A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering

Daniel MÉNDEZ FERNÁNDEZ, Birgit PENZENSTADLER, Marco KUHRMANN and Manfred BROY
Technische Universität München, Germany

Abstract. Requirements Engineering (RE) processes are highly volatile due to dependencies on customers' capabilities or used process models, both complicating a standardised RE process. A promising solution is given by artefact-orientation that emphasises the results rather than dictating a strict development process. At such a basis one is able to incorporate domain-specific methods for producing artefacts without having to take into account the variability of process definitions. Although artefacts are known to support customisable development processes, there still is no common agreement about the structure and semantics of artefact-based methodologies. In this paper we discuss different interpretations of the term *artefact* considering aspects like process integration capabilities and necessities within individual project environments. We contribute a meta model for artefact-orientation that is inferred from two RE models elaborated within industrial cooperation projects of our research group. We conclude with a discussion of performed case studies and ongoing work.

Keywords. artefact-orientation, model-based development, requirements engineering, meta-modelling

1. Introduction

Modelling systems during the development process has become a widely accepted paradigm in the development of software and embedded systems. One goal is, finally, to generate source code from models. Not only in such an approach, where system models are systematically derived from requirements, the precise specification of requirements is crucial. Obviously, if the requirements are to be precisely specified, the structure, syntax and semantics of requirements documentation, that capture various aspects, have to be described, too.

Typically, requirements are collected for a specific family of systems (like business information systems), so a systematic interpretation of the basic concepts that respect the needs of the application domain is required. This leads to the tendency of defining a systematic artefact-based requirements engineering (RE) process, where artefact models are used as reference models that capture the domain-specific results of the development steps. In addition to the need of capturing the basic concepts of a domain of application, early stages of development are characterised by volatile or changing project environments due to the dependency to customers' capabilities, to used process models and to produced specification documents. Also this barrier can be tackled by

artefact models since they guide the systematic elaboration of measurable results independent of the chosen development process for the artefacts, i.e. the development result is described independently of the process producing it.

In a model-based design and development paradigm, however, an artefact is seen as a structured abstraction of modelling elements used as input, output, or as an intermediate result of a process. An artefact has particular properties (structure, behaviour, etc.) and may be precisely described using standardised (semi-)formal modelling concepts. Upon such a description one is able to incorporate different techniques and notions, define clear responsibilities, and support a progress control for the production of artefacts. Hence, as the definition of an artefact model as a backbone supports a customisable process, it has become a well-accepted approach for defining development processes [11]. Internal (micro-)processes are transparent for contract-parties. They agree upon deliverables (artefacts) and do not have to care about the detailed realisation. Furthermore, if artefacts are defined for exchange, heterogeneous development environments can easily be coupled [12], also abstracting from concrete methods or tools.

However, a first step towards the definition of an artefact-based methodology for a company-wide use consists in the precise definition of the artefacts and their relations being of interest for a particular development process, in our case for RE. This guarantees that all parties in a project have the same understanding of what will be exchanged within this RE process. Once an artefact model is defined, it is obligatory for customers and all team members working on it.

The next step consists in the integration of the artefact-based RE approach into the development process by establishing the associations of requirements artefacts with development contents (process integration). A mandatory prerequisite for this integration is the definition of interfaces. Thereby, compatibility of the artefact-based RE process and the used development process has to be achieved. This means that the RE process has to be described by following the rules of the envisioned development process. Furthermore, the roles relevant for RE have to be harmonised with the roles given by the development process. This achieves finally a seamless integration of RE as an interconnected discipline to architecture, design, code, and quality assurance.

Problem Statement. So far, there is no common understanding and agreement about artefact-orientation in general and the term *artefact* in particular. There are interpretations in the granularity range from general document structures to data models defined using domain-specific languages (DSLs [3]). We believe all perspectives have to be considered in order to address the necessary process integration capabilities and at the same time precision of the artefacts produced within single projects. Although, this problem is understood, so far there exists not enough guidance to tackle it since the definition of an artefact model and the view taken strongly depends on the exact purpose of the artefact model.

Contribution. We contribute a meta model for artefact-based RE that results from lessons learned in different cooperation projects of our research group. This meta model is suitable to cover the RE domain, its usual artefacts and relations ready to be integrated into the process. Applying meta modelling on RE means to define a language (respectively an abstract syntax) to model (1) requirements artefact types, (2) artefact structures and dependencies/associations and (3) content items and content relations. Furthermore, operations can be introduced to build concrete exemplars (instances) of particular artefacts in a consistent frame and to provide capabilities for customisation and

express variability. To cover all necessary elements of a project, the RE meta model contains structure, information and semantics of not only artefacts, but also processes and roles. The meta model is compared to DSL-based modelling [3] a concrete DSL. In contrast to the multi-layered abstraction hierarchy of the Object Management Group (OMG) [8], we position the meta model according to Henderson-Sellers [28] as an abstraction of an artefact-based methodology and not as an abstraction of a modelling language for generating code (like the UML).

In general, meta models capture only basic types, therefore a domain-specific interpretation is needed to come up with concrete RE artefacts, e.g. by establishing an artefact-based methodology for the application domain of business information systems. Using a meta model, it is possible to integrate operations to support such interpretations systematically, e.g. commonalities and variabilities according to product line approaches [2], or explicit relations to enhance variability etc. However, meta models can be created in different ways: One option is to create a meta model constructively "from scratch". This option is usually used in completely new domains by developing DSLs. Another option is to infer a meta model by analysis of existing elements from given domain-specific models. In this paper we follow the second variant.

We present two RE models that have been elaborated in research projects for two different domains of applications: for the one of embedded systems and for the one of business information systems. We infer in a second step a meta model by analysing and abstracting the results and concepts of the two developed RE models. Hence, we construct a unified framework that benefits the future elaboration of artefact-based RE models. We use this contribution as a basis for tasks, such as domain-specific modelling, process integration and tool-support. We finally discuss the advantages of the approach and its applicability with respect to several performed case studies.

Outline. The remainder of this work is organised as follows: In the following Sect. 2 we describe the fundamentals and related work in the areas of our contribution. Section 3 describes two sample RE models that have been elaborated within research cooperations for two different domains of application. Both serve as a basis for analysing similarities in Sect. 4. The meta model for artefact orientation is then inferred in Sect. 5. In Sect. 6 we discuss the results for example with respect to performed case studies, before finally giving in Sect. 7 concluding remarks.

2. Foundation and Related Work

Method engineering [26] contributes important approaches for the design of domain-specific RE processes. Method engineering deals with the construction of domain-specific methods to deploy modelling techniques for the elaboration of artefacts. The selection of methods according to individual project settings is known as "situational method engineering", the integration of methods into a development process model is known under the term "method weaving". Compared to method engineering, where the focus lies on the definition, the selection and the integration of methods into a development process, artefact-orientation gives a more detailed view onto the corresponding results structure with the purpose of enabling seamless modelling of consistent results without having to take into account the variability of volatile processes and the compatibility of methods.

Still, while the area of method engineering provides with meta models a clear structure of method-based methodologies, there still is no common agreement on the structure and semantics of artefact-based methodologies. There exist different interpretations of same and similar terms, like *artefact*, *work product* or *product type*. Terms like these often co-exist in different process models, such as the referred work products that are described in the *rational unified process* or the product types that are in scope of the *V-Modell XT* [24], a German artefact-based development standard for software and systems development projects. Available approaches differ in their understanding towards artefacts and artefact models, whereby artefact models are defined considering different views (Fig. 1): structure models, generic content models, domain-specific content models, and finally integrated modelling theories.

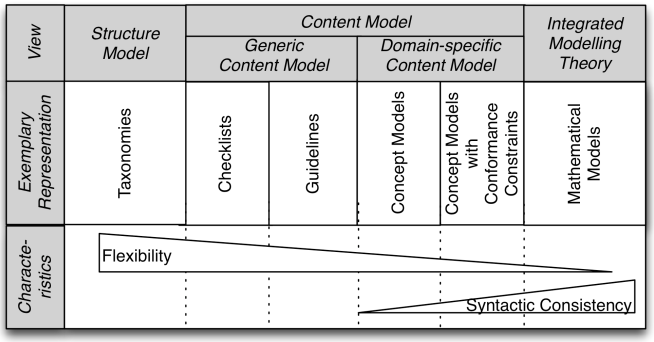

Figure 1 Variations of Artefact Models and Implications

The variations of artefact models as shown in Fig. 1 are characterised in the following:

Structure Models. Structure models include a basic description of general topics and terminology to be considered within a structure reference for documents or data sets. One example is given by the *IEEE Std. 830-1998 recommended practice for software requirements specifications* [1] that defines a taxonomy as a reference model for the construction of specification documents. Another example is given by the mentioned *V-Modell XT*, where artefacts are defined from a comprehensive process viewpoint as an abstract result set of an overall development process including for each artefact a taxonomy of general topics and content to be covered. Structure models in general emphasise the structure of the results and aim at a common, standardised understanding on what should be produced in general within individual projects.

Generic Content Models. Generic content models define contents, using for example semi-formally structured checklists. Such checklists describe independent of the chosen structure the content to be considered and give guidance for choosing exemplary methods. The *Volere requirements specification templates* [4] is an example. Still, the semantics of the contents is not precisely defined within such taxonomy-based standards. Hence, they give no guidance with respect to the elaboration of the contents and their interdependencies and fail therefore at guiding the elaboration of (syntactically) consistent artefacts. One step towards this direction in the context of RE has been made with the *requirements engineering reference model* (REM) [5] of the Technische Universität München and Siemens Corporate Research. REM defines the structure of goals, requirements and specifications within a proposed taxonomy-based guideline and infor-

mally describes dependencies between the elements of the guideline based on proposed refinement principles. REM was not developed for a specific domain of application. Existing approaches like the introduced ones use models only to establish a generic reference model of the general content and relations to be considered. Still, those models provide no concrete information about the possible concepts like the ones for constructing use case models and thereby they do not formally capture the interdependencies between the concepts like the dependency of use case models to actor descriptions. In fact, the description of concrete concepts strongly depends on a chosen domain of application what leads to domain-specific content models.

Domain-specific Content Models. The next step in the direction of formalisation are domain-specific content models that abstract from the content of documents and data sets of a particular domain of application. In such approaches, modelling concepts, including types and dependencies, are used to describe artefacts and complex artefact-based systems. Here also a formal syntax and bits of semantics are used to define artefacts' structure, their creation, and content. On this level of abstraction, (domain-specific) modelling languages can be defined [3,6,7] using, e.g., data models for their representation. These data models provide the possibility to specify the desired amount of redundancy of contents. Furthermore, one is able to establish a comprehensive tool support on this level, as data models are available for computing. Schätz describes in [18] the possibilities of additionally enriching domain-specific concept models with conformance constraints in order to guide the elaboration of RE models at project-level (as known from the area of automated model-driven approaches). Thereby, he ensures quality within those models in terms of ensuring conformance of the models being produced within a project to the reference model. In particular, the application of conformance constraints to the concept model benefits the domain-specific awareness of the results in terms of ensuring syntactic completeness and consistency in a tool-supported manner. How conformance constraints are defined, depends on the chosen quality assurance techniques. One possibility is to define the conformance constraints in terms of logic-based formalisms, as done in [9].

Integrated Modelling Theory. Mathematical models give the most formalised view onto artefact models and are often used to precise a comprehensive modelling theory. For instance, by defining properties of modelling concepts like "services" and mathematical relations to other concepts like "business processes" and, thus, they create a semantic foundation for commonly used terms. Mathematical models mostly offer their own syntax or are used as a basis for the definition of a syntax with a precise mathematical meaning (see e.g. *Focus*, [9]). Since the elements to be produced within a project are all defined according to the mathematical model, the syntactic consistency is ensured.

Summarised, the definition of an artefact model and the understanding of the term artefact strongly depends on the purpose of the model [19], e.g. concerning the coupling of different tools or concerning the definition of a flexible process model. Some of the resulting views onto an artefact exclusively focus on structure of artefacts and some (although implicating also a notion of structure) emphasise the content. Each of the given views implies going in Fig. 1 from left to right a higher degree of operationability and precision in the design of an artefact model. Precision considers the structure and the content of specifications from which the models abstract for a particular domain of application. The higher the degree of precision (as given by the latter exam-

ples), the higher the enforcement of syntactic consistency as artefact models are used as a domain-specific reference model for a particular syntax. At the same time the less the degree of flexibility. For practical usage, the establishment of domain-specific concept models has become a wide accepted technique, mostly arising from the benefit of enabling seamless modeling. Seamless modelling is enabled because such models do not only describe how a RE model is structured, but also how valid instances are created. For valid instances, all artefacts based on the defined reference model are created the same way and all artefacts of the same type have the same basic structure (e.g. each use case has at least one actor assigned, a detailed description of what this use case is good for, …). This information is a basic component to build up interfaces to other (process) disciplines and their artefacts and to provide finally tool-support in a project.

Therefore, defining artefact models from this perspective, a process-integrated, seamless and tool-supported modelling of requirements is possible, what finally is not in scope of the area of method engineering.

3. Requirements Engineering Approaches

This section describes two concrete samples of requirements engineering approaches that have been elaborated in recent research projects. We first describe the *REMsES* project considering a RE model for embedded systems and afterwards the *REMbIS* project considering the domain of business information systems. Similarities of both models are analysed in detail in Sect. 4 in order to infer a meta model for artefact orientation in Sect.5.

3.1. REMsES

The REMsES project was a research collaboration with partners from academia and industry1. Goal of the project was the elaboration of a practical guide for systematic requirements engineering and management of embedded systems, especially in the automotive domain [13]. The result is an approach with a reference artefact model (see Fig. 2). This reference model is based on two key concepts: support for abstraction levels and coverage of three content categories. The structure supports requirements engineers in determining which type of model they should use and what kind of abstractions they should create in a particular project.

Coverage of Abstraction Levels. Requirements at different levels of detail, ranging from business goals to fine-grained technical requirements (e.g. concerning the system hardware), need to be included in the requirements document. High-level requirements provide a justification for detailed requirements and support the understandability of the requirements. Low-level requirements are needed to provide enough information for implementing the system correctly.

In the *REMsES* project, a hierarchy of three abstraction levels was adopted: system level, function groups level, and HW/SW level. The abstraction levels form the vertical dimension of the structure shown in Fig. 2.

[1] The *REMsES* guide is available for free download at http://www.remses.org.

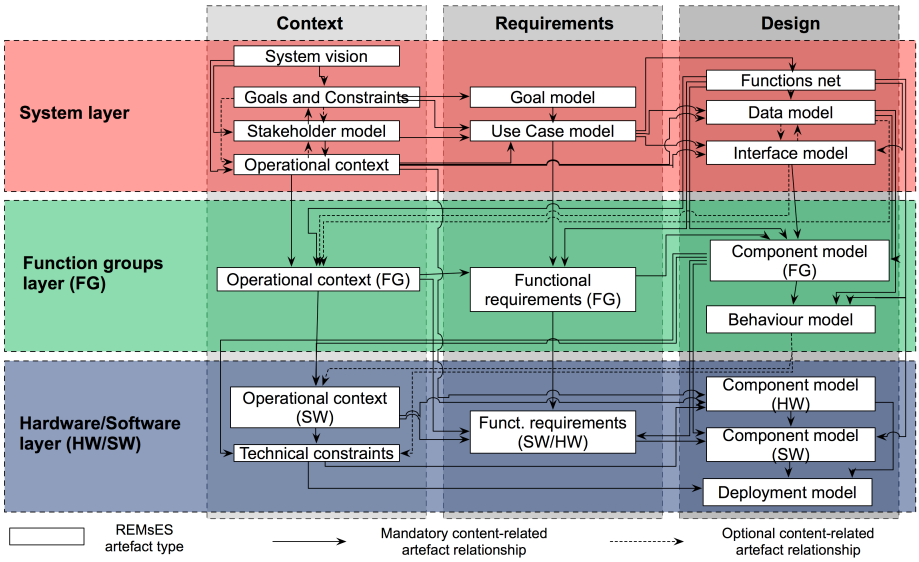

Figure 2 REMsES Artefact Model

At the *System Level*, the stakeholders have a black box view of the system. RE artefacts modelled at this levels focus on the usage of the system by human beings and other systems but no system-internal or auxiliary functions are considered.

The *Function Groups Level* represents a whitebox view onto the system. At this level, the system is viewed as a network of interacting, logical units obtained by a functional decomposition of the system. These function groups have defined interfaces and can interact with each other as well as with the system environment.

At the *Hardware/Software Level*, a coarse-grained partitioning of the system's functionality into HW and SW is defined. For this purpose, a system is decomposed into a (coarse-grained) HW topology and SW architecture.

Coverage of 3 Content Categories. By analysing RE documents in the automotive domain, three main content categories of such a document were identified: context, requirements, and (high-level) design. Therein, the categories context and design contain important information for the RE process and therefore have a considerable influence on the requirements. The content categories relate to the horizontal dimension of the structure shown in Fig. 2. The three categories are defined orthogonally to the abstraction layers.

The *Context* of the system is the part of the real world that influences the requirements for the system. Context artefacts are, for example, laws, business goals, general constraints, environmental conditions, etc. The context sets the frame within which the system is developed.

Requirements are expressed using modelling concepts. Three types of such concepts, i.e. goal models [14], scenario models [15], and function models [16], are identified as the most important models needed to support the RE process of an embedded system.

Design as the third part plays an important role. In the development of embedded systems, requirements and design are tightly intertwined, because the knowledge about major system components is inevitable to specify detailed requirements. By introducing

design explicitly as a content category, we support developers in documenting requirements and design as separate models, rather than intermingling the two.

3.2. REMbIS

REMbIS results from a research corporation between the Technische Universität München and Capgemini sd&m AG. *REMbIS* is a model-based RE approach for the application domain of business information systems. It consists of (1) an artefact abstraction model that defines horizontal abstraction and modelling views, (2) a concept model that defines those aspects dealt with during construction of models including the definition of possible notions for producing the models and finally (3) a method description that defines the activities and tasks of the RE process. Each sub-model is subsequently described.

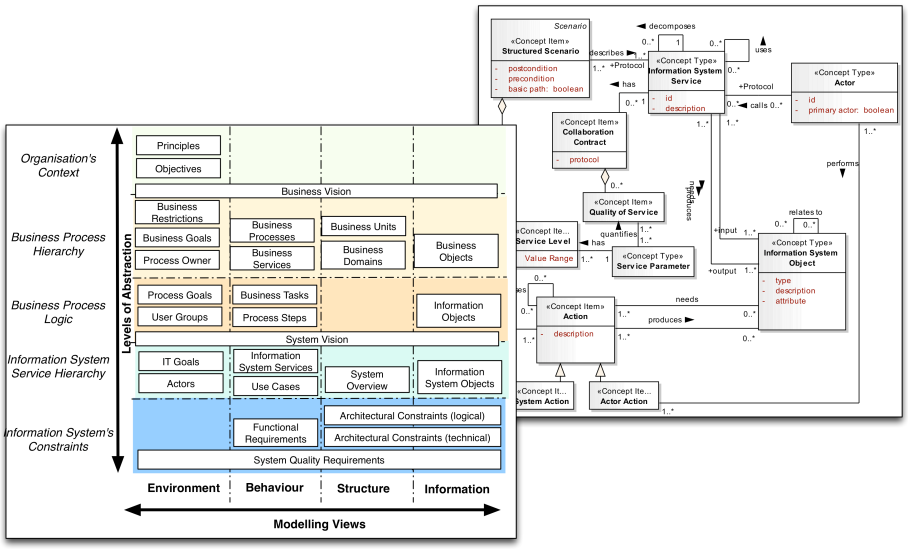

Figure 3 Excerpt of the Artefact Models and the Concept Model of REMbIS

Artefact Abstraction Model. The artefact abstraction model defines horizontal levels of abstraction that represent the stages of refinement of different RE-specific results. Modelling views structure each level according to environment, behaviour, structure and information. The levels and the views serve as a basis for traceability among the results. The left side of Fig. 3 depicts the artefact abstraction model of *REMbIS*, while embedding exemplary concepts (denoted by the boxes). The levels are divided into two major areas concerning the description of an organisation's and business structure (the upper three layers) and the compliant description of requirements for the underlying IT infrastructure (the last depicted levels).

The *organisation's context* defines the high-level, long-term steering principles and objectives of a company. The *business process hierarchy* describes the organisation's structure, including a taxonomy of the main offered business activities. Derived from this structural view, the *business process logic* emphasises the internal realisation of the activities in terms of a business process model that defines a workflow description including interchanged information and participating roles.

The *information system service hierarchy* defines the external behaviour of single systems by defining what parts of the business process model shall be supported by the system. It defines user-visible characteristics of the system by means of e.g. an (information system) service description and a use case model. The *information system's constraints* capture single quantified requirements that address the system in its applications, architecture and environment.

Finally, to define the scope of the different levels that address the business and system descriptions, *REMbIS* makes use of visions, e.g. the system vision that defines the system's context and summarizes all relevant information system services.

Concept Model. The concept model defines all elements and relations of used modelling techniques for the construction of domain-specific RE models in individual project environments (such as structure and content of use case models and their relation to service models). For each of the concepts, the approach includes a description of recommended modelling languages. The right side of Fig. 3 shows an exemplary excerpt of the concept model that defines the contents of the information system service hierarchy. The concepts shall not be discussed in detail, but illustrate how the model serves as a basis for the definition of terminology and consistency rules of all produced entities of a project.

Methods. Based on the concept model, *REMbIS* offers a description of all the tasks performed in order to use, modify and produce selected concepts, enriched with best practices that provide a set of chosen notions for constructing the concepts.

4. Analysis and Inference

Based on the two project samples described above, we analyse both approaches with respect to their similarities. This first step is aimed at abstracting from the specialities of the variants to identify similarities. Based on the results, we infer a basis for defining a meta model for artefact orientation in the following section.

Table 1. Analysis of the Example Models with respect to core Issues

Scope	REMsES	REMbIS	Superset
Domain	Embedded Systems	Business Information Systems	Generic, domain-independent approach
Refinement	Refinement over abstraction layers and content categories	Refinement over abstraction levels and modelling views	Abstraction levels and views
Results	Focus on structural aspects and notations	Concept models abstracting from description techniques and notions	Artefact types separating structure and content with syntax
Methods	Specification techniques and process close to RUP	Set of specific tasks (methods), checklists and milestones inspired by RUP	Process interface to couple methods and artefacts
Roles	Coarse-grained definition	Roles coupled to concepts and tasks	Interface to assign roles and artefacts

Table 1 illustrates a summary of both RE models, REMsES and REMbIS, which are subsequently compared according to different scopes.

Domain. The two approaches have been elaborated for different domains of application: *REMsES* for the one of embedded systems, *REMbIS* for the one of business information systems. As a consequence, the meta model shall not be described using any domain-specific aspects.

Refinement. *REMsES* and *REMbIS* both offer means to refine the results of RE over different domain-specific degrees of abstraction. For each of the described degrees of abstraction, a subset of the results is grouped according to either content category or view, where both describe common characteristics of different contents. Hence, the least common denominator is the necessity of describing vertical and horizontal abstraction for RE models.

Results. Regarding the results, *REMsES* emphasises the structure of the results and syntactical possibilities for producing these results. A major benefit of this is easy integration within process structures, because workflow entities can be coupled to cohesively described results. Instead, *REMbIS* defines a concept model based on the elements of domain-specific modelling techniques with focus on consistency among the results rather than on process integration. As a consequence, *REMbIS* has been defined in a process-agnostic manner and features an indirect overall structure that results from the dependencies between the concepts. Each content concept lays the foundation for choosing a specific syntax. The explicit distinction between structure and content enables a process integration and at the same time consistency among the results.

Methods. Both approaches offer a set of specification techniques and follow a process integration in terms of defining an abstract sequence of producing the results. In particular, *REMsES* defines concrete tasks and a coarse-grained integration into a general development process. *REMbIS* defines milestones and tasks both coupled to the results, but does not define any specific process. Hence, the meta model shall offer a description of a generic process model that allows to couple methods to results without dictating a concrete process.

Roles. Roles are defined both in *REMsES* and *REMbIS*, but *REMbIS* additionally provides their assignment to concepts and tasks.

5. Meta Model for Artefact-Oriented RE

As we identified a common superset for the two RE models, we are now able to infer a meta model that describes a language for the artefact-oriented paradigm. Fig. 4 illustrates the artefact-oriented RE meta model, which is structured into single sub-models. These sub-models define major areas of concern with a specific set of connectors (association classes) in between these areas. This modular structure fits to the aspects identified in Sec. 4 and an ease modification of the single identifies areas. We subsequently describe the single sub-models of the meta model: the artefact model, the artefact abstraction model, the generic process model and finally the generic role model.

Artefact Model. The artefact model builds the backbone by defining structure and content of domain-specific results of RE activities. The structure addresses the aspects of hierarchically ordered documents or data sets being produced during development tasks

in which single content items serve as containers for the concepts, respectively concept models. The concept models (the artefacts' content) define according to Schätz those aspects of a system under consideration dealt with during the development process [17] and reflected in the elements of the description techniques and their relations [18]. How the models are exactly defined depends on the chosen application domain from which the selection of description techniques arises (thereby often referred to as domain models). Based on these differentiating angles from which an artefact is tackled, we define an artefact as follows:

An artefact is a deliverable that is produced, modified, or used by a sequence of tasks that have value to a role. Artefacts are subject to quality assurance and version control and have a specific type. They are hierarchically structured into content items that define single areas of responsibility and that are the output of a single task. Each content item encompasses at its lowest level of decomposition:

1. Concepts: a concept defines the elements and their dependencies of domain-specific description techniques used to represent the concern of a content item. Concepts have a specific type and can be decomposed to concept items. The latter differentiation is made if different items of a concept can be described with different techniques.
2. Syntax: the syntax defines a concrete language or representation that can be chosen for a specific concept.
3. Method: The method (or task) describes the sequence of steps that is performed in order to make use of a concept.

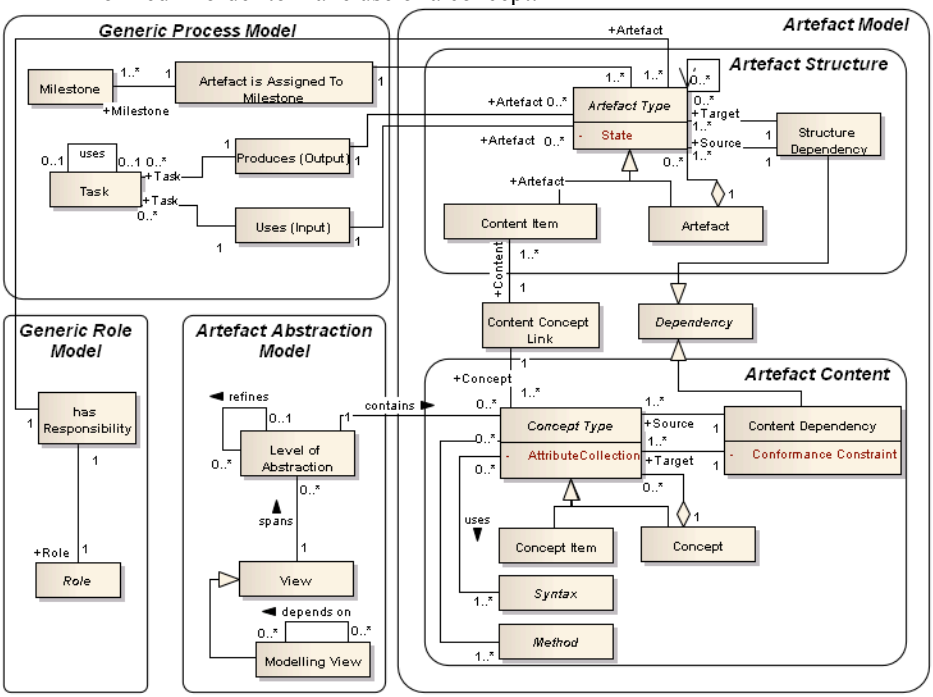

Figure 4 Meta Model for Artefact-Orientation

Note that when discussing artefacts, we usually refer to artefact types ("requirements specification") rather than to an artefact ("requirements specification 'Travel Ordering System'"). Similarly, when discussing concepts, we refer to concept types

("Use Case") instead of to a concept ("UML Activity Diagram: 'Search Hotel'"). Furthermore, each artefact has dependencies. We distinguish between content dependencies and structural dependencies. The structural dependencies are defined by composition, as an artefact type is decomposed into several content items (e.g. representing chapters within the specifications) and, finally, the dependencies that result from the content dependencies. For reasons of complexity, these dependencies are not restricted by specific types within the meta model. Hence, as the content dependencies are not typed, the domain- specific instances need a definition of conformance constraints. One possibility for the definition of such constraints is given by the use of the Object Constraint Language (OCL) for supporting the conformance of the models being created in a project to the reference models in a tool-supported manner. Note that the conformance of domain-specific instances to the meta model at hand is given, if these are described in the language defined by the meta model.

Artefact Abstraction Model. The artefact abstraction model defines the domain-specific abstraction that refers to the construction and refinement (and / or decomposition) of the artefacts. We distinguish between (1) horizontal abstraction, that reflect aspects like structure, behaviour or roles and referred to as modelling views [20] and (2) vertical abstraction that describes, for each of the views, the refinement and / or decomposition of corresponding artefacts, also referred to as levels of abstraction [21,22]. These levels of abstraction have specific characteristics that match with the degree of detail of an artefact's underlying concept and its concern.

Generic Process Model. The generic process model defines all entities of a workflow description that can be used to define a specific process. In particular, we define (1) milestones in terms of a point in time in which an artefact or content item should be completed and (2) tasks that build a part of phases and activities. A phase (e.g. requirements engineering) defines a repeatable set of activities which, in turn, define major areas of concerns (e.g. requirements elicitation). For such an area of concern, there exist several tasks where each consists of a sequence of atomic steps that are performed by a role in order to produce, modify and / or use an artefact as input or output. Hence, a task is a method in the sense as it is defined in the area of method engineering [23]. A task supports the selection of which artefacts are needed as basis for modifying or producing further artefacts by the choice of a specific syntax. The generic process model also describes the process interface to be used for integrating the RE meta model into a development process model. Model elements such as milestones are underspecified, as the precise descriptions have to be delivered by the process model.

Generic Role Model. The generic role model describes a logical representation of the individuals who directly participate in or indirectly contribute to the development process. A role has the responsibility for at least one artefact while performing a set of activities. Similar to the process model, the generic role model is underspecified, as the instance role model has to be provided by the development process.

6. Discussion

With the meta model, presented in the previous section, we define a language for artefact-oriented RE. This language provides applicable guidelines for domain-specific RE

models that support consistency and completeness within the results across different projects, continuity within the process and variability in the process definitions.

Syntactic Consistency and Completeness. Using the proposed language for the description of domain-specific RE models supports syntactic and semantic consistency and completeness among the results of RE. Syntactic consistency and completeness include the restriction of possible description techniques (by defining concept models) and thereby lay the foundation for consistency and completeness conditions with respect to the choice of a syntax. In explicit, the RE model answers the question "How is the syntax of artefact A related to the syntax of artefact B?". Semantic consistency means consistency between project-specific contents, i.e. it provides an answer to the question "Does the statement of artefact A fit to what the model of artefact B declares?". This can not be handled by an artefact model, but verification of semantic consistency requires syntactical consistency.

Continuity. Artefact-orientation fundamentally enables continuity within the development process. As the concept model provides the basis for seamless modelling, the process that is coupled to the artefact model provides a continuous workflow when elaborating the artefacts.

Variability supporting Customisation. Artefact-orientation tackles the problem of variability in designing a process within volatile environments. The meta model offers a modular structure and supports process integration capabilities, i.e. the customisation of artefact-based RE models at organisational levels by integrating them into a development processes of particular companies. This process integration can for example be performed with the *V-Modell XT* [24], a meta model-based German standard for software & systems development, by integrating the single elements over the association classes into the organisation-specific process models (see also Sect. 5). In general, the connections that are given by several association classes state what connection points have to be available (obviously, for integrating the RE meta model, a meta model-based development process model is a mandatory prerequisite). The artefact-centric view unifies the definition of taxonomy-based structures of documents or data sets and the definition of domain-specific modelling concepts (content). In a second step, we couple all elements of a process model to the artefact model. Therefore, while method-based methodologies face problems with respect to syntactic variability during the integration of new methods, artefact-oriented approaches support a systematic integration of methods by innately restricting the results' contents and therefore indirectly the possibilities of selecting methods producing them. As each domain-specific RE model, derived from the meta model, exhibits the same modular structure, it satisfies in addition to needed process integration capabilities the needs of customisation at project level. Therefore, it serves as a reference model that guides through the elaboration of the results of RE being compliant to the RE model. Moreover, it can be customised according to varying project parameters that strongly affect the elaboration of single artefacts. Therefore, it tackles the deficiencies of process-driven approaches.

Trade-Off between Precision and Flexibility. When establishing an artefact-based approach supporting an efficient and effective RE process, there is a trade-off between flexibility and precision: The more options are provided for syntax or concepts, the higher is the variability with respect to the relations between artefacts and the more

complex is the verification of consistency. The higher the restrictions for syntax, the more suffers the flexibility for applying the models at variable project-levels due to making knowledge explicit. Referring to Fig. 1, there is a conflict between a general and a precise view. The more precision there is in the chosen way of defining an artefact model, the more capabilities there are in gaining consistency and completeness within the models considered compliant to the artefact models. Following the provided meta model of Sect. 5, artefact models perform therefore a unique combination of two views, the view onto content and dependencies via concept models embedded into a second taxonomy-based structure view. Hence, the necessary degree of flexibility is ensured while the degree of variability when constructing and representing the content decreases since we still have restrictions on the content and the used syntax.

Evaluation in Case Studies. The introduced meta model is suitable to cover the stated needs since it abstracts from two concrete RE models that both have been evaluated in industry. The *REMsES* approach was evaluated in case studies, student experiments, and pilot projects at the sites of the industrial partners (see also [27]). The *REMbIS* approach is fully integrated into the development process of Capgemini sd&m AG. Their engineers have been trained on *REMbIS* in seminars and constantly apply it in custom software development projects in which an evaluation can not be published for reasons of confidentiality.
The meta model was the basis for the development of *BISA* (artefact-based approach for Business Information Systems' Analysis) that offers customisation capabilities for process integration and for project-specific customisation. The latter is performed by customising the artefact model according to project parameters that influence the possibilities and necessities of elaborating single artefacts to which the parameters are coupled (see also [29]). The process integration has been performed with the V-Modell XT, a meta model based standard process including a comprehensive tool chain. The resulting integrated, customisable *VM BISA* [25] has successfully been applied in a project hosted by Siemens.

7. Conclusion

This paper concentrates on the paradigm artefact-orientation in requirements engineering and presents a meta model. This meta model is inferred from two concrete domain-specific RE models of our research group: one for the application domain of embedded systems and one for the application domain of business information systems. Both RE models have been evaluated in real-life projects and thereby validated with respect to applicability.

We outlined the advantages of artefact-orientation: Taking an artefact model as backbone of project execution benefits (1) syntactic consistency and completeness of the results being compliant to the domain-specific reference model, (2) seamless modelling of the results and continuity within the development process chain and, (3) can be customised to individual needs. Such a customisation can be performed at organisational level in terms of process integration and at project level by adapting domain-specific artefact models according to the influences of individual project environments. The integration into a comprehensive process framework and the definition of a customisation approach considering the project level has been performed in [25] and successfully applied in a pilot project hosted by Siemens.

References

[1] IEEE: IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998, 1998.
[2] Rombach, D.: Integrated software process and product lines. In Li, M., Boehm, B., Osterweil, L., eds.: International Software Process Workshop. Lecture Notes in Computer Science 83–90, 2005.
[3] Greenfield, J., Short, K.: Software Factories. Wiley and Sons, 2004.
[4] Robertson, J., Robertson, S.: Volere Requirements Specification Templates-Edition 11. Available Online at http://www.volere.co.uk, August, 2007.
[5] Geisberger, E., Broy, M., Berenbach, B., Kazmeier, J., Paulish, D., Rudorfer, A.: Requirements engineering reference model. Technischer Bericht, Technische Universität München, 2006.
[6] Gronback, R.: Eclipse Modeling Project. Addison-Wesley, 2009.
[7] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF -Eclipse Modeling Framework. Addison-Wesley, 2009.
[8] OMG: Unified modelling language (uml) -infrastructure and superstructure 2.1.1. Specification, Object Management Group, 2007.
[9] Schätz, B.: The odl operation definition language and the autofocus/quest application framewoek aqua. Technical report, Technische Universität München, 2001.
[10] Ameluxen, C., Rötschke, T., Schürr, A.: Graph transformations with mof 2.0. In: 3rd International Fujaba Days, 2005.
[11] Managing Successful Projects with Prince 2. Office of Government and Commerce, 2009.
[12] Kuhrmann, M., Kalus, G., Chroust, G.: Tool-Support for Software Development Processes. In: Social, Managerial, and Organizational Dimensions of Enterprise Information Systems. IGI Global, 2009.
[13] Penzenstadler, B., Sikora, E., Pohl, K.: Guiding requirements modelling in the embedded systems domain with an artefact reference model. In: REFSQ, 2009.
[14] van Lamsweerde, A.: Requirements engineering: From craft to discipline. In: SIG¬SOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, USA, ACM 238–249, 2008.
[15] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.
[16] Davis, A.M.: Software requirements: objects, functions, and states. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
[17] Schätz, B., Pretschner, A., Huber, F., Philipps, J.: Model-Based Development of Embedded Systems. In: Advances in Object-Oriented Information Systems, Lecture Notes in Computer Science. Volume 2426., Springer-Verlag 298– 311, 2002.
[18] Schätz, B.: Model-Based Development of Software Systems: From Models to Tools. Habilitation, Technische Universität München, 2008.
[19] Endres, A., Rombach, H.: A handbook of software and systems engineering: empirical observations, laws and theories. Addison-Wesley, 2003.
[20] Broy, M., Schätz, B., Wild, D., Feilkas, M., Hartmann, J., Gruler, A., Penzenstadler, B., Grünbauer, J., Harhurin, A.: Umfassendes Architekturmodell für das Engineering eingebetteter software-intensiver Systeme. Technical Report TUM-I0816, Technische Universität München, 2008.
[21] Gorschek, T., Wohlin, C.: Requirements abstraction model. Requir. Eng. 11(1) 79–101, 2005.
[22] Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Trans. Softw. Eng. 27(1) 58–93, 2001.
[23] Braun, C., Wortmann, F., Hafner, M., Winter, R.: Method construction -a core approach to organizational engineering. In: SAC, ACM 1295–1299, 2005.
[24] V-Modell XT. http://v-modell-xt.de/, 2008.
[25] Mendez Fernandez, D., Kuhrmann, M.: Artefact-based requirements engineering and its integration into a process framework. Technical Report, Technische Universität München, 2009.
[26] Brinkkemper, S., Joosten, S.: Method Engineering and Meta-Modelling. Special Issue. Information and Software Technology, 1996
[27] Braun, P., Broy, M., Houdek, F. Kirchmayr, M., Müller, M., Penzenstadler, B,, Pohl, K., Weyer, T.: Entwicklung eines Leitfadens für das Requirements Engineering softwareintensiver Eingebetteter Systeme. Technical Report. Technische Universität München, 2009.
[28] Gonzalez-Perez, C. and Henderson-Sellers, B.: A powertype-based metamodelling framework. Software and Systems Modeling, Vol. 5, pp. 72-90, 2006.
[29] Mendez Fernandez, D. Wagner, S., Lochmann, K., Baumann, A.: Field Study on Requirements Engineering Artefacts and Patterns. Proceedings of 14th International Conference on Evaluation and Assessment in Software Engineering, Staffordshire, United Kingdom, 2010.