ITML: A Domain-Specific Modeling Language for Supporting Business Driven IT Management

Ulrich Frank
ulrich.frank@uni-due.de

David Heise
david.heise@uni-due.de

Heiko Kattenstroth
heiko.kattenstroth@uni-due.de

Chair of Information Systems and Enterprise Modeling
University of Duisburg-Essen
Universitaetsstr. 9, 45141 Essen, Germany

Donald F. Ferguson[a]
donald.ferguson@ca.com

Ethan Hadar[b]
ethan.hadar@ca.com

Marvin G. Waschke[c]
marvin.waschke@ca.com

CA Inc.
[a]New York, NY, USA; [b]Herzelia, Israel; [c]Washington, NY, USA

ABSTRACT

Management of today's IT is a challenging task that requires a profound understanding of both the IT landscape and the relevant business context. Numerous relations and dependencies between business and IT exist, which have to be accounted for, e.g., for better IT/business alignment. This paper presents ITML (IT domain specific Modeling Language) integrated with a comprehensive method for enterprise modeling. The language advantages are illustrated in terms of support for profound analyses, development of sophisticated IT Management tools (build-time), and use of corresponding models at run-time, e.g., as part of IT Dashboards.

Keywords

Domain-Specific Modeling Language, Enterprise Modeling, IT-Management

1. MOTIVATION AND SCOPE

IT Management is confronted with remarkable challenges. On the one hand, it is expected to serve the business with high efficiency, on the other hand it must cope with the diversity of IT platforms, networks, and information systems and their interdependencies. From a technical perspective, there is need for integrating and consistently maintaining these IT artifacts. From a managerial perspective, the transition from taylorism to process-oriented organizations as well as the growing relevance of cross-organizational business processes emphasizes the need for information systems that are not restricted to particular business functions, but that provide effective and versatile support for business processes and fulfill the business' needs ('IT/business Alignment', cf.[10, 15]). These challenges are made more difficult by language barriers between IT and business and between the different IT domains.

To cope with these challenges, methods and tools are required that support the range of IT management tasks. Existing tools and methods for IT Management are unsatisfactory in this respect. Approaches for integrating IS or managing the IT infrastructure, such as Enterprise Application Integration and Middleware (e.g., [20]) or Configuration Management Databases (CMDBs; e.g., [19]), focus on issues such as hardware and its operational metrics, e.g., address the matter of physical data exchange or management of concrete IT resources and the implementation of IT processes. Their support for elaborate technical analyses, e.g., for checking IT architectural weaknesses, or for integrating heterogeneous systems (cf. [6]) is somewhat limited, since these approaches abstract away the business context of the IS. In contrast, IT Management frameworks such as ITIL[1] or CobIT[2] present high-level guidelines for IT organization's services and processes. They provide an abstracted approach for managing IT for typical IT processes, and occasionally define metrics and key performance indicators for evaluating the quality of the operational status of the IT domains. However, there remains a gap between the IT Management and the business context on the one hand, and the detailed technical level on the other. While the gap is supposed to be overcome by IT managers, the complexity of this task suggests appropriate support – both for analysis purposes and for communicating with various stakeholders. In this respect, the motivation is twofold: First, we propose a domain-specific modeling language (DSML) for modeling IT infrastructures – the IT Modeling Language (ITML). It provides concepts for conveniently creating illustrative and consistent models of IT infrastructures, which enable various types of analyses and transformations. At the same time, it can be supplemented by corresponding process models to engineer modeling methods for IT Management. Second, the ITML is intended to support the design of tools for IT Management (build-time). We also show that ITML is useful as a versatile management instrument at run-time of these tools, e.g., by using diagrams as front end for instance data.

The ITML is part of a comprehensive method for enterprise modeling that includes various other modeling languages, e.g., an organization modeling language or a strategy modeling language. Therefore, ITML models can be supplemented by models of the relevant context in order to promote IT/business alignment and foster communication between stakeholders with different professional backgrounds.

[1]IT Infrastructure Library, [19]
[2]Control Objectives for Information and Related Technology, [11]

The remainder of the paper is structured as follows: In Section 2, the requirements for a DSML for IT Management are analyzed. In Section 3, two exemplary use cases illustrate concepts and graphical notation of the ITML and the language's benefits for IT Management. Subsequently, the conceptual background of the ITML is presented in the form of a meta-model and a language architecture. Related work is discussed in Section 5. The paper closes with an evaluation of the solution and an outlook on future work.

2. REQUIREMENTS

The following requirements analysis is aimed at preparing a foundation for the design of the ITML, and clarifies the choice between a domain-specific modeling language in general and a general-purpose modeling language (GPML).

For many planning and analyses scenarios, accounting for all resources in all details is not necessary. In fact, too much detail can obscure goals and make planning and analysis more difficult than an intelligently simplified view.

Req. 1 – Reduction of Complexity: IT Management demands for abstractions that allow for focusing on those concepts that are pivotal for certain types of analyses and application scenarios. This requires avoiding distraction caused by irrelevant technical detail. Nevertheless, ignoring technical details on principle will not be satisfactory, since some scenarios require information about concrete instances.

Ever changing and evolving technologies and corresponding "buzz words" are distinctive of the IT domain – although the basic concepts seldom change.

Req. 2 – Protection of Investment: To protect investments into models, the language concepts should neither represent technical aspects that are subject to change nor features that are specific to particular products. Note that stressing this kind of abstraction also contributes to the protection of investments into the IT itself, since it makes IT infrastructures less vulnerable against changes of variable details.

IT Management is hampered because various stakeholders, such as end-users, executives, IT experts etc., need to be involved in planning, designing, and managing IT. The language barriers between these groups may cause misunderstanding, compromising the efficiency of IT systems.

Req. 3 – Support for Multiple Perspectives: On the one hand, meaningful representations of IT at different levels of abstraction are required to satisfy the needs of the various groups of prospective users of the language. If possible, they should correspond to concepts and representations current in the prospective users' domain. On the other hand, these perspectives should be integrated to foster communication between stakeholders with different professional backgrounds.

IT is not an end in itself. Instead, it supports an organization's business processes, enterprise goals, and – in general – its competitiveness. Hence, adequate management of IT requires a profound understanding of the interdependencies between business and IT.

Req. 4 – Business Context: IT Management must not be treated as an isolated function. Instead, users should be informed of the organizational context of IT. This requires including concepts that represent the business context, e.g., strategies and goals or business processes.

IT Management still strives to integrate the multitude of different tools that are scattered over the enterprise. Data about the enterprise's IT areas (e.g., hardware, software, IT services, security, governance) are often gathered and managed separately in different information systems. This leads to independent data silos, which jeopardize data consistency (cf. [6]).

Req. 5 – Integration: To support analyses on interoperabilty of IT systems, there is need for concepts to express data or functional similarities or functions and integration deficiencies within business processes. To develop conceptual foundations for integrating heterogeneous artifacts, there is need for concepts – i.e., meta types – that can be instantiated into a range of corresponding types, e.g., different implementation of a type "Customer".

Models created with the ITML should be used to design tools for IT Management (build-time), for instance, by transforming the IT models into an enterprise-specific database schema for software to manage instances of hardware; and for providing versatile and extensible operational interfaces that can be used during business operation (run-time).

Req. 6 – Formal Foundation: The semantics of the ITML should be specified precisely enough for unambiguous transformations into implementation documents such as code.

Ostensibly, general-purpose modeling languages address the requirements. One could argue that a GPML like the 'Unified Modeling Language' (UML, [18]) or the 'Entity Relationship Models' (ERM, [2]) meet these requirements. However, such an approach has serious deficiencies (e.g., [3, 13, 16]). First, a GPML does not effectively support the construction of domain-specific models, because its syntax and semantics are designed to express any model; they are not designed to exclude inconsistent models, and thus they do not constrain users from creating – from a domain's perspective – wrong models, e.g., with 'hardware running on software' (*lack of integrity*). Second, it would be rather inconvenient to describe IT resources using only generic concepts such as 'Class', 'Attribute', or 'Association' (*lack of convenience*), which are well-suited to modeling software, but were not chosen with modeling IT in mind. Third, the graphical notation, i.e., concrete syntax of a GPML does not contribute to an illustrative visualization in the graphic idiom of IT stakeholders, such as business managers (*lack of comprehensive models*); hence, it would, if at all, provide only little support for cross-IT domain communication.

Against this background, a DSML specific for IT Management seems to be more likely to meet the preceding requirements than a GPML. In addition to the general requirements presented above, the design of a DSML is guided by the objective to reconstruct existing technical languages, in this

case the professional language of IT Management. This is for two reasons. First, it benefits from the elaborate and proven technical language of the domain instead of reinventing the wheel. Second, it makes the DSML more comprehensible, since its concepts correspond closely to terms the prospective users are familiar with – i.e., provide high semantics (cf. [6]).

3. ILLUSTRATION OF THE SOLUTION

The following scenario serves to illustrate the use of the ITML to support IT Management. The scenario is divided into two steps. First, focus is on a model of IT infrastructure that is integrated with a business process model. This step aims at indicating the potential of the modeling language to support for planning as well as 'strategic' analysis. Second, the models are extended with instance level data, which promises to support decisions dealing with concrete IT resources, e.g., a particular server or software. This requires integrating an ITML modeling tool with corresponding systems at the operational level such as CMDBs.

Figure 1 presents a model of an IT landscape (Resource/ Service and Location level), which is extended by a business perspective (Process Map). It shows various types of IT concepts such as (from top to bottom) IT services, software, diverse hardware like database systems, mainframe, mail servers, or web server, and locations (data centers). Furthermore, the elements are interrelated if they are dependent in some way, e.g., software runs on hardware and enables IT services. Such a model of the enterprise's IT already enables various analyses for IT Management. Two simplified examples are illustrated below.

Example 1 – Outsourcing: The depicted model at type level provides a foundation for outsourcing decisions. First, depending on the interencies (coupling) an IT resource type – e.g., the hardware type 'DBS 3' – has with other IT resource types, it might be a good/poor candidate for outsourcing. This is based on the following assumption: The higher the amount of interdependencies, the more complicated it will be to detach it and outsource to an external location. However, accounting only for the sheer amount of interdependencies would be an oversimplification. Rather, it is necessary to evaluate the importance of the interdependencies, i.e., the associations of a resource type. For instance, one can use the depicted models to assess the dependency between 'DBS 3' and the software type 'SAP BW'. If the association between these two types does not indicate strong coupling, and decoupling might be accomplished relatively easy, the impact of outsourcing will be less substantial.

With respect to the business perspective, the models allow for evaluating a resource's relevance for the business, e.g., by analyzing the 'business impact' of a resource in case of its breakdown/malfunction. In our example, an analysis of the associations among the modeled types reveals that the 'DBS 3' is used – to a yet unknown extent – in two services ('customer rating', 'customer contact'), which in turn support various business process types. Hence, in contrast to predominant descriptions of IT – such as records in configuration databases (e.g., a CMDB) – the model-based description, although still on type level, already indicates manifold advantages, like comprehensive analyses. The illustrative visualization further allows for inspecting a model on sight and by stakeholders that are not familiar with, e.g., data-querying languages that are necessary for analyses in database-oriented apporaches like CMDBs.

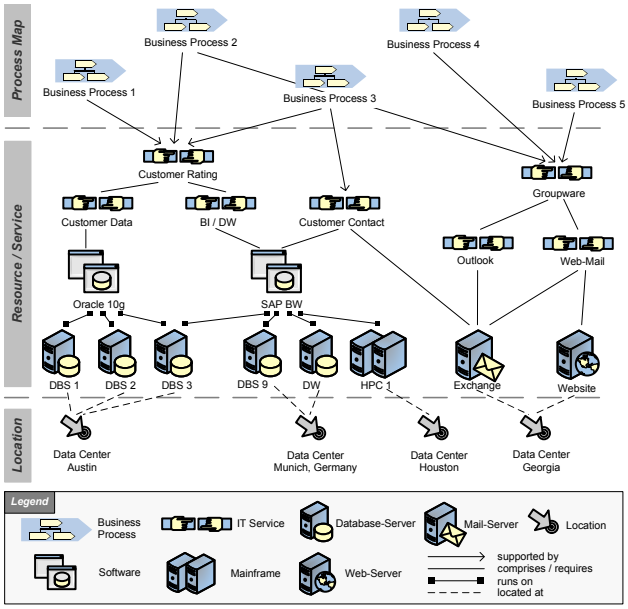

Figure 1: Exemplary Scenario

In a second step, these models can be enriched with additional information about the actual instances. Figure 2 shows the IT/business process models, in which two types, the 'business process 2' and the 'customer data' IT service, are enhanced with information about current instances. This requires that the types in the model (e.g., the IT service type 'Customer Data') are 'linked' to corresponding instances (e.g., information about actual instance of this service). Such an integration of models with corresponding instance information, i.e., the use of models at run-time, fosters a more profound decision-making and a variety of analyses than analyzing at instance level only, since information about particular instances are now enriched with the business context, while at the same time distracting complexity of the domain is still reduced. For our example, this can be applied to:

- Resource type 'DBS 3': Is there need for an upgrade in any way (e.g., based on purchase date, end of maintenance contract, number of breakdowns/incidents, costs)?
- Software types and their utilization of resource types: How frequent ly do they depend on each other (e.g., based on capacity utilization, amount of database accesses)?
- IT services: How frequent are the services accessed (e.g., charges, amount of instances)?
- Evaluating the return on investment by monitoring the IT services usage: How frequent are the services accessed (e.g., charges, amount of instances)?
- Business processes adjustments: What is the process' criticality (e.g., based on its value to customer, revenue, amount of instances)?

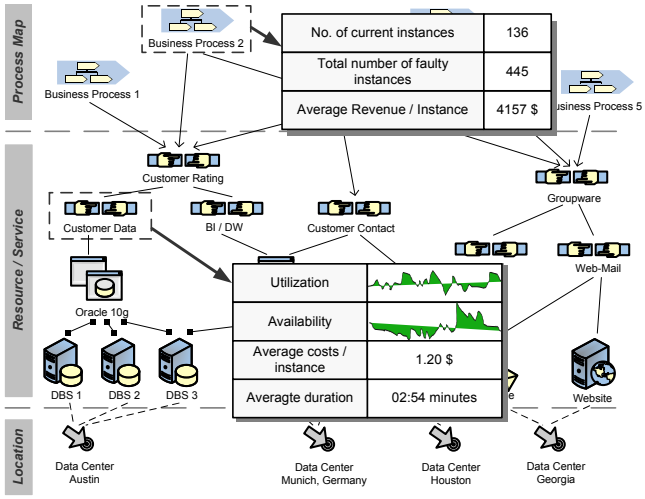

Figure 2: Exemplary Scenario enhanced with instance information

Example 2 – Consolidation/Integration: Already the development of an ITML model helps to structure the domain of interest and identify potential similarities, for instance between services offered to the business processes. Thereby, such models foster identification of candidates for consolidation and integration, e.g., of redundant data centers caused by mergers and acquisitions. If a data center offers services that are identical or closely related to services from another data center, it might be a candidate for consolidation. In many cases, such analyses still require an inspection and interpretation of the models by the users. However, depending on the analyses and application scenario tool-support might be possible, e.g., by highlighting IT services that have similar relationships; and, for instance, in contrast to querying datasets in a CMDB, it is more intuitive and comprehensive in terms of *Req. 3 & 4*. In the example illustrated in Fig. 1, the data center 'Austin' offers the service 'Customer Data' that is apparently closely related to the service 'Customer Contact' provided by data center 'Munich'. Moreover, both data centers jointly provide the service 'Customer Rating'. In order to decide about consolidating similar or related services into a single data center, the models can be enriched with information about the instances, for example, to assess the importance of the different services and accordingly of the data centers, the criticality of the underling infrastructure, and type of solution, the number of problems and tickets associated with the instance level over time, and more. Note that in decision-making usually far more information than only name and associations of, e.g., an IT service type – such as depicted in Fig. 1 & 2 – is required; in this respect, the models presented are simplifications (i.e., attributes are omitted for sake of space restrictions).

4. CONCEPTUAL BACKGROUND: META MODEL & LANGUAGE ARCHITECTURE

The DSML is specified in a meta model using the Meta Modeling Language MML (cf. [7]), which was specifically designed for specifying languages for enterprise modeling that feature a high degree of inter-language integration. The design of the DSML is guided by several objectives, driven by the requirements identified in Section 2. First, the modeling language should provide concepts that represent a reconstruction of the technical terminology of the IT domain (cf. *Req. 1*). This requires finding abstractions that closely correspond to concepts in the domain – i.e., provide a high level of semantics – in order to facilitate a comfortable use of the DSML and communication between the involved stakeholders. At the same time, the concepts of the modeling language should be rather generic in the sense that they apply to a wide range of enterprise settings and over a longer period (cf. *Req. 2*). The reconstruction also pertains to business terminology. The language has to consider concepts from the business domain that might be relevant for IT Management and provide integration between both domains (cf. *Req. 4*).

While there are various ways to structure the IT Management domain, we follow Kirchner [14], who proposes three categories of concepts for an earlier version of the ITML: *technological concepts*, such as hardware, software, network, peripherals, and so on; *organizational concepts*, which include business processes, roles/people, costs, and goals; and *additional abstractions* like IT services or information systems. Figure 3 illustrates a semantic net of the basic relations of the most prominent core concepts (cf. [14]): 'hardware' is located at a 'location' and required by 'software'. An 'information system' is an abstraction over a certain set of software and hardware. It provides 'IT services', which support 'business processes' and, in the end, contribute to the realization of the company goals and strategies. Organizational roles are related to these concepts in various ways, e.g., by means of utilization, maintenance, or responsibility, and as part of information systems (e.g., a database admin) or apart from them (e.g., help desk staff). Consequently, these core concepts constitute the foundation for the ITML language specification presented in Section 4.1.

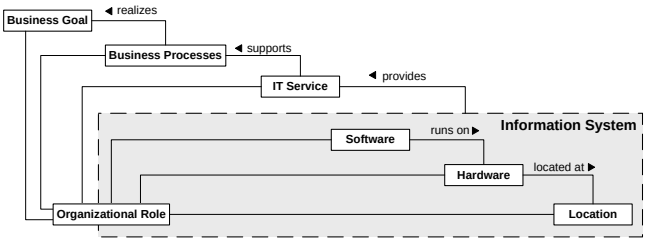

Figure 3: Core Concepts of ITML

A second design objective refers to offering a graphical notation (concrete syntax). In contrast to, e.g., textual or formal descriptions, a graphical notation supports a rich and intuitive documentation while it at the same time can depict numerous relations in a more comprehensible way. The notation has already been illustrated to some extent Section 3.

4.1 The ITML Meta Model (Excerpt)

The concepts in Figure 3 already provide a basis for the ITML meta-model. However, the specification of the modeling language still faces a number of challenges. The three most pivotal ones are discussed below. Subsequently, corresponding design decisions are presented.

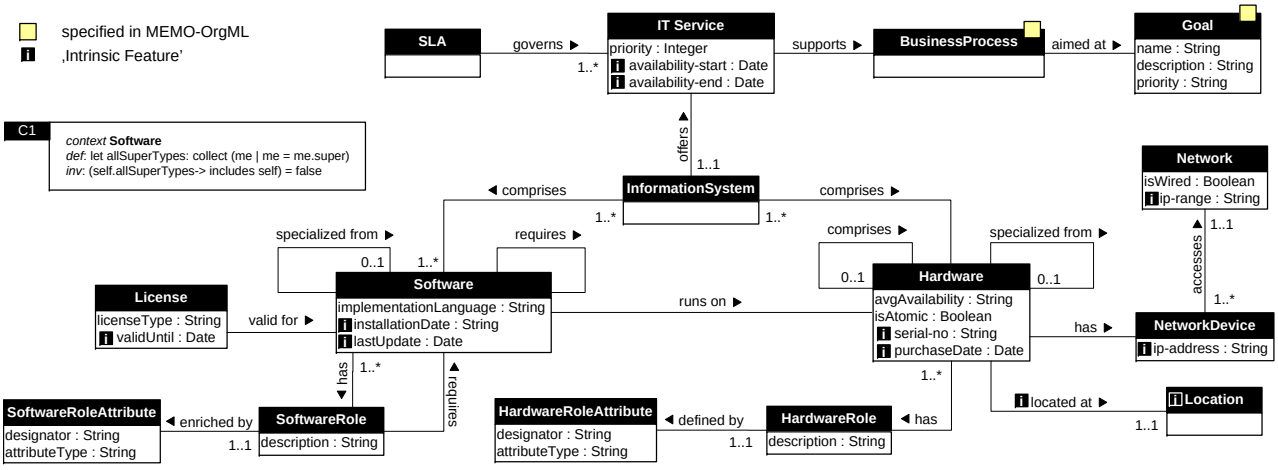

Figure 4: The ITML Meta Model (Excerpt)

Modelling Challenge A: Contingent Classification. Besides generic concepts such as 'software' or 'hardware' there exists a plethora of further refinements and characterizations for these concepts. For instance, software can be categorized with regard to its architecture (e.g., client/server), primary purpose (e.g., database management, middleware, web server), or its nature (e.g., infrastructure, application, frontend). From a modeling perspective such differentiations can be realized in different ways: In terms of specific meta types, by the use of generalization/specialization (i.e., differentiations as sub-types of 'software'), as a value of an attribute of the generic meta type 'software' (e.g., an enumeration 'type of software'), or as a role. At first glance, reconstructing all classifications as separate meta types or sub-types would conform to *Req. 1*. However, such a reconstruction would not be compliant with the demand for invariant concepts (*Req. 2*) and to an unambiguous assignment of real-world entities to concepts of the language. The concerns can be that the classification of software is often superficial and a matter of perspective – i.e., while in one decision scenario a software might count infrastructural, it can be regarded as application software in another. Though accurate in terms of technical concerns of a repeating structure, it remains conceptually different in a business context. Prominent examples are modern operating systems (usually infrastructure software) that provide integrated functionalities that count as application software; or complex application servers that provide, among others, database, middleware, and server functionalities. Furthermore, software can be assigned to several categories. The same accounts for the concept of 'hardware'. Even more, consider the convergence of devices, when multi-purpose hardware solutions such as a combined print/fax/copy machine, or a media-phone-handheld computer are introduced, classification issues are more evident

Modelling Challenge B: Interfacing to Instance Level. The DSML is designed for creating models at type level (cf. *Req. 5*); hence, the concepts in the models represent types. For design purposes, this focus is usually sufficient and necessary at the same time. However, often, it is required to differentiate between types and instances (cf. *Req. 1*). Ignoring instance information in general might generate wrongful assumptions as indicated by the above application scenarios. Therefore, the language concepts should allow for referring to instances somehow.

Modelling Challenge C: Type Differentiation. The modeling challenge pertains to the restrictions given by the type/instance dichotomy commonly applied in conceptual modeling (such as in [17]) and the semantic differences between instantiation and specialization. A discrimination of types and instances is – especially in the IT domain – not trivial, and it remains often unclear whether a real-world entity is represented as a modeling concept (i.e., a type) or as an application (i.e., an instance) of a modeling concept. Take, for instance, the meta-type 'software'. Possible type instantiations could be 'Word Processing Software', 'Microsoft Word', 'Microsoft Word 2003', or 'Microsoft Word 2003 Business Edition'. However, at the same time, 'Microsoft Word' could be regarded as an instance of a meta-type 'Word Processing Software' or as a specialization. Hardware concepts raise similar abstraction problems. For example, 'Printer' could be conceptualized as a meta-type with instantiated types such as 'Laser Printer' or 'Ink Jet Printer'. Alternatively, 'Laser Printer' could be specified as meta-type, with 'Color Laser Printer', 'HP Laser Printer XY-Series' etc. as instantiated types (cf. [5]). The decision for a certain abstraction, i.e., what is regarded as software type, as its specialisation and as its instance, variies among enterprises. If a modeling language is not flexible in this regard, it might constrain its application range or even be unsuitable for enterprises. Hence, the ITML should provide users with appropiate concepts and guidelines.

Figure 4 illustrates an excerpt of the ITML meta-model. Note that certain aspects were simplified due to space restrictions. It also shows only one exemplary OCL-constraint (*C1*), as well as '0..*'-cardinalities are omitted for reason of clarity. The meta-model illustrates the design decisions which target the above challenges:

Ad A – 'Roles': To enable users to express that a software/hardware type can be assigned to different categories, we use the concept ***'role'***. Software and hardware types are instantiated from meta-types *software* or *hardware*. To as-

sign a type a specific purpose, it can be associated to an according *softwareRole* or *hardwareRole*, which either already exists or has to be instantiated (cf. [14]). In order to reuse and extend software roles that also provide (higher) semantics, a software role can be enriched with further attributes (*SoftwareRoleAttribute*/ *HardwareRoleAttribute*), which allows for defining individual sets of software/hardware roles. While it would be possible to present a set of predefined role types by specializing *softwareRole*/*hardwareRole* in an enterprise specific language modification, our solution is more convenient because it can be used by users without meta modeling expertise – i.e., it is not necessary to adapt the meta model.

Ad B – 'Intrinsic features': There are certain apparent features of IT artifacts that we cannot express through the specification of a type only, since they are used to represent instance states (e.g., an IP address of a network device, serial number of hardware, or installation date of software). With regard to *Req. 1*, neglecting such instance features would not be satisfactory. To meet this challenge, we use the concept of 'intrinsic features' [7]. An intrinsic feature is a type, an attribute or an association that reflects a characteristic that we associate with a type that applies, however, only to the instance level. Hence, an intrinsic feature within a meta model is not instantiated at type level, but only one level further, i.e., at the instance level. In the MEMO Meta Modeling Language (MML), which is used to specify the present meta model in Fig. 4, intrinsic features are marked by an 'i' that is printed white on black (cf. [7]). A meta type that is marked as intrinsic, is actually a type (such as 'Location').

Ad C – Customized Specialisation: With respect to the restricted number of instantiation levels available for modeling, there is no perfect solution to this challenge. The ITML offers two approaches to cope with it: First, the meta-types are restricted to a few rather generic ones (such as 'Hardware', 'Computer', 'Printer' etc.– some are not shown in the excerpt). More specific types would then be created by instantiation, e.g. 'Laser Printer' from 'Printer'. Second, if there is need to create more specialized types, this can be done by making use of a 'specialized from'-relationship, which is specified for IT artifacts such as Hardware or Software. Note that the introduction of a specialization relationship implies additional constraints. These constraints are not included in the excerpt – along with further attributes and additional concepts such as (software) technical Standards, Software/Hardware Interfaces, and Organizational Roles.

4.2 Language Architecture

The integration with the business context requires to offer not only concepts that represent the IT domain, but that also account for concepts from business (cf. *Req. 4*). In the ITML meta-model the business context is represented by the meta types business process and goal. It would be inefficient to "re-invent" these modeling concepts for the ITML again, especially since they are not its primary concepts and main focus. To promote such reuse the ITML is integrated with other modeling languages for, e.g., business process or goal modeling in a way that allows for reusing concepts at the meta level and, by this, fosters the integrity of the corresponding models at the type level.

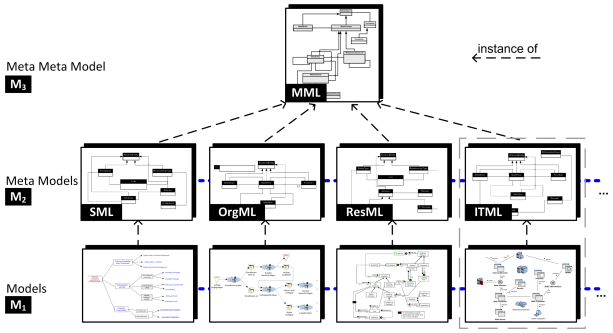

Figure 5: MEMO Architecture and the integration of ITML

For this purpose, the ITML is integrated with a method for enterprise modeling – the *multi-perspective enterprise modeling* (MEMO) method [4] – that already contains a number of domain-specific modeling languages. MEMO is multi-perspective in that it provides different groups of stakeholders with special abstractions and specific views on their relevant activities within the enterprise. Figure 5 illustrates a simplified version of the language architecture of MEMO. A more elaborate version can be found in [7]. All modeling languages within MEMO, including ITML, are specified using the MEMO Meta Modeling Language (MML, [7]) at the M_3 level. This fosters their integration since they are specified using the same modeling concepts – which allows for defining and re-using common concepts at the meta-level (M_2). This consequently leads to integrated models at type level (M_1), e.g., integrated IT and business process models. Thus, the ITML is integrated with a DSML for business processes and organizational structure (organizational modeling language, OrgML [4]), for resources (resource modeling language, ResML [12]), and for strategies and goals (strategy modeling language, SML [8]).

Concerning the use of the ITML, the integration with MEMO broadens the scope of the ITML as it is extended from an IT perspective to a more comprehensive view on an enterprise, thus fostering IT/business alignment and communication between the various enterprise stakeholders.

Note, even the excerpt in Fig. 4 might overstrain some users. Hence, the 'technical' details of modeling should be hidden from users, e.g., by a corresponding modeling tool. Furthermore, the amount of concepts that are necessary and the preferred level of detail vary between decision scenarios and the stakeholders involved. Thus, it is necessary to adapt the application of the language from case to case – which leads to the topic of 'method engineering' (cf. [1]). Method engineering is supported by MemoCenterNG[3], a modeling environment that implements the presented language architecture in Fig. 5). Thereby, it offers modeling editors for the MEMO languages, which includes an ITML modeling editor (see [7]). It also offers a meta-model editor that allows for creating further model editors. This enables experienced users to generate model editors that are based on the ITML meta-model and provide customized diagram types that, for

[3]Visit http://www.wi-inf.uni-due.de/fgfrank/memocenter-en or refer to [7] for more details.

instance, hide concepts that are irrelevant in the specific application scenario.

5. RELATED WORK

In practice, various tools for IT Management, e.g., for monitoring, network management, or IT Service Management are available, which are often based on a Configuration Management Database (CMDB) or similar databases. Such tools often allow for arbitrary models and do not provide a clear separation between different levels of abstraction, e.g., type and instance level (cf. *Req. 1–3*). Furthermore, these models mainly focus the management of instance data about IT resources and hardly account for the business context (cf. *Req. 4*).

An example of a related approach for modeling IT landscapes is the Common Information Model (CIM) published by the Distributed Management Task Force (DMTF[4]). It comprises a meta-model that defines basic concepts used for vendor-independent descriptions of IT landscapes. In this regard, the CIM solely focuses on describing IT resources and its main purpose is to use it as a schema for corresponding databases. However, since CIM does not provide any concepts from the business domain, e.g., abstractions for business processes, it neither contributes to a better IT/business alignment nor fosters communication between different stakeholders (cf. *Req. 3 & 4*).

Note, there are some modeling tools available (e.g., ARIS[5], ADOit[6]) that provide concepts for modeling IT landscapes and – to a certain extent – allow to integrate them with models of the business context. However, their language specification is usually not available. As far as we can extrapolate by examining these tools, they do not foster the development of customized tools, e.g., through code generation (cf. *Ref. 6*). Finally, these approaches and tools do not go beyond a company's boundaries – yet, there exist no mechanisms for exchanging and reusing IT models between enterprises or even within an enterprise (cf. *Req. 2 & 5*).

6. EVALUATION & FUTURE RESEARCH

In this paper, we outlined a domain-specific modeling language for IT landscapes. The language is aimed at accomplishing transparency by structuring and integrating the domain and, by this, reducing its complexity in order to support IT Management.

The language was designed to fulfill six requirements: The core concepts of the ITML have been reconstructed from the IT domain to provide abstraction that focus on relevant aspects (*Req. 1*). For this purpose, irrelevant technical details have been omitted. Thereby, focus is on invariant concepts so that efforts and investments made into IS/models are protected (*Req. 2*). This also fosters reuse of models and integration of IS (*Req. 5*). By embedding the ITML into a method for enterprise modeling, representations of IT infrastructures can be enriched with related representations. This supports not only accounting for the business context, e.g., for a better alignment with business objectives (*Req. 4*) but also facilitates the communication between stakeholders with different professional backgrounds (*Req. 3*). Finally, the semantics of the ITML allow for transforming an IT model into, e.g., a database schema for a CMDB, as well as for code generation in order to develop (integrated) software for managing IT resources (cf. [9, 13]; *Req. 6*).

Currently, the ITML primarily focuses on modeling IT infrastructures, albeit it also accounts for services and processes. Modeling of further important aspects of the IT domain, such as IT projects, is subject to future work. Moreover, we plan to refine the language specification, e.g., by further research projects with business practice, and advance the implementation of the modeling environment. This includes research on the aspects addressed, such as promoting the use of the ITML in IT Management dashboards, i.e., using models created with a DSML at run-time for advanced information systems. In this regard, the integration of instance information is a pivotal issue, which will be addressed next.

Compared to other approaches as described in Section 5, our approach shows clear advantages, mainly by featuring a higher level of semantics. Beyond satisfying the requirements discussed above, the ITML promises to supplement de-facto standards such as CobIT and ITIL by providing common concepts to describe the IT Management domain – thus fostering the integrated use of both standards. Further, the ITML serves as an instrument for bridging the gap between their high-level guidelines and the technical, i.e., more detailed view of IT Management. As highlighted in the application scenario and the language specification, the ITML features integration with the instance level, i.e., ITML models can be used to generate schemata for databases that manage instance data. Taken one-step further, this integration can be used to leverage ITML models for run-time, too. Envisioning an integration of the MEMO modeling environment with operational systems that manage the instance data (e.g., workflow management systems or the CMDB), ITML diagrams can also serve to build versatile and meaningful 'dashboards' for IT Management. Furthermore, we illustrated, that the ITML and corresponding models of the IT landscape foster comprehensive analysis and facilitate profound decision-making. Finally and with respect to *Req. 5*, the language serves as conceptual foundation for integrating information systems on a level of semantics that goes beyond all current capabilities: Information systems can describe themselves by referring to IT models extended (i.e., integrated) with business models – hence, enabling *self-referential information systems* (cf. [9]).

[4]http://www.dmtf.org/standards/cim/
[5]http://www.ids-scheer.com
[6]http://www.boc-group.com

7. REFERENCES

[1] S. Brinkkemper. Method engineering: engineering of information systems development methods and tools. *INFORM SOFTWARE TECH*, 38(4):275–280, 1996.

[2] P. P. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *ACM T DATABASE SYST*, 1(1):9–36, 1976.

[3] R. Esser and J. Janneck. A Framework for Defining Domain-Specific Visual Languages. In *Proc. of the* 1^{st} *OOPSLA Workshop on Domain-Specific Modeling (DSM'01)*, Tampa Bay, 2001.

[4] U. Frank. Multi-Perspective Enterprise Modeling

(MEMO): Conceptual Framework and Modeling Languages. In *Proc. of the* 35^{th} *Hawaii International Conference on System Sciences (HICSS-35)*. Honolulu, 2002.

[5] U. Frank. Ebenen der Abstraktion und ihre Abbildung auf konzeptionelle Modelle – oder: Anmerkungen zur Semantik von Spezialisierungs- und Instanzierungsbeziehungen. *EMISA Forum*, 23(2):14–18, 2003.

[6] U. Frank. Integration – Reflections on a Pivotal Concept for Designing and Evaluating Information Systems. In R. Kaschek, C. Kop, C. Steinberger, and G. Fliedl, editors, *Information Systems and e-Business Technologies*, pages 111–122, Berlin, Heidelberg, 2008. Springer.

[7] U. Frank. The MEMO Meta Modelling Language (MML) and Language Architecture. ICB Research Report 24, Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen, 2008.

[8] U. Frank and C. Lange. E-MEMO: a method to support the development of customized electronic commerce systems. *Inf. Syst. E-Business Management*, 5(2):93–116, 2007.

[9] U. Frank and S. Strecker. Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems. ICB Research Report 31, Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen, 2009.

[10] J. C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organisations. *IBM SYST J*, 32(1):4–16, 1993.

[11] IT Governance Institute, editor. *CobiT 4.1: Framework, Control Objectives, Management Guidelines, Maturity Models*. IT Governance Institute, Rolling Meadows, 2007.

[12] J. Jung. *Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung*. Logos, Berlin, 2007.

[13] S. Kelly and J.-P. Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley, New York, 2008.

[14] L. Kirchner. *Eine Methode zur Unterstützung des IT–Managements im Rahmen der Unternehmensmodellierung*. Logos, Berlin, 2008.

[15] J. N. Luftman, P. R. Lewis, and S. H. Oldach. Transforming the Enterprise: The Alignment of Business and Information Technology Strategies. *IBM SYST J*, 32(1):198–221, 1993.

[16] J. Luoma, S. Kelly, and J.-P. Tolvanen. Defining Domain-Specific Modeling Languages: Collected Experiences. In *Proc. of the* 4^{th} *OOPSLA Workshop on Domain-Specific Modeling (DSM'04)*, Oct 2004.

[17] Object Management Group. Meta Object Facility (MOF) Core Specification, http://www.omg.org/docs/formal/06-01-01.pdf. 2009-08-07.

[18] Object Management Group. Unified Modeling Language Infrastructure, http://www.omg.org/docs/formal/07-11-04.pdf. 2009-08-07.

[19] Office of Government Commerce, editor. *ITIL – Service Operation*. The Stationery Office, London, 2007.

[20] D. Serain. *Middleware and Enterprise Application Integration*. Springer, London, 2002.