An Artifact-oriented Framework for the Seamless Development of Embedded Systems

Wolfgang Böhm, Andreas Vogelsang
Technische Universität München
Institut für Informatik
Boltzmannstr. 3
85748 Garching b. München
{boehmw,vogelsan}@in.tum.de

Abstract: Transferring novel modeling concepts and approaches into a well established and customized industrial context is not easy. They have to be mapped to the specific development process of the application domain, must complement the existing tools, and exhibit certain representations. Artifact-oriented development distinguishes between the development process and the created artifacts in the context of a given development project. This paper provides a conceptual framework that encompasses an artifact-oriented view onto the development of embedded systems. We argue that this artifact-oriented view provides means to map academic models and description techniques onto existing development processes in industry. It furthermore provides the basis for the definition of tracing links and dependencies between the different contents and artifacts, allowing for a seamless development of artifacts.

1 Artifact-oriented Development

Over the years, a number of methods, processes, description techniques, and models have been proposed by academia in order to enhance the development of embedded systems. On the other side, there exists a plethora of well-established tools, development processes and best practices applied in industry. Therefore, the transfer of new ideas and approaches into a well-established and customized industrial context is not easy. Artifact-oriented development distinguishes between the development process, which might be very specific, and the created artifacts in the context of a given development project. It specifically aims at a detailed description of the structure, the content and the used concepts of the artifacts. We argue that this artifact-oriented view provides means to map academic models and description techniques onto existing development processes in industry. Additionally, it has a positive impact on the syntactic and semantic quality of the created artifacts [Me11].

An **artifact** is seen as a structured abstraction of modeling elements used as input, output, or as an intermediate result of the development process [Me10]. Artifacts capture and document information about the system, its development, and its context. Thus, they document system properties. This leads to an artifact model, which contains the artifacts to be developed during a specific development process together with their internal structure and dependencies between them.

In **artifact-oriented development**, the entire development process is understood as the stepwise construction of artifacts, always focusing on the results (the artifacts) that need to be produced during a development rather than focusing on the methods and processes that create them. We speak of “artifact-driven” versus “process-driven” development.

The basic assumption of artifact-oriented development is that, although development processes vary heavily from project to project, the content within the different artifacts is described by modeling concepts that are independent from the underlying development process. Note that content here refers to the logical content of an artifact, abstracting from its actual representation.

As artifact-oriented development puts emphasis on consistent result structures and used terminology, a given artifact can be created using quite different methods, processes and representations. The underlying development process is then just an arrangement of the artifacts produced during system development, together with the methods that produce them. This leads to a flat method structure. On the opposite, an activity-based (process driven) approach puts emphasis on how to produce something (rather than what to be produced) with a more vague description of content and structure, producing a flat artifact structure, where dependencies between artifacts only arise from dependencies between the methods that create them. It should be noted that even in a process-centric environment, such as the development of automation software, an artifact-oriented approach can be applied by filtering the results of the various process steps and abstracting from the methods to produce them in the first place.

Artifact orientation comes with various interpretations and manifestations in practice. Therefore, we need a clear definition of the term artifact itself. There is a variety of information that is embodied in an artifact. We distinguish between:

- the structure of the artifact (e.g., given by a table of content)
- the artifacts logical content, i.e., the pure assertions about a system
- the modeling concepts, i.e., the language by which the logical content is expressed
- the representation (including description techniques) of the artifacts content (e.g., natural text, diagrams, models, tables)
- the dependencies between the logical content

Contribution: This paper provides a conceptual framework that encompasses an artifact-oriented view onto the development of embedded systems by introducing two different models: The artifact model and the concept model. In a nutshell, the concept model defines modeling concepts that are used to describe a set of content items (e.g., elements and relations necessary to specify a state machine). These content items are arranged in a process-dependent hierarchical artifact structure that is defined in the artifact model. The artifact model contains the set of artifacts that need to be produced within the specific engineering process together with the dependencies and relations between them. Each artifact has a hierarchical structure (a table of content) of (sub-) artifacts with leafs being the various content items. Each content item of an artifact is linked to a concept of the concept model by a specific representation of the concept.

Upon such a framework, we are able to provide concepts for different content items independent from the engineering process. On top of that, we can define clear responsibilities and support a progress control for the production of artifacts. This framework can also be used to compare different engineering processes (e.g., from different application domains) and to pinpoint potential needs for optimization. Furthermore, the framework provides the basis for the definition of tracing links and dependencies between the different contents items and artifacts, allowing for a seamless development.

After introducing this conceptual framework, we instantiate it by providing a concept model for the development of embedded systems as it is worked out in the SPES_XT[1] project. We exemplarily show how this concept model can be linked to a given engineering process in industry by providing a process-dependent artifact model. We furthermore show the benefits of this approach by highlighting dependencies between artifacts, which need to be maintained in that engineering process.

2 Related Work

Artifact orientation has gained much attention in recent years, especially in requirements engineering approaches. In these approaches, artifact orientation is used to define RE reference processes and connect them with special concepts that are used within these processes.

REMsES [Br10] provides a process guide for supporting requirements engineering processes in the automotive industry. This approach is based on three models: an artifact model, a process model, and an environment model. The artifact model provides a basic structure for the definition of the artifacts, their assignment to abstraction layers and content categories, and the relations between the artifacts. It defines general control flow dependencies within requirements engineering processes. The process model defines the coarse-grained course of action and fine-grained task descriptions. It defines individual artifact-related tasks. The environment model defines the interfaces between the environmental processes that interact during the engineering process of the system with its requirements engineering process. The approach in this paper follows this idea and extends it to be applicable for the entire engineering process and not just RE. Additionally, it also focuses on the concepts used to define parts of the artifacts allowing for a precise definition of dependencies between artifacts.

REMbIS [Me10] is a model-based RE approach for the application domain of business information systems. It consists of (1) an artifact abstraction model that defines horizontal abstraction and modeling 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 generic process model with milestones, phases, and roles that

[1]`http://spes2020.informatik.tu-muenchen.de/spes_xt-home.html`

defines the activities and tasks of the RE process. We follow this approach in large parts and extend it to capture the entire development process.

3 Conceptual Framework for an Artifact-oriented Development

The conceptual framework introduced in this section distinguishes between the process-dependent set and structure of artifacts, defined in an artifact model, and the process-independent use of concepts used to describe certain content, defined in the concept model.

The concept model is a collection of modeling concepts together with dependencies that may exist between the different concepts. Each concept is characterized by an ontological basis, which describes the ontological entities of the concepts and the relations between them (see Figure 1). Thus, a concept defines an abstract syntax of a modeling language. Besides this pure description of the syntax, the concept model could also provide semantics for the concepts used. A concept model captures the complete vocabulary of the engineering tasks necessary to develop a system. Therefore, it provides modeling languages for a well-defined, structured specification of the content, while at the same time abstracting from the actual representation used in a specific artifact (e.g. tables, plain text, models, code). Concept models can have different levels of "richness" with regard to how expressive and customized the defined concepts are. Simple concept models could just provide general modeling concepts like state machines or Petri nets. Richer concept models could provide more specialized concepts, which build upon such simple concepts in order to define a more specific content. A specialized concept for the definition of a system function could use the concept of a state machine to describe a functions behavior.

Since systems and their descriptions and documentation can get very large, it is essential to adequately structure the produced artifacts as well as their logical content. Therefore, we define an artifact model in the conceptual framework that defines the artifacts produced in a development process as a hierarchical structure with specific content items as leafs (see Figure 1). The artifact model defines for each artifact the expected content (e.g., defined by means of a taxonomy). In this model, content items serve as containers for the actual content and define single areas of responsibility. In addition, we can regard these content items as artifacts by themselves and different content items may be grouped together to form another artifact at a higher level, which may be the outcome of a specific process step. This leads to a hierarchical structure of artifacts with content items being the leaves of the tree (see Figure 2).

As an example, the reader may consider a system specification as a concrete artifact to be produced during the development process. The structure of this specification (e.g., given by its table of content) will also be described in the artifact model. The concepts used to describe the individual pieces of content within the system specification are described in the concept model. To link the two models together, the concepts are related to content items. In doing so, a concept is given a specific representation that is used to describe the concept within an artifact. The concept now becomes content that appears in one or more artifacts, depending on the development process being used. However, the representation

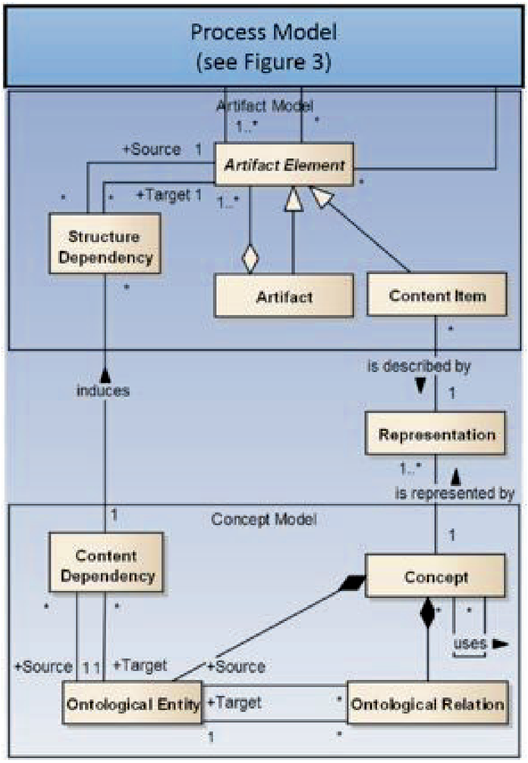

Figure 1: Meta Model of the Conceptual Framework (cf. [Me10])

of the concept might vary depending on the artifact in which it is expressed. Therefore, we say that a concept has a number of representations (e.g., diagram, text, or table), which are used to describe different content items in artifacts. As a simple example of the above, we consider the concept of a state machine, which can be used to describe the behavior of a logical component in the architecture of a system. The development process used may call for a system architecture specification, which includes the state machines of all components as state transition diagrams. Another way to describe these state machines is a tabular representation, which could be used in an interface specification document. The representation link also allows for the use of different concrete modeling languages. For example in an avionic context, a state machine might be represented by a SysML State Machine Diagram, whereas in an automotive context this might be expressed using a Simulink Stateflow Chart.

Please note that the concept model does not only cover content items that appear in documents. It also includes content that arises from producing code for example. In this case the appropriate concept might be a specific programming language or a structural element (e.g., a class or a method).

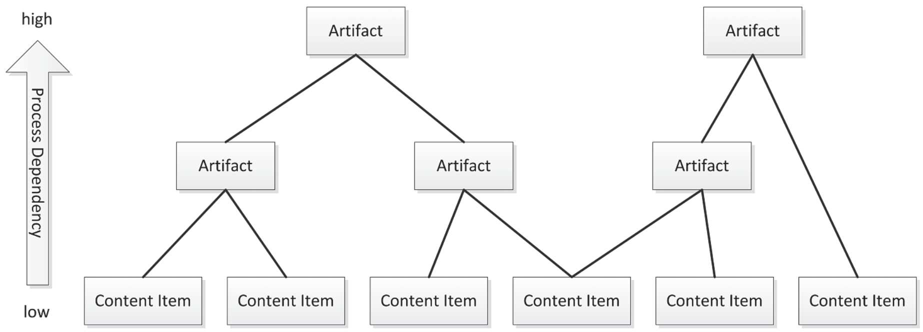

Figure 2: Artifact Hierarchy

A special challenge in a structured system development approach is an appropriate handling of dependencies. In our framework we aim at a precise description of dependencies between concepts, content items, and artifacts. Dependencies on the level of concepts describe the relation between ontological entities of different concepts. These inter-concept dependencies express dependencies with regard to content and are independent of the development process (e.g., events of a state machine must be consistent with the interface of the logical component in which they are embedded).

When concepts are mapped to the artifact model and thus become content items in a specific artifact, these content dependencies induce dependencies between content items and therefore also between artifacts. The induced dependencies arise from the chosen artifact structure and constrain the way the artifacts can be created. Please note that the dependencies between the content items are not limited to a single artifact. An artifact model thus defines a description of the set of required artifacts, their structure and contents, and the relations between the artifacts.

4 Mapping Content and Artifacts to a Development Process

The content items of the artifacts together with the related concept model and their dependencies can be regarded as a blue-print of a comprehensive system specification covering the whole development process. Therefore, an artifact model can be used as a reference model that captures the domain-specific results of the development steps. As in artifact-oriented development the content items are independent of the development process there must be a mapping of the artifact model to the actual development process.

This mapping is established by assigning the artifacts of the artifact model to tasks and milestones of the development process. Conversely, we can obtain content items from a given development process by filtering the results of the various process steps and abstracting from the methods to produce them.

Figure 3 provides a meta model for this mapping. The generic process model structures a development process into a set of milestones and tasks. Each artifact is assigned to a milestone where it has to be delivered. Artifacts are produced within tasks, in which potentially other artifacts are needed as inputs in order produce new artifacts.

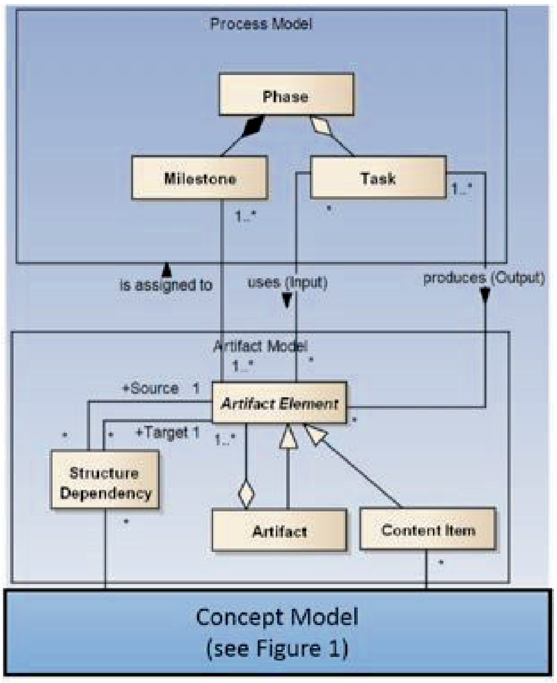

Figure 3: Mapping of the Artifact Model onto a Generic Process Model (cf. [Me10])

5 Example: Mapping the SPES Modeling Framework to the V-Model XT Process

The SPES Modeling Framework, as described in [Br12], provides a structured set of model types that are considered beneficial for the development of embedded systems. These model types are structured into so called Viewpoints, which group the model types according to some concerns following the standard of IEEE42010. Each model type has an ontological basis. Thus, the SPES Modeling Framework can be considered as a concept model. However, the model types in SPES do not only cover the information about the concepts to be used. They additionally provide information about the content that should be addressed by using these concepts. Therefore, the model types in SPES can also be considered as a basic set of content items that can/need to be created in the development of an embedded system. In summary, the SPES Modeling Framework defines a set of content items (basic artifacts) together with concepts that are used to describe the content items. In the following, we will map these content items and concepts to an artifact model instance for development process V-Model XT [Fr09].

The V-Model XT is a development process meta model that needs to be instantiated for a given project context. The instantiation provides an organisational tailoring of considered roles, activities, and products to be produced during the development. In the context of the V-Model XT, artifacts are called (work)products. For this paper, we exemplarily examine the high-level products “Anforderungen (Lastenheft)”[2] and “Gesamtsystemspezifikation (Pflichtenheft)”[3]. We will create an instance of an artifact model containing these two products (artifacts) and show how the content items of the SPES Modeling Framework can be related to these artifacts. Figure 4 shows the artifact model for the two products. The mapping of the content items to the artifacts is based on the textual descriptions given for the products and the SPES Modeling Framework. Note, that this model is not complete. Both products contain further content that is not considered here.The V-Modell XT documentation additionally provides a mapping for these two products to tasks in which they need to be created.

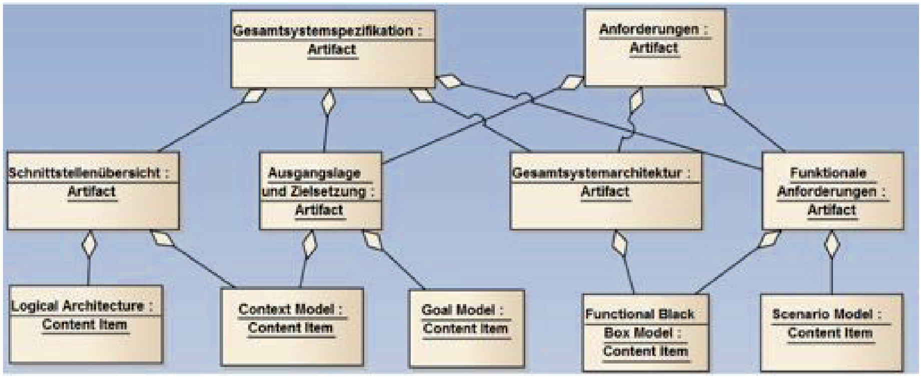
Gesamtsystemspezifikation : Artifact
Anforderungen : Artifact
Schnittstellenübersicht : Artifact
Ausgangslage und Zielsetzung : Artifact
Gesamtsystemarchitektur : Artifact
Funktionale Anforderungen : Artifact
Logical Architecture : Content Item
Context Model : Content Item
Goal Model : Content Item
Functional Black Box Model : Content Item
Scenario Model : Content Item

Figure 4: Artifact model instance for the V-Modell XT products “Gesamtsystemspezifikation” and “Anforderungen”.

The SPES Modeling Framework also defines dependencies between the model types that are expressed on the level of the used concepts. There is for example a dependency defined which states that the interface behavior of the Functional Black Box Model must be refined by the interface behavior of the Logical Architecture. This dependency on the level of concepts induces a dependency between the content items and finally between the artifacts

[2]http://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/1.4/Dokumentation/V-Modell%20XT%20HTML/14794f684e963e8.html#ref14794f684e963e8

[3]http://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/1.4/Dokumentation/V-Modell%20XT%20HTML/f436f8cfc083ae.html#reff436f8cfc083ae

to which they are linked to in the artifact model. Exactly this dependency between the two artifacts is also described in the V-Modell XT documentation[4]. However, with the definition of dependencies on the level of concepts we can be much more precise and process-independent.

6 Conclusion and Outlook

In this paper, we have shown that an artifact-oriented view can be used to structure the system development into process-dependent and process-independent parts. This separation is useful in order to assess and classify academic development approaches and to map them to specific development processes used in industry. This does not only cover specific tasks and artifacts used in the process but also specific representations used.

The presented conceptual framework is open to increments in order to cover additional aspects. One aspect, for example, could cover the artifacts life cycle by extending the artifact model with attributes that entail the current state of an artifact (e.g., draft, in review, released).

A further benefit of the presented framework is that it opens the way to a fully integrated tooling environment, in which content items, which are described by concepts can be linked to specific tools that are used in order to create these content items. The dependencies in the concept model clearly state the connection between the models within tools that need to be maintained during development. For practical purposes the content items can be stored in a content repository, similar to a Product-Lifecycle-Management system (PLM). The relations between the content items are maintained by the repository system such that changes in one content item are automatically updated in all related content items. Given the artifact model, the actual artifact documents can be generated by compiling the corresponding content items from the repository. This way, artifacts always reflect the current state of development and have a higher quality.

References

[Br10] Peter Braun, Manfred Broy, Frank Houdek, Matthias Kirchmayr, Mark Müller, Birgit Penzenstadler, Klaus Pohl, and Thorsten Weyer. Guiding requirements engineering for software-intensive embedded systems in the automotive industry. *Computer Science - Research and Development*, 2010.

[Br12] Manfred Broy, Werner Damm, Stefan Henkler, Klaus Pohl, Andreas Vogelsang, and Thorsten Weyer. Introduction to the SPES Modeling Framework. In Klaus Pohl, Harald Hönninger, Reinhold Achatz, and Manfred Broy, editors, *Model-Based Engineering of Embedded Systems*. Springer Berlin Heidelberg, 2012.

[4] `http://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/1.4/Dokumentation/V-Modell%20XT%20HTML/18296108af1c73c3.html#ref18296108af1c73c3`

[Fr09] Jan Friedrich, Ulrike Hammerschall, Marco Kuhrmann, and Marc Sihling. Das V-Modell XT. In *Das V-Modell XT*, Informatik im Fokus. Springer Berlin Heidelberg, 2009.

[Me11] Daniel Méndez Fernández, Klaus Lochmann, Birgit Penzenstadler, and Stefan Wagner. A case study on the application of an artefact-based requirements engineering approach. In *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, 2011.

[Me10] Daniel Méndez Fernández, Birgit Penzenstadler, Marco Kuhrmann, and Manfred Broy. A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering. In Dorina Petriu, Nicolas Rouquette, and Øystein Haugen, editors, *Model Driven Engineering Languages and Systems*, volume 6395 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010.