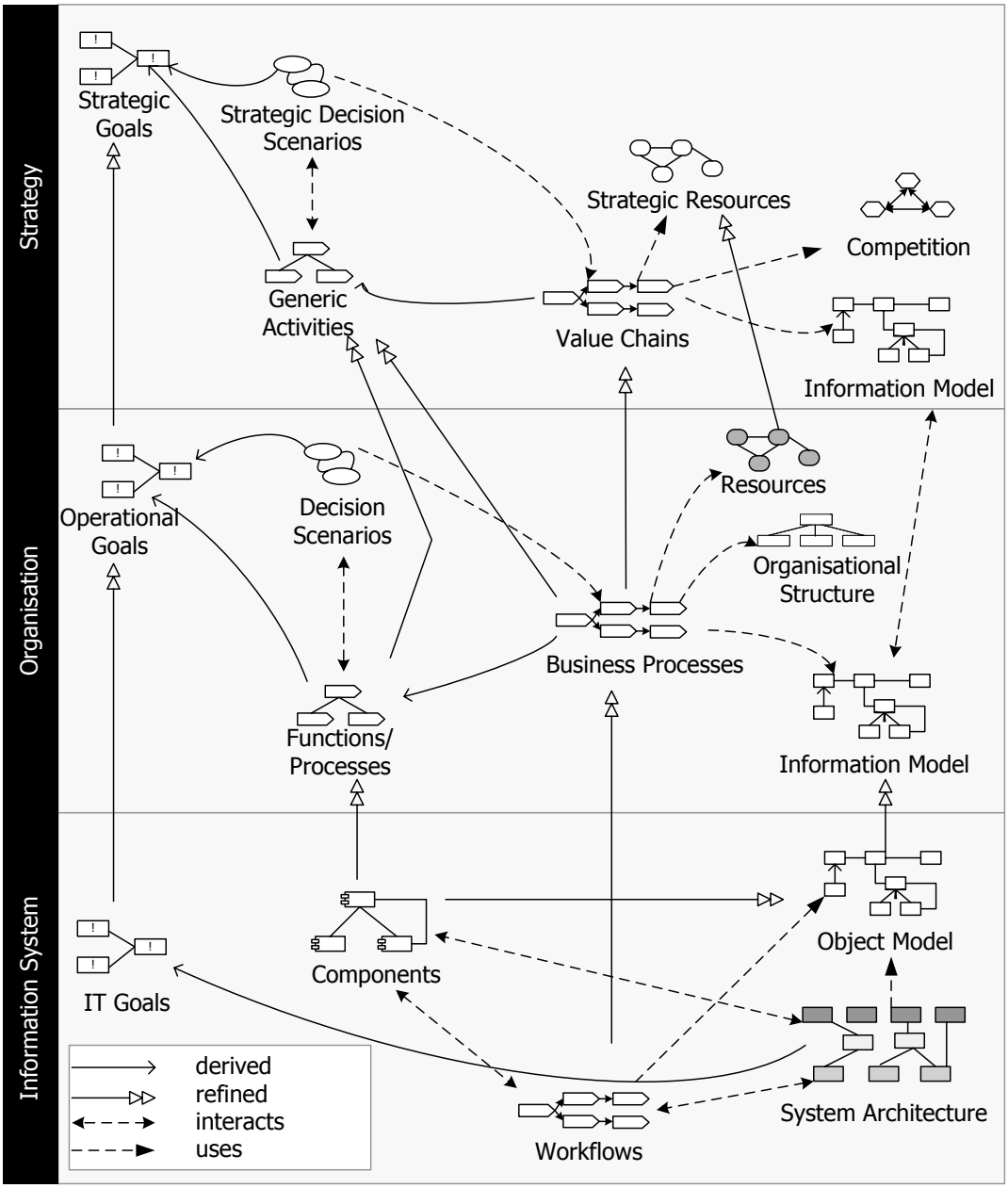


Fig. 1: Illustration of a multi-perspective enterprise model as a conceptual foundation for SRES

They also facilitate organizational integration, because they integrate conceptual models of software systems, e.g. object or component models, and the surrounding action system. Conceptual models are mainly used within systems analysis and design. They allow for abstracting from ever changing details of the underlying technology. Thereby, they support the construction of systems that are resistant against the perils posed by technological change – fostering a higher level of flexibility and the protection of investments. At the same time, they promote a better understanding of a system's functionality, thereby fostering the participation of prospective users and other stakeholders who are not IT experts. Enterprise models take this approach one step further by not only representing the conceptual foundation of a software system but also of the action system, it is embedded in. This allows for analyzing

software within the relevant business context and promotes a better alignment of IT and business. In addition to that, enterprise models promote modeling productivity and model quality: Enterprise models are – among other languages – specified by domain-specific modeling languages. These languages provide concepts that were reconstructed from elaborate technical languages of the respective domains. Therefore, modelers are not required to build these concepts from primitive concepts such as "class" or "attribute". Since domain-specific languages include syntactical and semantic constraints, they help to prevent building models that are inappropriate. Fig. 2 illustrates the use of enterprise models during the build-time phase. Note that the figure shows a simplified structure of an enterprise information system. The edges that connect the elements of the run-time level with those of the schema level remain unspecified on purpose. In an ideal case, specific run-time systems – for managing GUIs, object systems or workflows – operate on instances of the respective schemata.

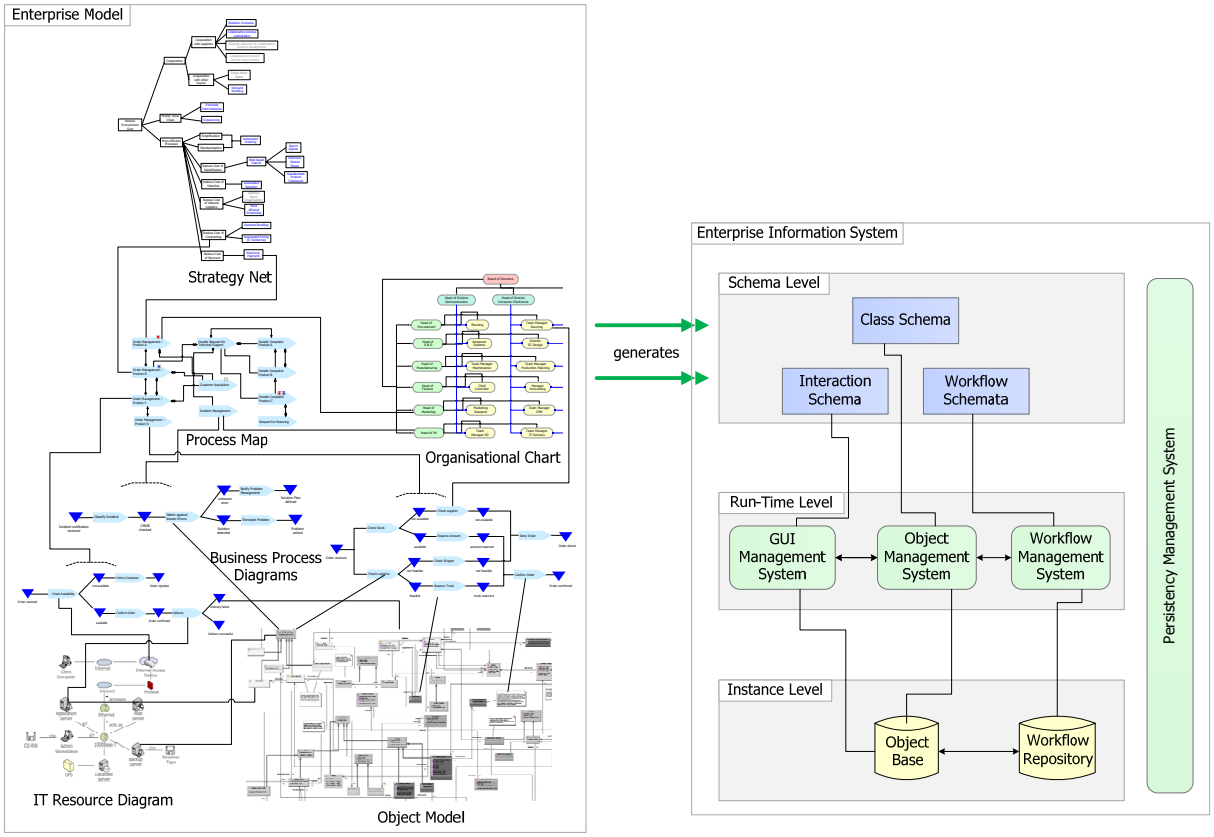


Fig. 2: Using enterprise models during the build-time phase for generating schemata

While an enterprise model represents a versatile medium to promote communication between various stakeholders, it is also very well suited for generating implementation level documents. This is for two reasons. Firstly, enterprise models integrate static (e.g. object models), functional (e.g. service models), and dynamic (e.g. business process models) abstractions. Therefore, it is possible to generate implementation level representations that integrate static, functional and dynamic aspects. In an ideal case, the entire code required of the corresponding software, including the GUI (represented by "interaction schema" in Fig.

2), can be generated. The following, simplified example, illustrates such an approach. The processes a business process model is composed of can be associated with the methods of classes within a corresponding object model that they require (see Fig. 3). For each class within the interfaces of these methods, a default interaction element (e.g. text field, button, list etc.) could be specified. It would then be possible to generate a class schema from the object model together with a workflow schema that refers to the class schema, for an example, see e.g. Jung (2004). For each context within the workflow, a user interface could be generated, too – by composing it from the default interaction elements defined for the services. Needless to say that it is not always possible to generate the entire code required to run the targeted software. If methods are required that allow for more than accessing object attributes, further, more detailed representations are required. Fig. 2 illustrates the generation of implementation level documents by referring to the schema level of an idealized enterprise information system architecture. Note that the generation process could be split into more than one phase. In a first step design level documents, e.g. object models enhanced by constraints, could be generated, which would then serve to generate implementation level schemata.

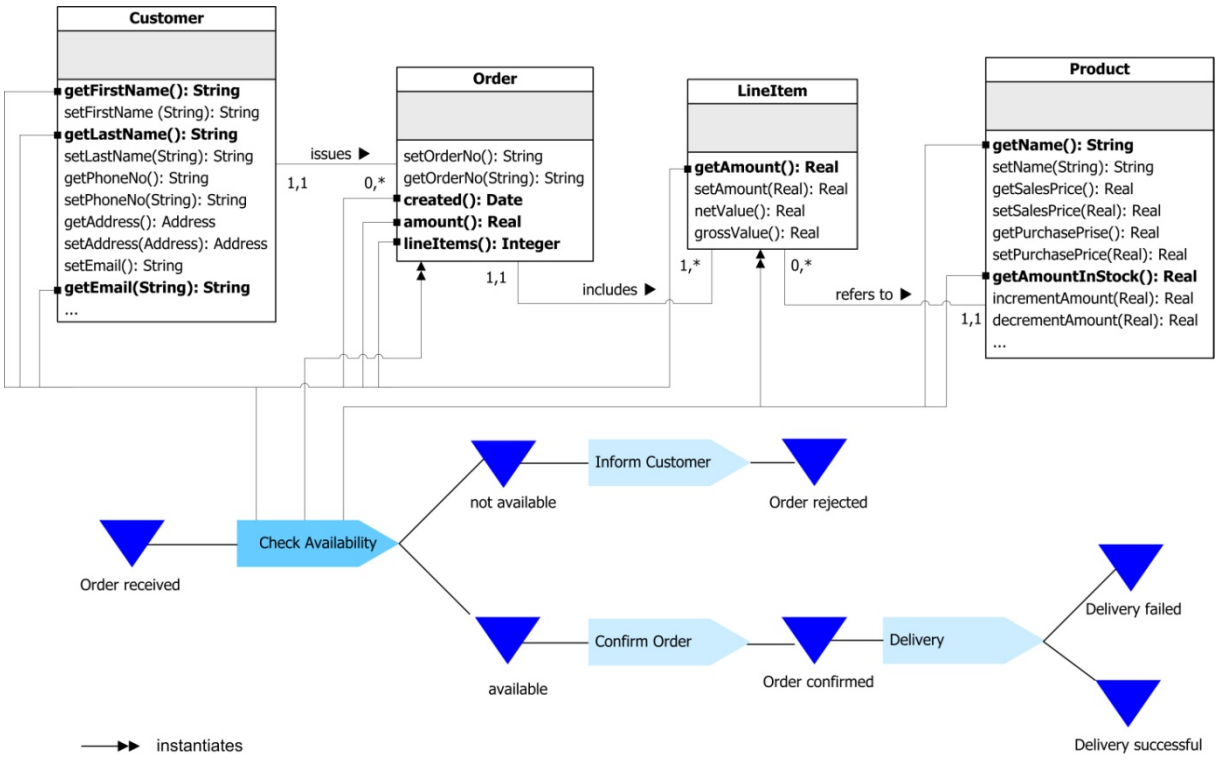


Fig. 3: Integrating a business process model with an object model

Secondly, enterprise models incorporate a higher level of semantics than traditional conceptual models used for systems design: On the one hand, the domain-specific modeling languages used for enterprise models allow for expressing more specific semantics than generic modeling languages such as the UML. On the other hand, enterprise models include repre-

sentations of aspects that are not accounted for in traditional conceptual models, e.g. models of organizational units or of the corporate strategy.

While there is a plethora of approaches to generate code from different kinds of representations (e.g., Pastor and Molina 2007; Sousa et al. 2008; Agrawal 2003; Phung-Khac et al. 2008), there is no comprehensive solution to the problem of synchronizing model and code evolution. In an ideal case, adaptations to changing requirements are performed on the enterprise model rather than on the implementation level (generic requirements #2, #3 as specified in Sec. 2). However, generated code will often have to be supplemented by further code or modified accordingly. Hence, re-generating code from an updated enterprise model would threaten to destroy code extensions. With respect to the time pressure and the lack of resources characteristic for many change requests, it is very likely that the semantic gap between the enterprise model and the corresponding enterprise information system will grow over time.

3.2 Prospects of Integrating Enterprise Models and Enterprise Information Systems

To better understand the benefits to be expected from SRES, we will focus on two key aspects. Firstly, we will demonstrate the use of enterprise models at run-time. The demonstration focuses on a system's capability to refer to the context of its use. Secondly, we will illustrate how the use of domain-specific modeling languages enable SRES to refer to the concepts they are based on as well as to refer to the concepts that characterize their application context. This also facilitates rich communication between different SRES – hence contributing to more versatile and efficient cross-organizational cooperation promising to reduce transaction costs considerably. To address the problem of synchronizing the evolution of an enterprise model and the corresponding information system, we will further outline our conception of SRES. It is aimed at overcoming the separation of information system and enterprise model. For this purpose, it emphasizes the integration of both through a common conceptual foundation. This architecture would extend enterprise information systems with the ability to refer to the concepts they are based on as well as to the concepts that characterize the context they are embedded in – hence, it enables *self-referential* information systems. It also stresses the use of enterprise models not only during build-time, but also during run-time.

3.2.1 Using Enterprise Models at Run-Time

Making an enterprise model – and the corresponding modeling environment – an integral part of an enterprise system allows for using models at run-time as well. In today's business environment, presentation software, e.g. Microsoft PowerPoint, is frequently used to provide illustrative sketches of complex decision scenarios. Usually, they are created for one or a few occasions without further reuse. Moreover, slides do not incorporate the semantics of the representation. As a consequence, they permit arbitrary and devious modifications, while they do not allow for performing automated analyses.

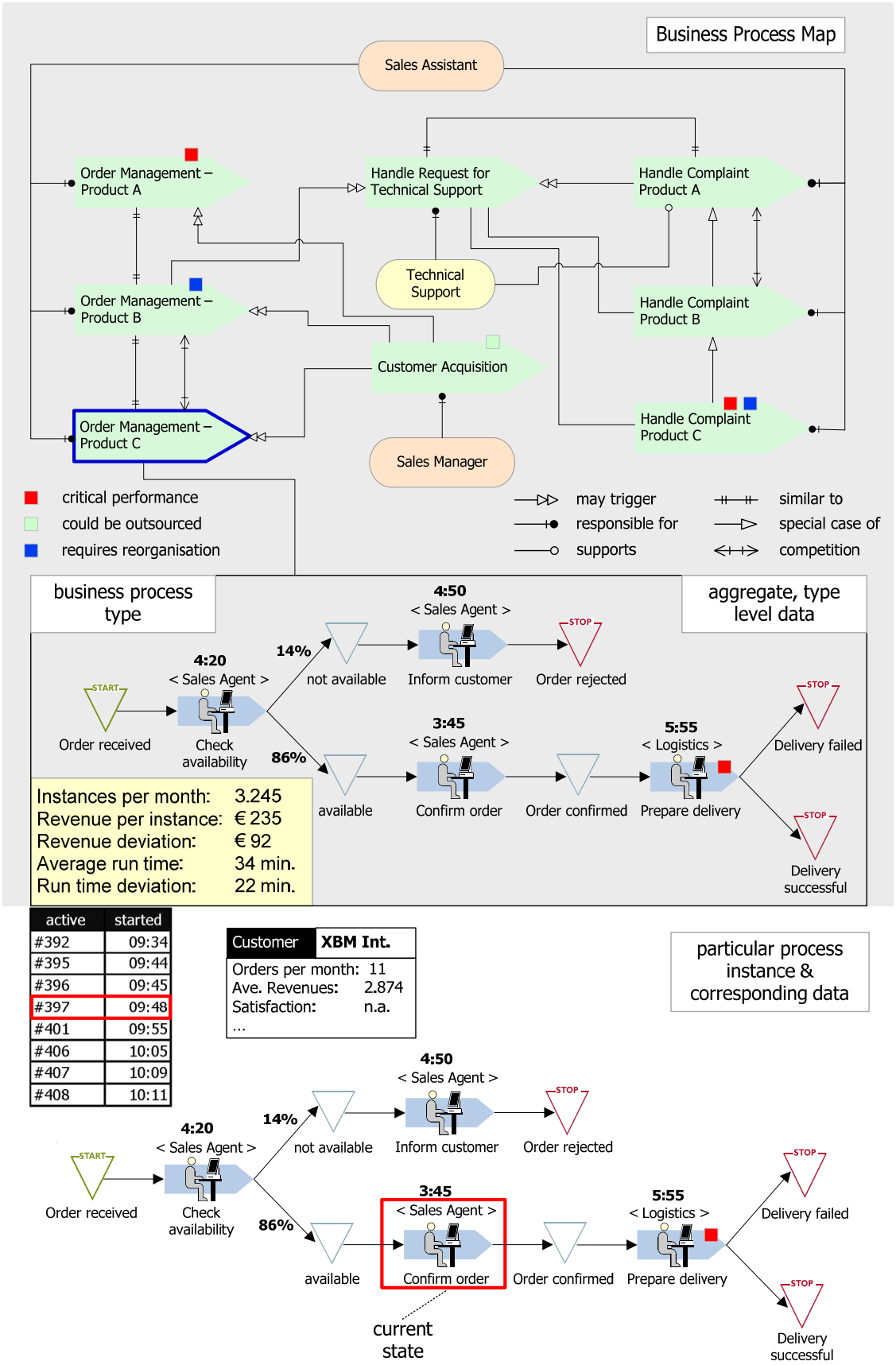


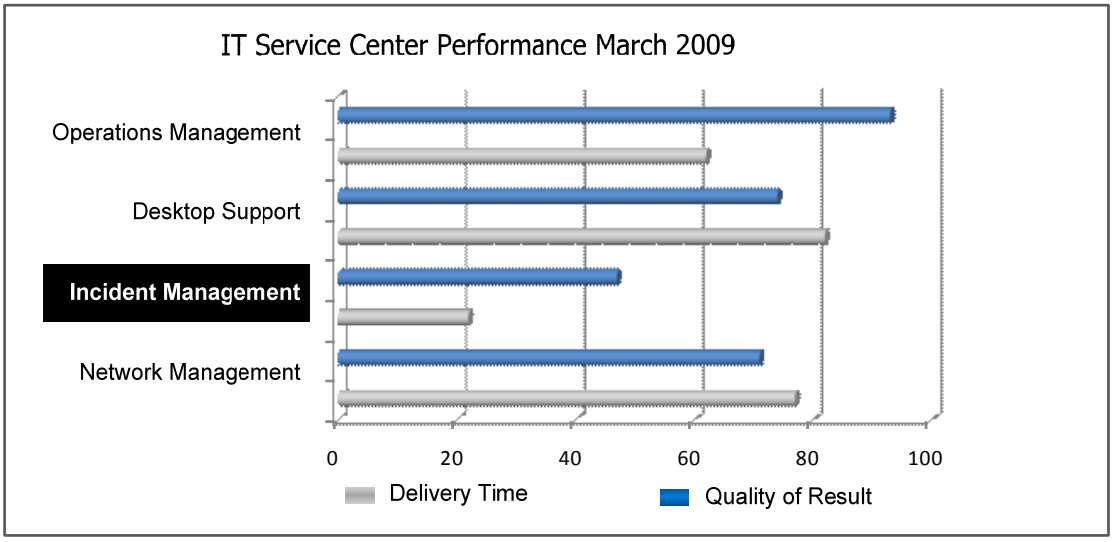
Fig. 4: Example for "drilling down" to instance level, using the same graphical representation as on the type level

A self-referential enterprise system would offer a range of (graphical) representations as views on the integrated enterprise model. Since these views were created with domain-specific modeling languages, they would prevent arbitrary modifications; would enable performing various kinds of analysis; would allow for navigating through corresponding models and for "drilling down" to instance-level data. Visual representations used for enterprise models could also be used for representing instances.

The following example is visualized in Fig. 4: A department manager, who is new to a firm, wants to obtain a better understanding of the way business is done. For this purpose, he could browse a graphical representation of the corporate business process map, which shows all business process types, their interrelationships and key performance indicators at a glance. The level of the enterprise model is marked by a grey background. Note, however, that the values of the performance indicators do not belong to the original enterprise model, but extend it with aggregated instance-level data. The concepts that define the performance indicators and the relationships between them could be represented in a further specialized model, i.e. an indicator system (Frank et al. 2008).

At this point, the department manager could select a business process type he is interested in, study the model that describes its control flow, i.e. execution, and demand for further aggregate data that describe the process, e.g. number of instances per month, average revenue per instance etc. Such an approach has obvious parallels to so-called "business intelligence" operations. Different from those, analyzing type-level data would not require elaborate – and error-prone – inductive analyses. Also, the department manager could select specific analysis views, e.g. a view that associates a selected business process type with types of the IT resources it requires and uses. If he was interested in one particular business process type, he could view the corresponding model. Then, he could leave the conceptual level and ask for the list of currently active business processes of this type, each of which could be presented in the same graphical notation as is used for the process type – enhanced by additional notational elements to represent the state of the process instance under consideration.

Upon request, the graphical representation of a business process could be supplemented with selected data – e.g. of the customer who is served by the process. It is also possible to enrich instance-level data with corresponding parts of an enterprise model. To give an example: An IT manager checks a "dashboard" that presents a graphical overview of IT service indicators as they were measured for a certain period of time. He realizes that one IT service, "incident management", presently suffers from a particularly poor performance. To get a better understanding of root causes, he zooms into the part of the IT service model of the underlying enterprise model. It shows interrelationships between IT services, the business processes that are supported by them and the responsible as well as receiving organizational units. Upon request, he can view the associated service contracts as well as the processes that implement the IT services.



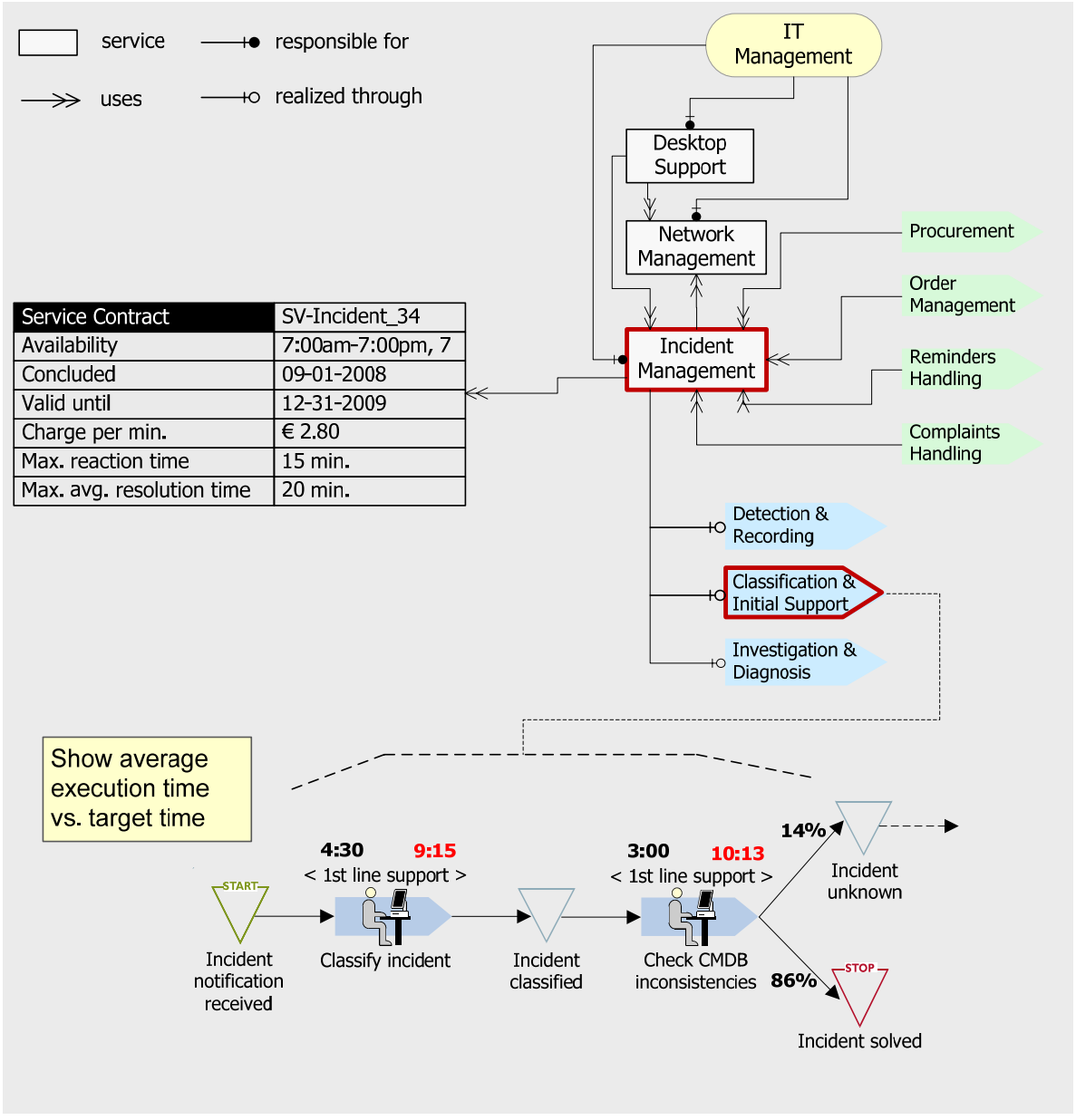


Fig. 5: Supporting the analysis of IT management indicators by providing conceptual context

When he selects the process types that realize "incident management", the system indicates that one particular process type, "classification and initial support" is performing significantly below average. Therefore, he expands the view to the corresponding business process model, which he annotates by the average execution time for the selected time period and the given target time.

The two scenarios illustrate that a SRES would not only provide a representation of an enterprise at different levels of abstraction, satisfying a range of stakeholders with situation- and goal-specific perspectives. It would also serve as a repository of methods to guide the analysis of various managerial problems. For this purpose, the various model editors would have to be supplemented by problem-specific – and adaptable – process models (in the sense of a recommended course of action), which guide users when performing certain kinds of analysis. Apparently, this feature addresses mainly requirement #4, support of analysis methods, but also #3, transparency (see Sec. 2.3).

An enterprise model represents the context of an enterprise information system at a conceptual level. Against this background, the growing relevance of miniaturized transponder technologies, e.g., RFID, enables an attractive perspective: The enterprise model could directly refer to the objects that constitute the environment it represents. For instance: If an enterprise model includes a model of a company's IT, i.e. a specification of the key concepts ("application system", "platform", "peripheral device", "operating system" etc.), the corresponding physical instances could be referred to without first (manually) representing them in an information system. In other words: Physical objects would carry a representation of their own and the enterprise model would provide the corresponding concepts, i.e. would define their semantics.

3.2.2 Integration with Language Layers

Enterprise models are usually specified by domain-specific modeling languages. In order to support the integration of the various models an enterprise model is composed of, the corresponding modeling languages should be integrated, too. For this purpose, it is advisable to specify the modeling language using a common meta language. Fig. 8 illustrates the corresponding language architecture of MEMO (Frank 2008b, p. 29f.). The meta modeling language is specified by a meta meta model, which is instantiated to the meta models that define the abstract syntax and semantics of the actual modeling languages. Note that this architecture is extensible: Additional modeling languages can be defined as further instances of the meta meta model. While such a language architecture is primarily intended as a conceptual foundation for enterprise modeling, it is a promising starting point to enrich enterprise systems that are integrated with enterprise models as well.

Making the language specification part of the system allows for a higher level of flexibility (requirement #2) and more versatile communication (#1, integration). Take the following example: A company runs a web shop that sells a wide range of products. Frequently, new

product types are added. Normally, this would require changing the corresponding schema. However, that would not be feasible, since the required recompilation would compromise the system's availability. Also, the company needs to send documents to an ever varying range of suppliers and customers. For instance: To search for suppliers of a particular product, a corresponding request – including a product type such as "Lawnmower XX" – could be disseminated on the Internet. However, that would require potential suppliers to know this type, which will often very unlikely be the case. If it was possible to describe a product type by using an adequate language, this problem could be solved. On the one hand, the language would allow for adding new product types during run-time (for a detailed description see Frank 2002a). On the other hand, it would enable prospective receivers of a message to determine the semantics of the formerly unknown product type (see Frank 2008a). Fig. 6 illustrates how an additional language layer would enable sending a new product type together with a meaningful description from one software component to another – without having a common corresponding type at first. The implementation level could gradually be adapted by including new products types, which are generated from the specifications in the domain-specific language.

It may be argued that a similar effect as the one shown in Fig. 6 could be accomplished by providing a specification of the "new" product type as a class, e.g. by using an Interface Definition Language (IDL) such as the one suggested by the OMG for CORBA. However, this would not be the same for two reasons. Firstly, a general purpose language such as the UML (and a corresponding IDL) would not allow for expressing the semantics of a product type in a way a specific product modeling language would enable it. It would not, for instance, include a concept such as "optional feature". Instead, any concept can be characterized only by referring to a few generic constructs – class, attribute etc. At the same time, a general purpose language would allow for defining products types that were apparently inconsistent – e.g. by specifying features that include a product. Secondly, a domain-specific language would enable the use of a specific concrete syntax – either graphical or textual, which would also contribute to the convenient and consistent use of the language. (Standardized) Languages that were introduced to specify electronic business documents, such as cXML (http://xml.cxml.org) or xCBL (http://www.xcbl.org/xcbl40/documentation.shtml), allow for defining specific document types. However, the product descriptions included in corresponding documents are either represented by strings only or refer to reference catalogues such as those provided by BMEcat (http://www.bmecat.org/), which in turn do not include a specification of product types, but only a structured directory of product categories. Furthermore, electronic documents are used for data transmission purposes only. They are not suited as a conceptual foundation of software systems.

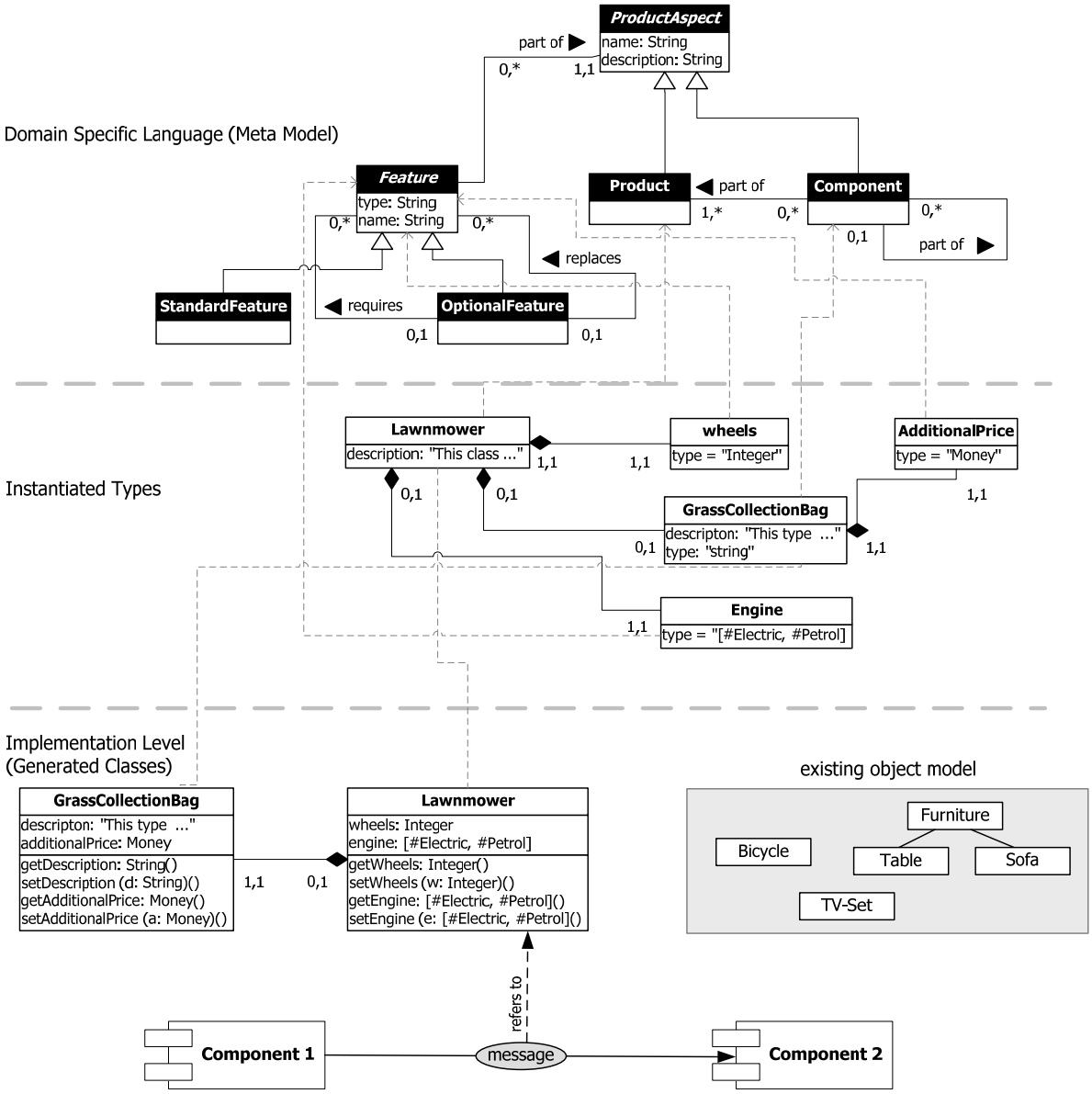


Fig. 6: Using domain specific languages to add flexibility and to foster cross-organizational integration

3.3 Outline of Basic Architecture Options

While to our knowledge the conception of a SRES is a relatively original contribution in the arena of enterprise information systems, the challenges it poses are certainly not new. They are related to a number of research problems, such as:

Language mismatch: Usually modeling languages are specified with meta languages that make use of concepts, which are different from those used for the specification of implementation languages. This results in – sometimes subtle – semantic differences, e.g. concerning the semantics of generalization/specialization (see, e.g. Frank 2003). In any case, the different specification paradigms interfere with the demand for a unified integrated representation. Vari-

ous approaches have addressed this problem, e.g. by using graphs for representing both, conceptual models and code. But so far, a satisfactory solution is not known of.

Formalization deficiencies: In addition to different meta language paradigms, integrating modeling and implementation languages is impeded by the fact that modeling languages often lack a formal specification. Therefore, representing them on a machine requires additional interpretations. While this could be done by a programmer, it can hardly be accomplished by a machine. A solution to this problem could be to "formalize" a modeling language by implementing it within a corresponding model editor. Unfortunately, such an approach may jeopardize the exchangeability of models between SRES.

Incomplete specification: Enterprise models are by definition abstractions that do not include every detail which may be required to run a program. For instance, they do not include variables that allow for representing particular instances. Also, they will usually not describe processes to a level of detail required for implementing them. This is an inherent problem because it does not address a shortcoming of enterprise models, but an intended feature.

Unbearable costs: The development of a comprehensive enterprise model that is sufficiently specified for being embedded in a self-referential enterprise system requires an effort that is beyond most companies' potential. Independent from the scientific and technical questions that still need to be answered, it is mandatory to overcome this economic obstacle. This challenge will be addressed in Sec. 4.

One could argue that decades of research in software engineering have shown that a tight integration of models and code are nothing more than a fine academic vision. However, there are a few reasons for not giving up the idea: Firstly, despite severe obstacles, its realization does not seem to be infeasible. Secondly, the vision is promising huge economic benefits – with respect to build-time productivity, information system quality and support of managerial decision making. Thirdly, previous research has already produced valuable insights into various aspects of the shown problems. A plethora of approaches focus on generating code from conceptual models (e.g. Pastor and Molina 2007; Sousa et al. 2008; Agrawal 2003) (Phung-Khac et al. 2008). Other approaches are aimed at domain-specific languages, which allow to represent frequently used code patterns and to benefit from additional abstractions (e.g. Czarnecki and Eisenecker 2000; Cazzola et al. 2004; Walter and Ebert 2009). For our purpose, approaches that are aimed at language-independent, joint representations of code and domain-specific languages (e.g. Simonyi et al. 2006) are of particular relevance.

To illustrate how the challenge of designing an architecture for SRES could be approached – and how existing research results could be factored in – we will outline four generic approaches to realize SRES, which differ with respect to synchronization and internal representation. For analytical purposes, we assume that there are two essential components that constitute a SRES: an enterprise modeling environment (EME) and the core enterprise information system (EIS), which is aimed at the management of operational-level data mainly. Note,

however, that in the ideal case, both components will not be separated. Instead, an SRES is an integrated representation of an organization and its relevant environment that supports a wide range of use cases and corresponding perspective – and contributes to integrating these perspectives. With respect to synchronization, we distinguish between one-way and mutual synchronization. Note that synchronization may involve automatic transformation processes such as generation or compilation. Any of the four alternatives allows for navigating from the EME to corresponding parts of the EIS and vice versa. One-way synchronization means that the EME allows for reading data from the core EIS in order to associate it with model elements. Whenever data that is viewed through the enterprise modeling environment gets updated, the affected views in the EME should be updated, too. For instance: After a business process instance is terminated – which would happen under the control of the core EIS – the total revenues associated with the corresponding business process type would be updated accordingly. If there was an active view in the EME that showed the corresponding value, it would be updated on the fly. Note that this would not imply changing the enterprise model itself. Mutual synchronization means that – in addition to one-way synchronization – changes applied to the enterprise model affect corresponding changes in the schema of the core EIS. If, for instance, a business process model was changed within the enterprise model, this should result in the modification of the corresponding process schema in the core EIS. With respect to representation, it is conceivable to separate the internal representation of the enterprise model from the one of the core information system – or to use a common representation for both. If there are two separate representations, it is required to integrate them through some kind of interface. Synchronization could be accomplished by using some kind of observer pattern, implemented e.g. by a model-view-controller mechanism. Fig. 7 illustrates the four generic architecture alternatives. Note that the box that represents the EME (upper quadrants) is moved a little upwards in order to express that the model schema, which serves to specify the modeling languages, is on a higher level of abstraction than the EIS schemata. The latter rather corresponds with the level of abstraction featured by the enterprise model. Some analysis tasks recommend modifying an enterprise model in order to get a better understanding of potential re-design alternatives. In this case, further instances of an enterprise model could be created that serve temporal analysis purposes only.

It may appear that the distinction between one-way and mutual synchronization is obsolete, when one makes use of a common representation. However, this is not necessarily the case. With respect to the challenges posed by changing schema information, one could decide for doing so without changing the enterprise model at run-time. Then, there would be no way for mutual integration. If the enterprise model may be changed at run-time, mutual synchronization would not require an extra step in the case of a common representation. Instead, the changes that affect both parts of the SRES would be applied only at one common place. This would be the tightest – and certainly most challenging – integration.

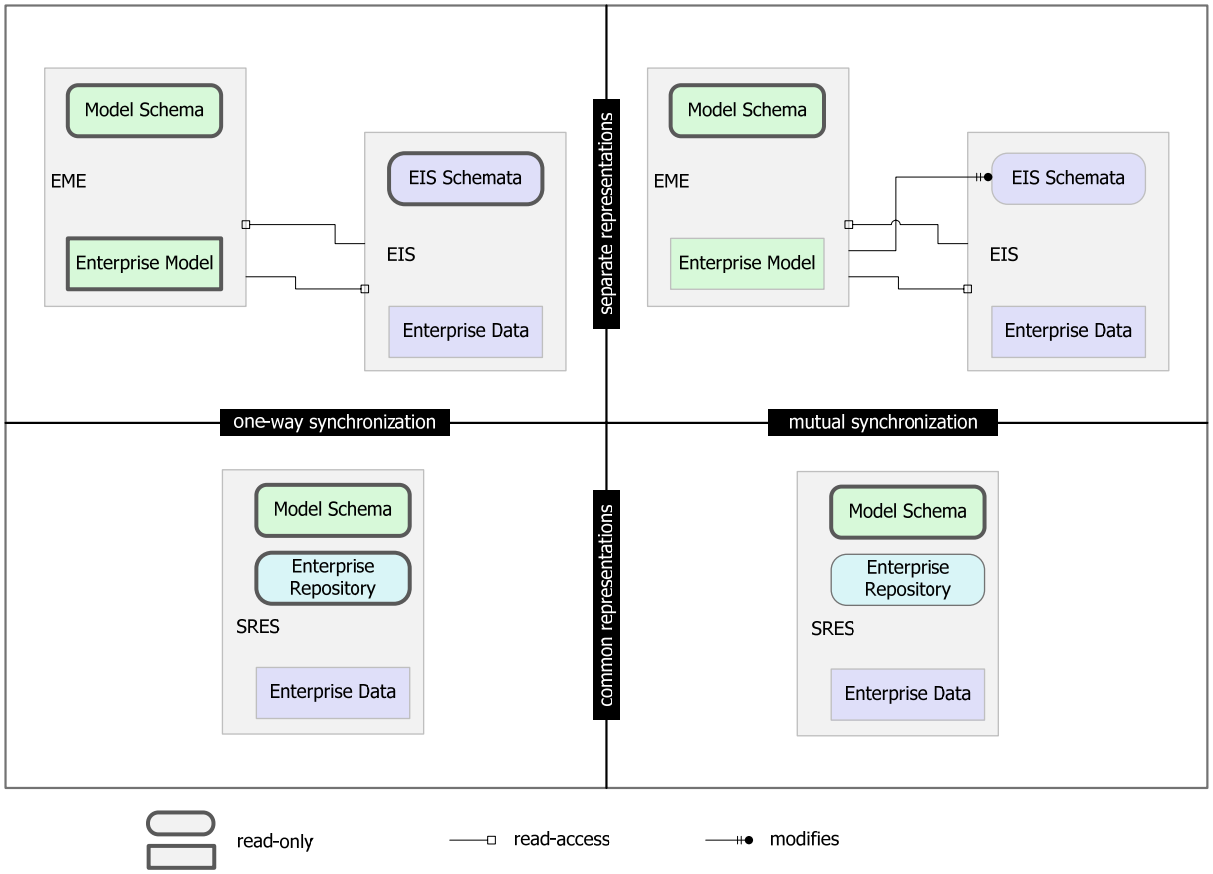


Fig. 7: Generic alternatives for SRES architectures

In order to accomplish this kind of integration, it is necessary to represent both the EIS and the EME in common conceptual models – which would then need to be implemented using a common representation, e.g. a common object-oriented execution schema. Fig. 8 provides an impression of how such a solution could be approached. It uses a language architecture – like that included in MEMO (Frank 2008b, p. 29f.) – as a versatile conceptual foundation for an SRES. The language architecture consists of a common meta meta model and an extensible set of meta models that serve to specify the various modeling languages. With respect to supporting the realization of a corresponding tool environment, both the meta meta model and the meta models can be reconstructed by using software design representations such as object models. A meta model editor could be used to specify new modeling languages – and to generate implementation-level representations for corresponding modeling tools. To accomplish this kind of integration between languages and (meta) modeling tools, MEMO Center, the MEMO modeling environment, makes use of the Eclipse Graphical Modeling Framework (Frank 2008b, p. 31f.). To promote the integration of the various models that constitute an enterprise model, the object models that were reconstructed from the meta models can be integrated into one common object model that serves as a conceptual foundation of an EME. A particular enterprise model that was created with an EME serves as a conceptual foundation for the schemata the core EIS is based on.

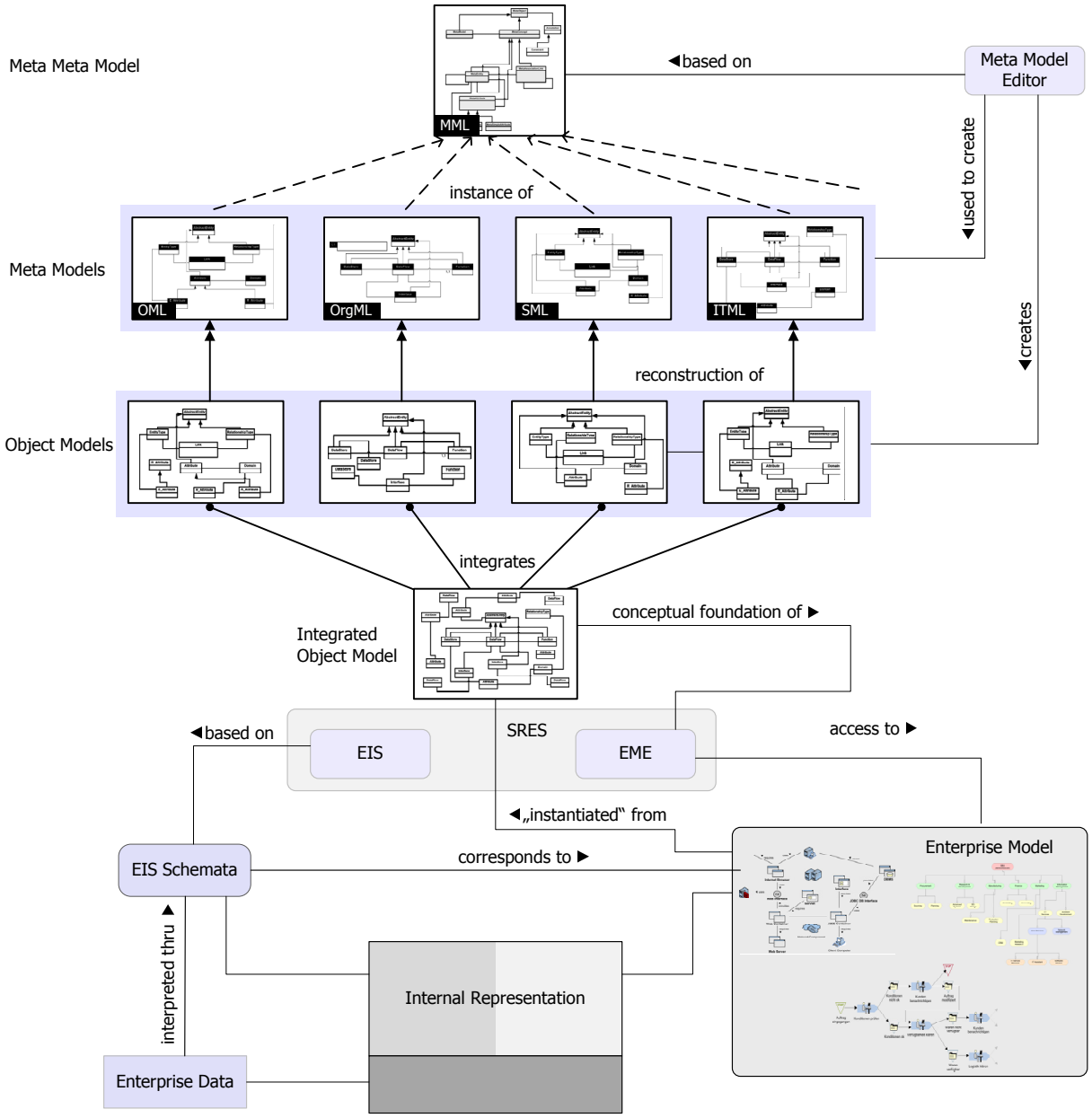


Fig. 8: Outline of an SRES architecture and its conceptual foundation

The models created and maintained by the EME represent types. However, within the modeling tools, these types, such as classes, resource types, organizational unit types etc., are usually represented by instance-level objects. Hence, they cannot be further instantiated (at least not with prevalent object-oriented programming languages). An alternative would be to represent the types by classes – which is actually done by some modeling tools. In any case, a consistent semantic mapping of the objects (or classes) that represent the enterprise model and the classes that serve as a schema for managing operational level data is a pivotal challenge. Usually, there is no simple one-to-one correspondence. Take, for instance, the representation of an organizational position type within a class schema. It is not sufficient to represent the position type as a class. Instead, it is required to also represent all constraints that constitute the semantics of the position type. In an ideal case, these constraints can be

described by a monotonic, declarative representation. In a less favorable case, the constraints would be spread in various code fragments. Hence, the box labeled "internal representation" points at a pivotal challenge. Accomplishing a common representation with overlapping views for run-time schemata and a corresponding enterprise model involves numerous problems, which still require substantial research efforts. Fig. 7 indicates that even a higher level of flexibility is conceivable: The modeling languages (referred to as "model schema" in Fig. 7) could also be subject of change, which would in turn affect corresponding models and the information system schemata.

In order to cope with the significant challenges posed by the – still to be refined – conception of SRES, it seems reasonable to follow an evolutionary research path. One could, for instance, start with separate representations and one-way synchronization (represented by the top left quadrant in Fig. 7). A corresponding prototype could be used to demonstrate the benefits, to analyze further requirements and to study design alternatives. In later stages, more ambitious architectures could be targeted. With respect to the amount of work and the range of required expertise, such a project recommends establishing larger research communities.

3.4 SRES as Open Systems

This aspect is mainly a response to requirement #5, software reuse (see Sec. 2.5). To take advantage of large scale reuse, SRES need to be open systems. According to the conception of SRES outlined above, openness does not only refer to low level interfaces. Instead, it includes the essential concepts such a system is built upon. Software reuse at the conceptual level is accomplished, e.g., by enterprise reference models that allow for reusing the domain knowledge encapsulated in its concepts to build reusable software (Frank 2006; Koch et al. 2006; Scheer 1994). Note that reference models would not only facilitate reuse. They also facilitate cross-organizational integration (requirement #1 in Sec. 2.1), because open enterprise reference models contribute to a *lingua franca* of (international) business: Data could be exchanged on a high level of semantics simply by referring to the concepts of a common language. To make the use of such a language more convenient, its concepts could be accompanied by multi-lingual denominators.

A further degree of openness would include source code as is promoted by free and open source software. In addition to the conceptual level and the implementation level, openness also applies at the instance level. If all objects that are represented in SRES were not only supplemented by the concepts that define their semantics, but also by a unique object identifier, communication could be facilitated even more – allowing for a tremendous increase of efficiency, e.g., in logistics processes.

3.5 Implications and challenges

This brief outline of a conceptual foundation for future SRES only touches upon the potentials offered by widely deployed open enterprise reference models and corresponding system architectures. At the same time, there is no doubt that realizing SRES following this conception implies formidable research challenges. They include the development of appropriate abstractions that fit a range of organizations. This is not just a matter of analyzing commonalities of today's enterprises. It also requires looking ahead and thinking beyond existing reality, e.g., taking into account possible future business models and possible future patterns of organizational action. This implies serious evaluation problems, since empirical approaches are of limited use only (Frank 2006). Furthermore, developing reference enterprise models as a foundation for future SRES requires involving researchers from relevant disciplines including Information Systems, Computer Science, and the various business- resp. management-related sub disciplines. It includes major software engineering challenges, e.g., the tight integration of conceptual models and code or mechanisms that support convenient and safe modifications. It also implies substantial challenges caused by design conflicts. Semantics, as shown, promotes the benefit of reuse, but compromises its range. Hence, the challenge is to discover/construct abstractions that cover a wide range of organizations and, at the same time, include a substantial level of semantics. A further conflict is related to the recent trend to propagate "loosely coupled" systems. While such a conception of fairly autonomous modules that are interconnected by interfaces with little semantics offers obvious advantages with respect to flexibility, it compromises integration and foregoes to take advantage of cross-module reuse. Fig. 9 provides an overview of essential requirements, related more specific requirements, and the contributions of the features provided by enterprise models. To illustrate the complexity of the subject, it shows selected relationships and conflicts.

However, dealing with the intellectual challenges sketched above is not sufficient: Developing SRES requires involving prospective users throughout all phases of the development process. It also demands resources beyond the capabilities of academic research institutions. As a consequence, one could conclude that the development of future SRES needs to be left to the major vendors of enterprise software systems. However, that would be dissatisfactory for IS research for two reasons: Firstly, SRES are at the core of Information Systems with respect to both research and teaching. They are also an important research subject for software engineering and the business- and management-related sub disciplines. Leaving the conception of future SRES to software vendors implies acting as interested observers only, which cannot be in the interest of the IS discipline. Rather, IS research should actively shape future SRES. Secondly, software vendors are likely to benefit from academic input on future SRES. On the one hand, they are often caught in the trap of eternal backward compatibility with little incentive to think about radical redesigns way beyond the usual hype cycle. In any case, they are committed to (short-term) profits, while academic research is (still) characterized by a higher level of contemplation and intellectual freedom.

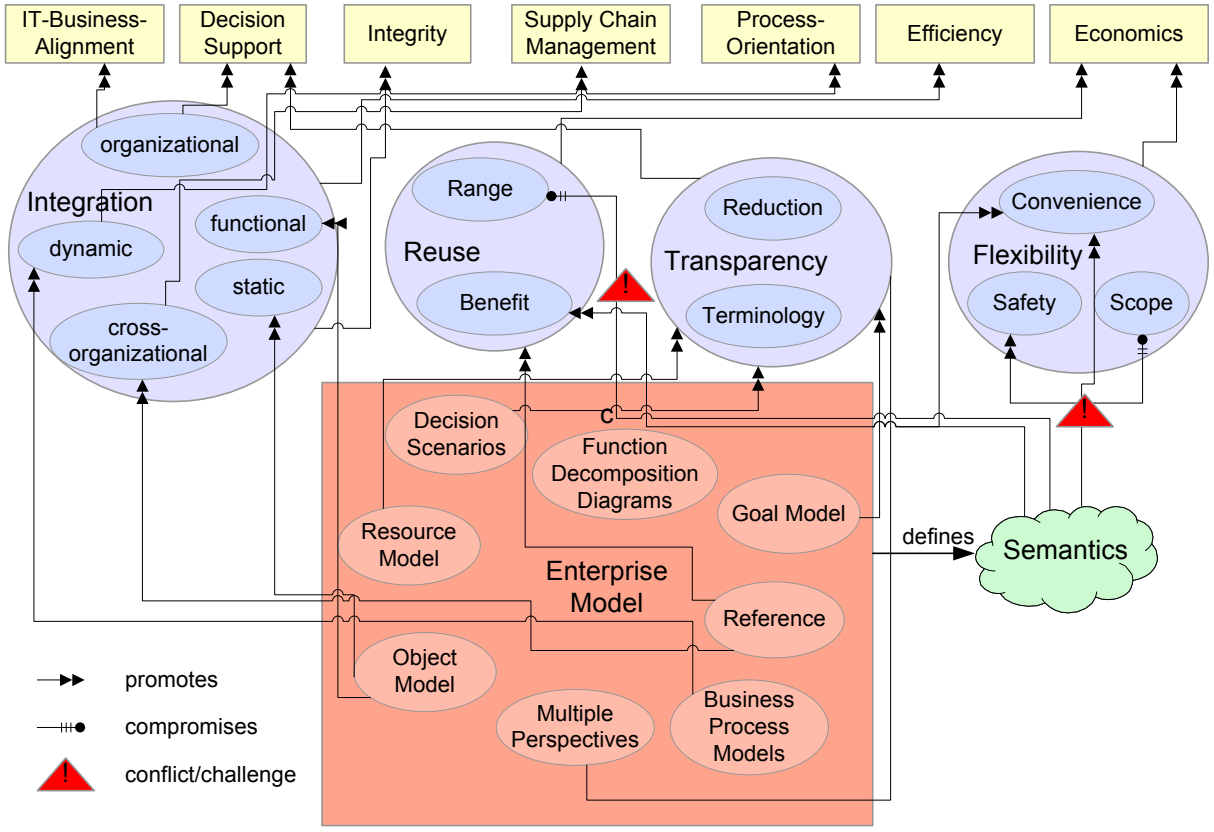


Fig. 9: The contribution of multi-perspective enterprise models to satisfy the generic requirements – selected relationships and challenges

4 Driving the Development of SRES through Open Models

Designing a conceptual foundation for future SRES as described in Sec. 3 is a complex, time-consuming, and costly endeavor that recommends joint efforts by research and practice. Restrictive licensing of enterprise reference models has in the past impeded such collaboration. The recent 'open licensing' movement has, however, prepared the ground for new forms of collaboration based on less restrictive legal provisions. One of the prime examples is free resp. open source software (FOSS) (Stallman 2002; Perens 1999). The FOSS movements have shown how collaboration in communities of geographically dispersed participants leads to complex and reliable software systems. Examples include open source enterprise systems such as tinyERP and GNUe (Dreiling et al. 2005). These systems are of a remarkable quality sometimes even challenging established commercially developed software products. The perceived successes of FOSS have, in fact, initiated an 'open movement' in several areas of society including open access and open science (Guadamuz 2005). It is, hence, obvious to question which aspects of FOSS transfer to enterprise reference modeling (Koch et al. 2006). Although FOSS is anything but a homogeneous phenomenon (Rossi 2004), a number of motivational, organizational, and technical characteristics have been identified that are shared by most 'successful' FOSS projects (Stewart and Gosain 2006). Our findings with respect to applying these characteristics to enterprise reference modeling are promising (Frank and

Strecker 2007): Licensing, organization, coordination, culture as well as evolution paths do not reveal any substantial obstacles that would suggest fundamental intransferability – considering the specific characteristics of enterprise models as opposed to source code.

For the sake of clarity, let us outline the main aspects of open (enterprise reference) models and their modeling processes. The term 'open model' refers to the licensing terms which provide the licensee with unrestricted access to all model representations and documentations as well royalty-free, non-exclusive rights to access, copy, redistribute, use, and modify the model and its documentation. It likewise applies to corresponding systems architectures. Building upon these legal provisions, we envision geographically dispersed participants to collaboratively develop open enterprise reference models and corresponding open systems architectures ('open models' for short). The community consists of researchers and students from relevant disciplines and of practitioners, e.g., from software businesses, systems users, and consulting firms. Apparently, this community-driven approach is suited to meet the challenges we identified above:

Bundling of resources: Such an open approach bundles distributed resources and facilitates to deal with tasks too demanding for individual stakeholders. Economically, one's own "manageable" contribution is rewarded with a relatively larger return through community efforts. The approach also assembles the necessary competencies: Practitioners provide, e.g., domain knowledge. Researchers add, for instance, conceptual modeling skills.

Fostering exchange between academia and business practice: The community approach involves prospective users from business and academia. The user involvement improves the acceptance and, lastly, the wide-spread use of artifacts in both research and teaching. The involvement of practitioners helps to reduce the known barrier to adopt models in practice that have been created in academia. Researchers gain access to the problems and challenges faced by practitioners and profit from their domain knowledge. Practitioners gain access to documented models and systems architectures that are free to (re-)use and modify for their own purposes. They also profit from access to academic resources and skills.

Mutual enrichment of FOSS and 'open models': The proximity to FOSS projects fosters mutual exchange with the FOSS community, e.g., with projects developing open source enterprise systems. Although the developers of these systems recognize the need for a conceptual foundation (McConnell 1999; Wilson 1999), we are not aware of any project that currently focuses on building it. The benefit is likely mutual: The OSS community can benefit significantly from the availability of open models, while the input from OSS enterprise systems projects feeds into the initiative (Koch et al. 2006).

Laboratory for Research and Teaching: The use of the conceptual foundation in teaching conveys a well-founded, solid impression of enterprise systems in practice – on a level of abstraction suited for academic institutions. In this respect, the artifacts become a laboratory for both research and teaching. This expedites transfer of research results to practice on the

basis of illustrative models and familiarizes students with the incorporated concepts. This way, future IT professionals are better qualified to work with enterprise systems.

These prospective benefits suggest that a collaborative community-driven approach based on the notion of 'open models' provides a way to build the conceptual foundation for future SRES. However, its realization is faced with a number of serious challenges. Especially the initialization of such an endeavor is a challenging task as it has to overcome several barriers. Among them, missing incentives for researchers and practitioners to participate are the most pressing issue. In the light of a high uncertainty about the quality and usability of outcomes, practitioners are presumably reluctant to join until a comprehensive set of initial open models provides a convincing proof. Researchers are confronted with the problem that their contributions may involve a significant amount of time and effort, yet the recognition of their contribution is by no means guaranteed. One way to resolve this uncertainty is to establish significant contributions as 'official', citable publications that either stand alone or supplement regular paper publications. The latter is especially important, since enterprise reference models tend to be too large to be published in their entirety – at least as journal papers. Another challenge pertains to the quality assurance of open models. Besides several approaches borrowed from FOSS such as hierarchical roles (contributor, maintainer, etc.) with distinct rights and responsibilities or voting schemes, possible solutions include review boards that review contributions upon request and accept or reject contributions very much like a journal accepts or rejects a paper.

5 The Open Model Initiative

Inspired by the exciting prospects of the described community-driven approach, we launched the Open Model Initiative (OMI) in 2006 following a well-received proposal at the Open Source Systems conference (Koch et al. 2006). The launch was accompanied by a number of papers in which we further investigate its prospects and challenges and propose specific solutions to overcome these issues (Frank and Strecker 2007; Frank et al. 2007a; Frank et al. 2007b). In Frank et al. (2007a; 2007b), we develop guidelines for the organization of the OMI and propose a process for initiating, growing, and sustaining the initiative.

As one of the initial activities, an online portal was launched at openmodels.org intended as the central hub for the open modeling community. It provides the usual means for communication such a forum. The portal also hosts a model repository where community members can access and post open models. Currently, the portal is characterized by a pragmatic design that prefers a workable solution to a sophisticated approach.

The initiative is endorsed by the Special Interest Group on Modeling Business Information Systems (SIG MoBIS) of the German Informatics Society (GI) and by most pertinent research groups in German-speaking countries. The Open Model Initiative also maintains links with related international initiatives such as the Repository for Model Driven Development

Project (ReMoDD) (France et al. 2007). At the same time, the initiative receives a remarkable though hesitant interest from the software industry. The hesitation is apparently due to the aforementioned concerns resulting in a waiting position for a convincing body of open models. Therefore, it is essential to develop a comprehensive initial open enterprise reference model as part of a respective research project. Currently, a proposal for a large-scale project that targets reference models as a foundation SRES is being prepared by the associated research groups.

The Open Model Initiative is a promising endeavour for a number of reasons: It strengthens IS research at its core and supports IS teaching. It fosters collaboration between Information Systems, Computer Science, Software Engineering and the business- and management-related sub disciplines such as organization, finance, and marketing. It can, furthermore, be seen as a new model for organizing research and disseminating research results.

6 Concluding Remarks

This report was aimed at presenting an orientation for the development of future generations of enterprise information systems, called self-referential enterprise systems. SRES are based on the idea that an enterprise information system is not just used to provide support for the operational level by managing corresponding processes and resources. Instead, it should be regarded as an integrated instrument for analyzing, planning and doing business. For this purpose, the conception of SRES suggests integrating information systems with multi-perspective enterprise models. This allows for providing business users with specialized modeling methods that support various kinds of analysis that can be linked to corresponding instance populations. Hence, an SRES is a versatile toolbox for managerial problem solving beyond today's resource management. At the same time, the conception of SRES can be regarded as an attractive laboratory for future research, because its rich semantic foundation allows for many supplementary concepts – e.g. powerful business rules, business patterns – and corresponding applications. However, despite the appealing prospects, the realization of SRES poses formidable challenges. In part, these are related to software engineering issues, such as finding a common representation for code and models – or powerful mechanisms to synchronize code and models. There are a number of research streams that address these issues. Unfortunately, there is only little cooperation between the respective communities. Bundling existing efforts should accelerate progress. While these issues are far from being solved, they are supplemented by a further problem that is not less challenging. To cope with the immense effort required to build comprehensive enterprise models and corresponding software systems, is it mandatory to accomplish economies of scales by substantial reuse of artifacts. This is important, too, to promote the inter-organizational integration of SRES in order to lower transaction cost. To address the second challenge, we suggest establishing initiatives shared by academics and practitioners that are aimed at the joint development and use of open reference models, which can be reused in a large number of cases. In addition to

that, it is required to bundle research capacity in order to develop prototypes of SRES. In the long run, the development of open reference models within open model initiatives could be supplemented by corresponding OSS initiatives.

References

Agrawal, Aditya (2003): Metamodel based model transformation language to facilitate domain specific model driven architecture. In: *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.* New York, NY, USA. ACM, pp. 118-119.

Carr, Nicholas G. (2003): IT Doesn't Matter. In: Harvard Business Review 5, pp. 41-51.

Cazzola, W.; Ghoneim, A.; Saake, G. (2004): RAMSES: a Reflective Middleware for Software Evolution. In: *RAM-SE'04 – ECOOP'04 Workshop on Reflection.* Oslo. pp. 21-29.

Czarnecki, K.; Eisenecker, U. (2000): Generative Programming: Methods, Tools, and Application. Addison-Wesley, Reading, MA.

Davenport, Thomas H. (1998): Putting the Enterprise Into The Enterprise System. In: Harvard Business Review 76 (4), pp. 121-131.

Davenport, Thomas H. (2000): The Future of Enterprise System-Enabled Organizations. In: Information Systems Frontiers 2 (2), pp. 163-180.

Dreiling, Alexander; Klaus, Helmut; Rosemann, Michael; Wyssusek, Boris (2005): Open Source Enterprise Systems: Towards a Viable Alternative. In: *Sprague, R.* (Ed.) *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38).*

Fan, Ming; Stallaert, Jan; Whinston, Andrew B. (2000): The adoption and design methodologies of component-based enterprise systems. In: European Journal of Information Systems 9 (1), pp. 25-35.

Ferstl, Otto K.; Sinz, Elmar J. (1998): SOM Modeling of Business Systems. In: *Bernus, Peter; Mertins, Kai; Schmidt, Günter* (Eds.): Handbook on Architectures of Information Systems. Springer, Berlin et al., pp. 339-358.

France, R.; Biemann, J.; Cheng, B. H. C. (2007): Repository for Model Driven Development (ReMoDD). Models in Software Engineering,Workshops and Symposia at MoDELS 2006, Genoa, Italy, October 1-6, 2006, Reports and Revised Selected Papers. Springer, Heidelberg, pp. 311-317.

Frank, Ulrich (1994): Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenbourg München.

Frank, Ulrich (2002a): Modeling Products for Versatile E-Commerce Platforms - Essential Requirements and Generic Design Alternatives. In: *Arisawa, Hiroshi; Kambayashi,*

Yahiko; Kumar, Vijay; Mayr, Heinrich C.; Hunt, Ingrid (Eds.): Conceptual Modeling for New Information System Technologies. Springer Berlin, pp. 444-456.

Frank, Ulrich (2002b): Modeling Products for Versatile E-Commerce Platforms - Essential Requirements and Generic Design Alternatives. In: *Arisawa, Hiroshi; Kambayashi, Yahiko; Kumar, Vijay; Mayr, Heinrich C.; Hunt, Ingrid* (Eds.): Conceptual Modeling for New Information System Technologies. Springer Berlin, pp. 444-456.

Frank, Ulrich (2002c): Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In: *Proceedings of the Thirty-Fifth Hawaii International Conference on System Sciences (CD/ROM).* Computer Society Press, pp. 72-81.

Frank, Ulrich (2003): Ebenen der Abstraktion und ihre Abbildung auf konzeptionelle Modelle - oder: Anmerkungen zur Semantik von Spezialisierungs- und Instanzierungsbeziehungen. In: EMISA Forum 23 (2), pp. 14-18.

Frank, Ulrich (2006): Evaluation of Reference Models. In: *Fettke, Peter; Loos, Peter* (Eds.): Reference Modeling for Business Systems Analysis. Idea Group, pp. 118-140.

Frank, Ulrich (2008a): Integration - Reflections on a Pivotal Concept for Designing and Evaluating Information Systems. In: *Kaschek, Roland; Kop, Christian; Steinberger, Claudia; Fliedl, Günther* (Eds.): Information Systems and e-Business Technologies. Springer, Berlin, pp. 11-22.

Frank, Ulrich (2008b): The MEMO Meta Modelling Language (MML) and Language Architecture. ICB Research Report, No. 24, Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen, Essen.

Frank, Ulrich; Heise, David; Kattenstroth, Heiko; Schauer, Hanno (2008): Designing and Utilising Business Indicator Systems within Enterprise Models – Outline of a Method. In: *Loos, Peter; Nüttgens, Markus; Turowski, Klaus; Werth, Dirk* (Eds.): *Modellierung betrieblicher Informationssysteme (MobIS 2008).* Bonn. GI, pp. 89-105.

Frank, Ulrich; Lange, Carola (2004): A Framework to Support the Analysis of Strategic Options for Electronic Commerce. Arbeitsberichte des Instituts für Wirtschafts- und Verwaltungsinformatik, Nr. 41, Universität Koblenz-Landau, Koblenz.

Frank, Ulrich; Lange, Carola (2007): E-MEMO: A Method to support the Development of customized Electronic Commerce Systems. In: Information Systems and E-Business Management 5 (2), pp. 93-116.

Frank, Ulrich; Strecker, Stefan (2007): Open Reference Models - Community-driven Collaboration to Promote Development and Dissemination of Reference Models. In: Enterprise Modelling and Information Systems Architectures 2 (2), pp. 32-41.

Frank, Ulrich; Strecker, Stefan; Koch, Stefan (2007a): 'Open Model' - ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung). ICB Research Report, No. 8, Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen.

Frank, Ulrich; Strecker, Stefan; Koch, Stefan (2007b): Open Model - ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik. In: *Oberweis, Andreas; Weinhard, Christof; Gimpel, Henner; Koschmider, Agnes; Pankratius, Victor; Schnizler, Björn* (Eds.): eOrganisation: Service-, Prozess-, Market-Engineering (8. Internationale Tagung Wirtschaftsinformatik). Universitätsverlag Karlsruhe, Karlsruhe, pp. 217-234.

Guadamuz, Andrés L. (2005): Open Science: Open Source Licences in Scientific Research. University of Edinburgh, School of Law, AHRC Centre for Studies in Intellectual Property and Technology Law, Edinburgh. http://ssrn.com/abstract=764064, access on 2007-02-24.

Jung, Jürgen (2004): Mapping of Business Process Models to Workflow Schemata - An Example Using MEMO-OrgML and XPDL. Arbeitsberichte des Instituts für Wirtschafts- und Verwaltungsinformatik, Nr. 47, Universität Koblenz-Landau, Koblenz, Germany.

Keyser, Chris (2007): Composite Application - The New Paradigm. In: Microsoft Architect Journal - The Architecture Journal (10). http://msdn2.microsoft.com/en-us/arcjournal/default.aspx, access on 2007-10-28.

Klaus, Helmut; Roseman, Michael; Gable, Guy G. (2000): What is ERP? In: Information Systems Frontiers 2 (2), pp. 141-162.

Koch, Stefan; Strecker, Stefan; Frank, Ulrich (2006): Conceptual Modelling as a New Entry in the Bazaar: The Open Model Approach. In: *Damiani, Ernesto; Fitzgerald, Brian; Scacchi, Walt; Scotto, Marco; Succi, Giancarlo* (Eds.): Proceedings of the The Second International Conference on Open Source Systems. Springer, Heidelberg.

Lopes, Cristina V.; Kiczales, Gregor; Mendhekar, Anurag; Maeda, Chris; Loingtier, Jean-Marc; Irwin, John (1997): Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming.

Markus, M. Lynne ; Petrie, David ; Axline, Sheryl (2000): Bucking the Trends: What the Future May Hold for ERP Packages. In: Information Systems Frontiers 2 (2), pp. 181-193.

Markus, M. Lynne; Tanis, Cornelis (2000): The Enterprise Systems Experience: From Adoption to Success. In: *Zmud, Robert W.* (Ed.) Framing the Domains of IT Research: Glimpsing the Future Through the Past. Pinnaflex Educational Resources, Inc, Cincinnati, OH, pp. 173-207.

References

McConnell, Steve (1999): Open-Source Methodology: Ready for Prime Time? In: IEEE Software 16 (4), pp. 6-8.

Møller, Charles (2005): ERP II: a conceptual framework for next-generation enterprise systems? In: Journal of Enterprise Information Management 18 (4), pp. 483-497.

Pastor, Oscar; Molina, Juan C. (2007): Model-Driven Architecture in Practice. A Software Production Environment Based on Conceptual Modeling. Springer, Berlin.

Perens, Bruce (1999): The Open Software Definition. In: *Di Bona, Chris; Ockman, Sam; Stone, Mark* (Eds.): Open Sources: Voices from the Open Source Revolution. O'Reilly, Sebastapol, CA, pp. 171-88.

Phung-Khac, An; Beugnard, Antoine; Gilliot, Jean-Marie; Segarra, Maria-Teresa (2008): Model-driven development of component-based adaptive distributed applications. In: *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing.* New York, NY, USA. ACM, pp. 2186-2191.

Popper, K. R. (1982): Logik der Forschung J.C.B. Mohr, Tübingen.

Rettig, Cynthia (2007): The Trouble with Enterprise Software. In: Sloan Management Review 49 (1), pp. 21-27.

Rossi, Maria A. (2004): Decoding the "Free/Open Source (F/OSS) Puzzle" - a Survey of Theoretical and Empirical Contributions. 424, Dipartimento di Economia Politica, Università di Siena, Siena. http://opensource.mit.edu/papers/rossi.pdf, access on 2007-02-21.

Scheer, August-Wilhelm (1992): Architecture of Integrated Information Systems : Foundations of Enterprise Modelling. Springer, Berlin.

Scheer, August-Wilhelm (1994): Business process engineering : reference models for industrial enterprises. 2nd, completely rev. and enl. ed., Springer, Berlin.

Shang, Shari; Seddon, Peter B. (2007): Managing process deficiencies with enterprise systems. In: Business Process Management Journal 13 (3), pp. 405-416.

Simonyi, Charles; Christerson, Magnus; Clifford, Shane (2006): Intentional software. In: *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications.* New York, NY, USA. ACM, pp. 451-464.

Sousa, K.; Mendonca, Hildeberto; Vanderdonckt, Jean; Rogier, Els; Vandermeulen, Joannes (2008): User interface derivation from business processes: a model-driven approach for organizational engineering. In: *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing.* New York, NY, USA. ACM, pp. 553-560.

Sprott, David (2000): Componentizing the Enterprise Application Packages. In: Communications of the ACM 43 (4), pp. 63-69.

Stallman, Richard M. (2002): Free Software, Free Society: Selected Essays of Richard M. Stallman. GNU Press, Boston.

Stewart, Katherine J.; Gosain, Sanjay (2006): The Impact of Ideology on Effectiveness in Open Source Software Development Teams. In: MIS Quarterly 30 (2), pp. 291-314.

Walter, Tobias; Ebert, Jürgen (2009): Combining DSLs and Ontologies using Metamodel Integration. In: *Proc. IFIP TC2 Working Conference on Domain-Specific Languages.* Oxford, July 15-17, 2009 (to appear).

Wilson, Greg (1999): Is the Open-Source Community Setting a Bad Example? In: IEEE Software 16 (1), pp. 23-25.

Previously published ICB - Research Reports

2009

No 30 (March 2009)
Schauer, Hanno; Wolff, Frank: Kriterien guter Wissensarbeit – Ein Vorschlag aus dem Blickwinkel der Wissenschaftstheorie (Langfassung)

No 29 (January 2009)
Benavides, David; Metzger, Andreas; Eisenecker, Ulrich (Eds.): Third International Workshop on Variability Modelling of Software-intensive Systems

2008

No 28 (December 2008)
Goedicke, Michael; Striewe, Michael; Balz, Moritz: „Computer Aided Assessments and Programming Exercises with JACK"

No 27 (December 2008)
Schauer, Carola: "Größe und Ausrichtung der Disziplin Wirtschaftsinformatik an Universitäten im deutschsprachigen Raum - Aktueller Status und Entwicklung seit 1992"

No 26 (September 2008)
Milen, Tilev; Bruno Müller-Clostermann: " CapSys: A Tool for Macroscopic Capacity Planning"

No 25 (August 2008)
Eicker, Stefan; Spies, Thorsten; Tschersich, Markus: "Einsatz von Multi-Touch beim Softwaredesign am Beispiel der CRC Card-Methode"

No 24 (August 2008)
Frank, Ulrich: "The MEMO Meta Modelling Language (MML) and Language Architecture – Revised Version"

No 23 (January 2008)
Sprenger, Jonas; Jung, Jürgen: "Enterprise Modelling in the Context of Manufacturing – Outline of an Approach Supporting Production Planning"

No 22 (January 2008)
Heymans, Patrick; Kang, Kyo-Chul; Metzger, Andreas, Pohl, Klaus (Eds.): "Second International Workshop on Variability Modelling of Software-intensive Systems"

2007

No 21 (September 2007)
Eicker, Stefan; Annett Nagel; Peter M. Schuler: "Flexibilität im Geschäftsprozess-management-Kreislauf"

No 20 (August 2007)
Blau, Holger; Eicker, Stefan; Spies, Thorsten: "Reifegradüberwachung von Software"

No 19 (June 2007)
Schauer, Carola: "Relevance and Success of IS Teaching and Research: An Analysis of the ‚Relevance Debate'

No 18 (May 2007)
Schauer, Carola: "Rekonstruktion der historischen Entwicklung der Wirtschaftsinformatik: Schritte der Institutionalisierung, Diskussion zum Status, Rahmenempfehlungen für die Lehre"

No 17 (May 2007)
Schauer, Carola; Schmeing, Tobias: "Development of IS Teaching in North-America: An Analysis of Model Curricula"

No 16 (May 2007)
Müller-Clostermann, Bruno; Tilev, Milen: "Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning"

No 15 (April 2007)
Heise, David; Schauer, Carola; Strecker, Stefan: "Informationsquellen für IT-Professionals – Analyse und Bewertung der Fachpresse aus Sicht der Wirtschaftsinformatik"

No 14 (March 2007)
Eicker, Stefan; Hegmanns, Christian; Malich, Stefan: "Auswahl von Bewertungsmethoden für Softwarearchitekturen"

No 13 (February 2007)
Eicker, Stefan; Spies, Thorsten; Kahl, Christian: "Softwarevisualisierung im Kontext serviceorientierter Architekturen"

No 12 (February 2007)
Brenner, Freimut: "Cumulative Measures of Absorbing Joint Markov Chains and an Application to Markovian Process Algebras"

No 11 (February 2007)
Kirchner, Lutz: "Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements – Grundlagen, Anforderungen und Metamodell"

No 10 (February 2007)
Schauer, Carola; Strecker, Stefan: "Vergleichende Literaturstudie aktueller einführender Lehrbücher der Wirtschaftsinformatik: Bezugsrahmen und Auswertung"

No 9 (February 2007)
Strecker, Stefan; Kuckertz, Andreas; Pawlowski, Jan M.: "Überlegungen zur Qualifizierung des wissenschaftlichen Nachwuchses: Ein Diskussionsbeitrag zur (kumulativen) Habilitation"

No 8 (February 2007)
Frank, Ulrich; Strecker, Stefan; Koch, Stefan: "Open Model - Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung)"

2006

No 7 (December 2006)
Frank, Ulrich: "Towards a Pluralistic Conception of Research Methods in Information Systems Research"

No 6 (April 2006)
Frank, Ulrich: "Evaluation von Forschung und Lehre an Universitäten – Ein Diskussionsbeitrag"

No 5 (April 2006)
Jung, Jürgen: "Supply Chains in the Context of Resource Modelling"

No 4 (February 2006)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part III – Results Wirtschaftsinformatik Discipline"

2005

No 3 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part II – Results Information Systems Discipline"

No 2 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part I – Research Objectives and Method"

No 1 (August 2005)
Lange, Carola: „Ein Bezugsrahmen zur Beschreibung von Forschungsgegenständen und -methoden in Wirtschaftsinformatik und Information Systems"

Research Group	Core Research Topics
Prof. Dr. H. H. Adelsberger Information Systems for Production and Operations Management	E-Learning, Knowledge Management, Skill-Management, Simulation, Artificial Intelligence
Prof. Dr. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. F.-D. Dorloff Procurement, Logistics and Information Management	E-Business, E-Procurement, E-Government
Prof. Dr. K. Echtle Dependability of Computing Systems	Dependability of Computing Systems
Prof. Dr. S. Eicker Information Systems and Software Engineering	Process Models, Software-Architectures
Prof. Dr. U. Frank Information Systems and Enterprise Modelling	Enterprise Modelling, Enterprise Application Integration, IT Management, Knowledge Management
Prof. Dr. M. Goedicke Specification of Software Systems	Distributed Systems, Software Components, CSCW
Prof. Dr. T. Kollmann E-Business and E-Entrepreneurship	E-Business and Information Management,E-Entrepreneurship/ E-Venture, Virtual Marketplaces and Mobile Commerce, Online-Marketing
Prof. Dr. B. Müller-Clostermann Systems Modelling	Performance Evaluation of Computer and Communication Systems, Modelling and Simulation
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
Prof. Dr.-Ing. E. Rathgeb Computer Networking Technology	Computer Networking Technology
Prof. Dr. A. Schmidt Pervasive Computing	Pervasive Computing, Uniquitous Computing, Automotive User Interfaces, Novel Interaction Technologies, Context-Aware Computing
Prof. Dr. R. Unland Data Management Systems and Knowledge Representation	Data Management, Artificial Intelligence, Software Engineering, Internet Based Teaching
Prof. Dr. S. Zelewski Institute of Production and Industrial Information Mangement	Industrial Business Processes, Innovation Management, Information Management, Economic Analyses

For more information visit us on the Web: **http://www.icb.uni-due.de**

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)