# Constructing Tool Chains Based on SPEM Process Models

Matthias Biehl, Martin Törngren
*Embedded Control Systems*
*Royal Institute of Technology*
*Stockholm, Sweden*
{*biehl,martin*}*@md.kth.se*

***Abstract*—The development of embedded systems requires a number of tools and it is widely believed that integrating the tools into an automated tool chain can improve the productivity of development. However, tool chains are not accepted by practitioners if they are not aligned with the established development culture, processes and standards. Process models exist for a variety of reasons, i.e., for documenting, planning or tracking progress in a development project and SPEM is the standardized formalism by the OMG for this purpose. We explore in how far a SPEM process models can be used for creating the skeleton of a tool chain, which is aligned with the process. We identify a number of relationship patterns between the development process and its supporting tool chain and show how the patterns can be used for constructing a tool chain. In two case studies, we examine the practical applicability of the patterns, when tailoring the design of a tool chain to a development process.**

***Keywords-Generative Approach; Model Driven Development; Process Modeling; Tool Integration; Embedded Systems.***

## I. INTRODUCTION

The engineering of an embedded system requires experts from a number of different engineering disciplines. Each engineering discipline prefers a different set of development tools that excel in that particular discipline [1]. The use of single, specialized tools has the potential to improve the development process, depending on the degree of automation they provide [2]. Since engineers need to exchange data and these tools do not interoperate well, a software external to the tools – a tool chain – is needed to facilitate the integration. Multiple tools have the potential to improve the productivity in the development process, depending on how well they are integrated with each other and their degree of automation [3]. Tool chains can provide different coverage of the development process; therefore, we distinguish between task-oriented tool chains with a small coverage and lifecycle-oriented tool chains with a larger coverage.

Many existing tool chains cover only one task in the development process, e.g., the tool chain between source code editor, compiler and linker. We call these tool chains task-oriented. The tools are used in a linear chain, so that the output of one tool is the input for the next tool. These tool chains have a relatively small scope and integrate a small number of tools from within one phase in the lifecycle. Characteristic for these traditional tool chains are their linear connections, using a pipes and filter design pattern [4].

In contrast, lifecycle-oriented tool chains have a larger scope, they focus on supporting the complete lifecycle from requirements engineering over verification and implementation to maintenance. In embedded systems development, these tool chains may span multiple disciplines such as software engineering, hardware engineering and mechanical engineering. These tool chains integrate a large number of different development and lifecycle management tools. In addition, modern development processes put new demands on the tool chain: processes might be agile, iterative or model-driven, which implies that the supporting tool chain cannot be linear.

When building a tool chain, it is thus important to study which development tools need to be connected. This information about the relationship of development tools is often already available in a formalized model. The Software & Systems Process Engineering Metamodel (SPEM) [5] can be used to describe the lifecycle. A SPEM model might already be available independently from a tool integration effort, e.g., as it is the case development with the Automotive Open Software Architecture (AUTOSAR) [6]. The information available in process models forms the skeleton of a tool chain, i.e., which tools are involved and how are they connected in the process. To construct an executable tool chain as a software solution, more detailed information is needed than is available in process models, e.g., information about the data of tools, how to access it, how to convert it and how to describe the relation between data of different tools. In this paper we evaluate to what extent information from existing SPEM models can be used for constructing a tool chain.

This paper is organized as follows: In Section II, we explain our approach for creating an initial design of a tool chain from a SPEM process model. We introduce SPEM for describing the processes and TIL to describe the architecture of a tool chain in Section II-A. This allows us to describe the relationship between process and tool chain as patterns in Section II-B and introduce ways of using the patterns in Section II-C. We apply the approach in two case studies in Section III. In the remaining sections, we discuss our approach, relate it to other work in the field, sketch future work and consider the implications of this work.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 

## II. Approach

Tool chains are intended to increase the efficiency of development by providing connections between tools [3]. Ideally, connections for all tools used throughout the development process are provided; and in this case the tool chain supports the development process. The process provides constraints and requirements for the construction of the tool chain. While generic process models are available, e.g., the SPEM models for the Rational Unified Process (RUP) [7] or for AUTOSAR [6], companies also create individual process models for various purposes, e.g., to customize these generic models to their individual environments, to document the development process, to plan the development process, to track the progress in the development or to document their selection of tools.

If the process and the tool chain are described in a model, information from the process model can be reused for constructing a tool chain model. This approach ensures that the tool chain and the process are aligned. Process models only contain some, but not all information necessary for specifying tool chains. Especially the type of the connection between tools needs to be added later on.

### A. Formalized Description of Processes and Tool Chains

In this section, we introduce modeling languages that are used for both the process and the design of the tool chain. We select two specific modeling languages, which on the one hand limits the scope of the work, on the other hand is a necessary preparation for formalizing and using the relationship between process and tool chain (cf. future work in Section VI).

*1) Modeling the Product Development Process:* There are both formal and informal processes in companies, documented to different degrees and there is an increasing trend to model processes. Several established languages exist for modeling processes or workflows. These languages have various purposes, BPMN [8] and BPEL [9] describe business processes and SPEM describes development processes. We apply SPEM, since it is a standardized and relatively widespread language for modeling development processes with mature and diverse tool support. A SPEM model describes both the product development process and the set of tools used and can thus be applied to describe the process requirements of a tool chain. An example model is provided in Figure 2. A number of concepts are defined in SPEM, we introduce here the core concepts that are relevant in the context of tool chains: a *Process* is composed of several *Activities*; an *Activity* is described by a set of linked *Tasks*, *WorkProducts* and *Roles*. A number of relationships, here represented by $\ll . \gg$, are defined between the concepts of the metamodel: a *Role*, typically an engineer, can $\ll perform \gg$ a *Task* and a *WorkProduct* can be marked as the $\ll input \gg$ or $\ll output \gg$ of a *Task*. A *WorkProduct* can be $\ll managed\ by \gg$ a *Tool* and a *Task* can $\ll use \gg$ a *Tool*.

*2) Modeling the Design of the Tool Chain:* We need an early design model that describes all important design decisions of a tool chain and chose to use the Tool Integration Language (TIL) [10], a domain specific modeling language for tool chains. TIL allows us not only to model a tool chain, but also to analyze it and generate code from it. The implementation of a tool chain can be partly synthesized from a TIL model, given that metamodels and model transformations are provided. Here we can only give a short overview of TIL, for an elaborated description of concrete graphical syntax, abstract syntax and semantics we refer to [10]. TIL has two basic types: *Components* and *Channels*, where *Components* are connected by *Channels*. The most important *Components* are *ToolAdapters*. For each tool, a *ToolAdapter* describes the set of data and functionality that is exposed by that tool in form of a tool adapter metamodel. Events can be triggered by *Users*. The relation between the tool adapters is realized as any of the following Channels: a *ControlChannel* describes a service call, a *DataChannel* describes data exchange by a model transformation and a *TraceChannel* describes the creation of trace links.

### B. Relationship Patterns between Process and Tool Chain

If the process and tool chain are formalized as a model, we can also model the relationship between them more formally. A process described in SPEM might provide several opportunities for tool integration. Such an opportunity involves two tools and a direct or indirect connection between them. The tools and the connections found in SPEM are included into the tool chain architecture as ToolAdapters and Channels. The direction of the data flow can be determined by the involved work products, which have either the role of input or output of the task. Tasks connected to only one tool or tasks dealing with work products connected to the same tool do not require support from a tool chain; in these tasks engineers work directly with this tool, e.g., by using the GUI of the tool. To describe this relationship in more detail, we list patterns of both SPEM and TIL models and their correspondences.

Table I
Correspondences between SPEM and TIL Metaclasses

| SPEM Metaclass | TIL Metaclass |
|---|---|
| RoleDefinition | User |
| ToolDefinition | ToolAdapter |
| TaskDefinition | Channel |

The relationship patterns consist of a SPEM part, which matches a subgraph of a process model in SPEM, and a TIL part, which will become a new subgraph in the tool chain model in TIL. In the following, we show four SPEM patterns that describe tool integration related activities, they are illustrated in Figure 1, (1) - (4). The corresponding TIL pattern is the same for all SPEM integration patterns,

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 

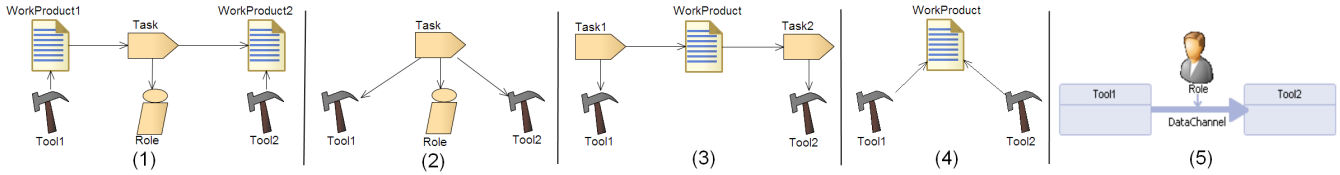


Figure 1. SPEM and TIL Patterns

visualized in (5). This mapping is established by pairs of model elements from both SPEM and TIL, whose name attribute is equivalent and whose types are of the metaclasses presented in Table I.

- *Task-centered Integration Pattern:* For each *TaskDefinition* in SPEM that has associated *WorkProducts* as input and output, where the input *WorkProduct* has a different associated *ToolDefinition* than the output *WorkProduct*, this pattern produces *ToolAdapters* and a *Channel* between them in the TIL model. The SPEM pattern is shown in (1) and can be observed in case study 1 in Figure 2 for the *Task TraceReq2UML* connecting the *WorkProduct RequirementsDatabase* and the *WorkProduct UMLFile*.
- *Multi-tool Task-centered Integration Pattern:* For each SPEM *TaskDefinition* with two SPEM *ToolDefinitions* associated with it, this pattern produces *ToolAdapters* and a *Channel* between them in the TIL model. The SPEM pattern is illustrated in (2).
- *WorkProduct-centered Integration Pattern:* For each SPEM *WorkProduct* that is both input and output of its associated *TaskDefinitions*, which have a different associated *ToolDefinition*, this pattern produces *ToolAdapters* and a *Channel* between them in the TIL model. The SPEM pattern is illustrated in (3) and can be observed in case study 2 in Figure 4 for the *WorkProduct ECUConfigurationDescription*, which is output of the *Task GenerateBaseECUConfiguration* and input to the *Task GenerateRTE*.
- *Multi-tool WorkProduct-centered Integration Pattern:* For each SPEM *WorkProduct* in SPEM that is associated to two different *ToolDefinitions*, this pattern produces *ToolAdapters* and a *Channel* between them in the TIL model. The SPEM pattern is illustrated in (4).

For all relationship patterns, the following constraints need to be fulfilled: For each *RoleDefinition* in SPEM that is connected to the *TaskDefinition*, we create a *User* model element in the TIL model. If a *ToolAdapter* corresponding to the *ToolDefinition* already exists in the TIL model, the existing *ToolAdapter* is connected, otherwise a new *ToolAdapter* is produced.

*1) Implementation as Model Transformations:* The implementation of the patterns offers possibilities for automation of the pattern usage. We implement the relationship patterns as model transformations, with SPEM as the source metamodel and TIL as the target metamodel. We chose the model-to-model transformation language in QVT-R, with the mediniQVT engine, and the Eclipse Modeling Framework (EMF) for realizing the metamodels. We use the SPEM metamodel, which is provided by the Eclipse Process Framework (EPF) under the name Unified Method Architecture (UMA), and for the visualization of SPEM models we use Enterprise Architect. For modeling and visualization of TIL, we use the TIL Workbench described in [10].

Patterns (1) to (5) are graphical representations of the relational QVT model transformation rules. Since QVT relational is a declarative language, the implementation describes the source patterns (1) - (4) and the target pattern (5) in the form of rules. Additionally, the attributes between source and target pattern are mapped, as described in Table I. Due to space constraints, the QVT rules are not included here.

### C. Usage of Relationship Patterns

The relationship patterns can be used in different ways. Here, we apply the relationship patterns for constructing the initial design of a new tool chain starting from a process model. Other forms of using the relationship patterns are possible, but are not considered in depth here. We can use the patterns, e.g., for verification: based on a process model and a tool chain model we check if the requirements provided by the process are realized by the tool chain model.

The focus of this paper is the application of the relationship patterns to create an initial tool chain design in TIL from the process requirements expressed in the SPEM model. The patterns can be applied to a SPEM model that is complete and contains all necessary references to *ToolDefinitions*. The patterns ensure that the design of the tool chain is aligned with the process, a necessity for acceptance of the tool chain with practitioners. This design of the tool chain can be created in an automated way and might need to be iteratively refined by adding details.

The process model only provides the skeleton for the specification of a tool chain, such as the tools, which tools are connected and which user role is working with the tools. The process model does not provide the nature of the connections and the exact execution semantics of the automated tool chain. The nature of the connection can be data exchange, for creating trace links between tool data or for accessing specific functionality of the tool. This

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 

information needs to be added manually by configuring and choosing the right type of channel in TIL, a DataChannel, TraceChannel or ControlChannnel. Also, events need to be specified that trigger the data transfer or activate the tracing tool. For each ToolAdapter, a metamodel describing the data and functionality of the tool need to be added to the TIL model. For each DataChannel, a model transformation needs to be added.

To handle these cases, we add a refinement step, which complements the automated construction. Once this information is added, the TIL model can be used as input to a code generator for tool chains, as detailed in [11].

## III. Case Studies

In this section, we apply the identified relationship patterns between a process model and a tool chain in two industrial case studies. This gives us the opportunity to study different ways of using the patterns and to explore the impact of different modeling styles.

### *A. Case Study 1: Construction of a Tool Chain Model for a Hardware-Software Co-Design Process*

This case-study deals with an industrial development process of an embedded system that is characterized by tightly coupled hardware and software components. The development process for hardware-software co-design is textually described in the following:

- The requirements of the embedded system are captured in the IRQA[1] tool. The system architect designs a UML component diagram and creates trace links between UML components and the requirements.
- The UML model is refined and a fault tree analysis is performed by the safety engineer. When the results are satisfactory, the control engineer creates a Simulink[2] model for simulation and partitions the functionality for realization in software and hardware.
- The application engineer uses Simulink to generate C code, which is refined in the WindRiver[3] tool. The data from UML and Simulink is input to the IEC-61131-3 conform ControlBuilder tool. The data from ControlBuilder, Simulink and WindRiver is integrated in the Freescale development tool for compiling and linking to a binary for the target hardware.
- A hardware engineer generates code in the hardware description language VHDL from Simulink and refines it in the Xilinx ISE[4].

Based on the description of the process, we have created the corresponding SPEM model visualized in Figure 2.

We apply the model-to-model transformation that realizes the relationship patterns on the SPEM model in Figure 2. This yields a tool chain model that is aligned with the process, as shown in Figure 3. By applying the task-centered integration pattern shown in (1), we identify integration tasks that are linked to two work products that in turn are linked to different development tools (e.g. the task *Trafo_UML2Safety*). Some other tasks are not concerned with integration, they are related to one tool only (e.g. the task *Use_UML*).

The TIL model resulting from application of the relationship patterns is internally consistent; this means that there are no conflicts, missing elements or duplications in the model. All tools mentioned in the SPEM model are also present in the TIL model as ToolAdapters and all ToolAdapters are connected. In addition, the approach ensures that the design of the tool chain matches the process.

Since the tool chain is modeled, we can easily change, extend and refine the initial model before any source code for the tool chain is developed. The TIL model is relatively small compared to the SPEM model, thus hinting at its effect to reduce complexity. When using the simple complexity metric of merely counting model elements and connections, we see that in the TIL model their number is reduced by $2/3$ compared to the SPEM model (cf. table II).

Table II
Size of the SPEM and TIL Model of Case Study 1

| Count | Model Elements | Connections |
|---|---|---|
| SPEM Model | 43 | 71 |
| TIL Model | 13 | 26 |

The important architectural design decisions of the tool chain (such as the adapters and their connections) can be expressed in TIL, while the complexity has been decreased compared to a SPEM model (cf. table II). The tool chain model can be analyzed and - after additional refinement with tool adapter metamodels and transformations - can be used for code generation, as detailed in [11], [10]. Moreover, the presented model-driven construction of the tool chain ensures that the tool chain is aligned with the process.

### *B. Case Study 2: Verification of a Tool Chain Model for AUTOSAR ECU Design*

In this case study, we model a tool chain for AUTOSAR. AUTOSAR is developed by the automotive industry and defines an architectural concept, a middleware and in addition a methodology for creating products with AUTOSAR. The AUTOSAR methodology describes process fragments, so called capability patterns in SPEM. Generic AUTOSAR tool chains are implemented in both commercial tools and open frameworks, however, it is a challenge to set up tool chains consisting of tools from different vendors [12] and tool chains customized to the needs of a particular organization.

The SPEM process model is provided by the AUTOSAR consortium and is publicly available, which contributes to

[1] http://www.visuresolutions.com/irqa-web
[2] http://www.mathworks.com/products/simulink
[3] http://www.windriver.com
[4] http://www.xilinx.com/ise

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 

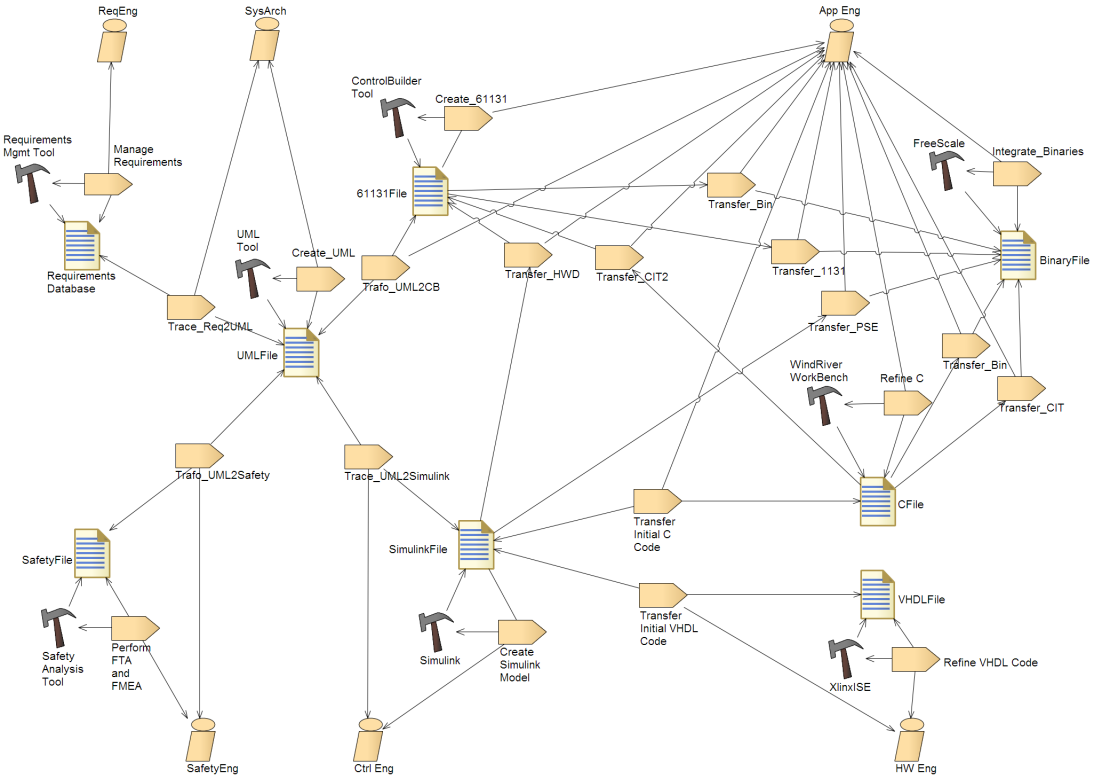


Figure 2. Case Study 1: Product Development Process of the Case Study as a SPEM Model

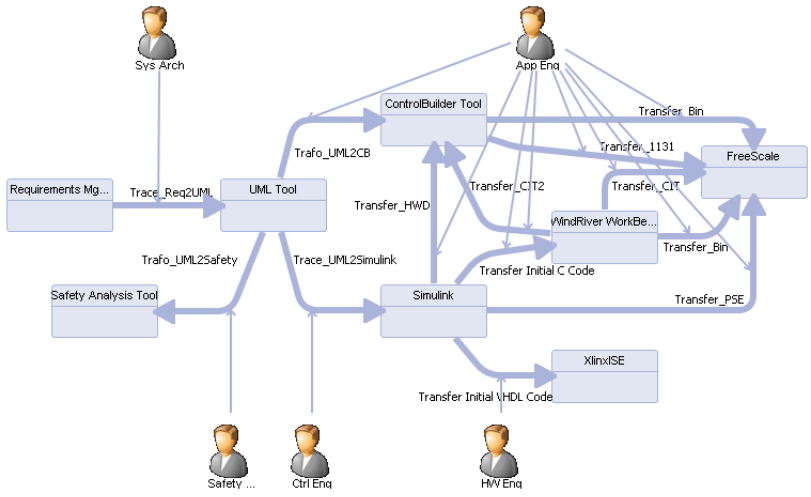


Figure 3. Case Study 1: Tool Chain of the Case Study as a TIL Model

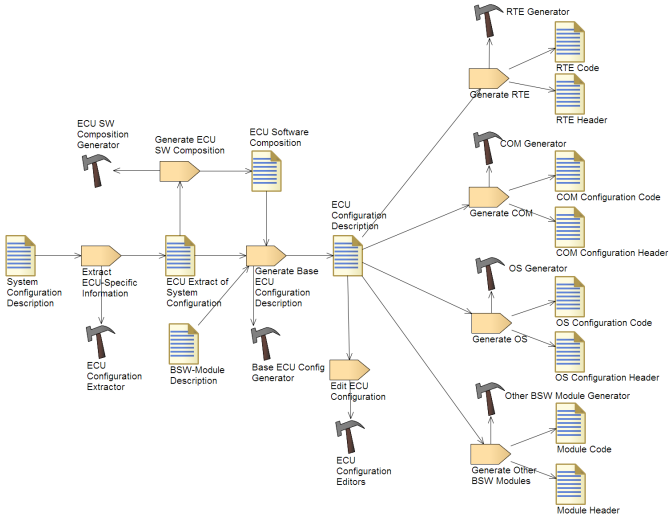


Figure 4. Case Study 2: Excerpt of the AUTOSAR Methodology for Designing an ECU [6].

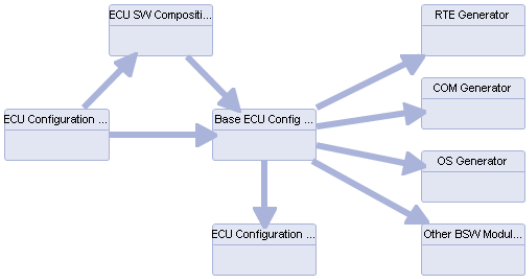


Figure 5. Case Study 2: AUTOSAR Tool Chain for Designing an ECU as a TIL Model

the transparency of this case study. An excerpt of this model that is relevant for the design of a ECU, is depicted in Figure 4. We use this excerpt of the SPEM model to initialize a tool chain. Applying the patterns creates the tool chain model in TIL, illustrated in Figure 5. Out of the four different SPEM parts of the relationship patterns (1) - (4), only the workproduct-centered integration pattern (3) matched several times in the SPEM model. This is due to the modeling style used in the AUTOSAR methodology, where *WorkProducts* are used as an interface for integrating tools.

The generated skeleton of the tool chain lays the foundation for ensuring that the AUTOSAR methodology can be realized by this tool chain. The skeleton can now be refined with metamodels, model transformations and the behavior.

## IV. Discussion

This approach assumes that an appropriate process model for tool chains is available. We assume that the process model does not contain any integration related overhead, i.e., explicit representation of a model transformation tool and intermediate data model. We assume that tools have been assigned to process activities. The choice for certain tools is usually independent of automating the tool chain, the choice merely needs to be documented in the process model.

The use of the presented patterns is limited to processes represented in SPEM and tool chains modeled in TIL. However, the patterns could be adapted to similar process metamodels.

While it is possible to describe a part of the requirements for a tool chain with SPEM, SPEM models are not executable. We thus extract the relevant information from SPEM models to ensure that all tools and connections are represented in the tool chain. Based on this information, we generate a TIL models, which can be made executable by following a well-defined process, described in [10].

We have evaluated the approach in two case studies from the area of embedded systems. We do not see any reason why the patterns could not be applied for creating tool chain from process models in other application areas in the future.

## V. Related Work

Related work can be found in the areas tool integration and process modeling. There are a number of approaches for tool integration, as documented in the annotated bibliographies [13], [14]. Most of the approaches do not take the process into account; in this section we focus on those approaches that do. We also take approaches from process modeling into account and classify them according to two dimensions: The first dimension comprises different execution mechanisms, which can be interpretation vs. compilation. The second dimension comprises different process modeling languages, which can be proprietary vs. standardized.

Interpretation-based approaches [15], [16], [17] use the process definition for tool integration. This technique is also known as enactment of process models. Since the description of the process is identical to the specification of the tool chain, no misalignment between process and tool chain is possible. There are two preconditions for this approach: the process model needs to be executable and the access to data and functionality of the development tools needs to be possible. The use of a proprietary process model for interpretation in tool chains is introduced in [18], as the process-flow pattern. Approaches that extend SPEM make the process model executable [15], [16]. The orchestration of tools by a process model is shown in [17]. However, the interpretation of integration related tasks is often not possible, since the interfaces to the development tools are not standardized. Thus, the use of process enactment to build tool chains is limited.

Compilation-based approaches transform the process model into another format, where the process model serves as a set of requirements. Proprietary process models provide great flexibility to adapt them to the specific needs of tool integration. An integration process model is developed in [19], where each process step can be linked to a dedicated

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 

activity in a tool. For execution, it is compiled into a low-level process model. The proprietary process model needs to be created specifically for constructing a tool chain. In this work, we use the standardized process metamodel SPEM [5], which allows us to reuse existing process models as a starting point for building tool chains and as a reference for verification for tool chains.

## VI. Conclusion and Future Work

Process models exist for a variety of reasons, i.e., for documenting, planning or tracking progress in a development project and SPEM is the standardized formalism by the OMG for this purpose. In this paper, we recognize the development process modeled in SPEM as a set of requirements for the architecture of tool chains. We devise a number of patterns for creating the skeleton of a tool chain, which is aligned with the process.

In this work, we have selected specific languages to express the patterns; in the future, we would like to experiment with additional languages for describing the process model, such as BPMN. This might help us to further generalize the patterns.

*Acknowledgement*

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement 100203.

## References

[1] J. El-khoury, O. Redell, and M. Törngren, "A Tool Integration Platform for Multi-Disciplinary Development," in *31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 442–450, 2005.

[2] T. Bruckhaus, N. H. Madhavii, I. Janssen, and J. Henshaw, "The impact of tools on software productivity," *Software, IEEE*, vol. 13, no. 5, pp. 29–38, Sep. 1996.

[3] M. Wicks and R. Dewar, "A new research agenda for tool integration," *J. of Sys. and Sw.*, vol. 80, no. 9, pp. 1569–1585, Sep. 2007.

[4] M. Shaw and D. Garlan, *Software architecture*. Prentice Hall, 1996.

[5] OMG, "Software & Systems Process Engineering Metamodel Specification (SPEM)," "OMG", Tech. Rep., Apr. 2008.

[6] AUTOSAR Consortium. (2011, Apr.) Automotive open software architecture (AUTOSAR) 3.2. [Online]. Available: http://autosar.org/

[7] P. Kruchten, *The Rational Unified Process*. Addison-Wesley Pub (Sd), 1998.

[8] OMG, "Business Process Model And Notation (BPMN)," "OMG", Tech. Rep., Jan. 2011.

[9] OASIS, "OASIS Web Services Business Process Execution Language (WSBPEL) TC," "OASIS", Tech. Rep., Apr. 2007.

[10] M. Biehl, J. El-Khoury, F. Loiret, and M. Törngren, "On the Modeling and Generation of Service-Oriented Tool Chains," *Journal of Software and Systems Modeling*, vol. 275, 2012 [Online]. Available: http://dx.doi.org/10.1007/s10270-012-0275-7

[11] M. Biehl, J. El-Khoury, and M. Törngren, "High-Level Specification and Code Generation for Service-Oriented Tool Adapters," in *ICCSA2012*, pp. 35–42, Jun. 2012.

[12] S. Voget, "AUTOSAR and the automotive tool chain," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10, pp. 259–262, 2010.

[13] M. N. Wicks, "Tool Integration within Software Engineering Environments: An Annotated Bibliography," Heriot-Watt University, Tech. Rep., 2006.

[14] A. W. Brown and M. H. Penedo, "An annotated bibliography on integration in software engineering environments," *SIGSOFT Softw. Eng. Notes*, vol. 17, no. 3, pp. 47–55, 1992.

[15] A. Koudri and J. Champeau, "MODAL: A SPEM Extension to Improve Co-design Process Models New Modeling Concepts for Today's Software Processes, LNCS vol. 6195, ch. 22, pp. 248–259, 2010.

[16] R. Bendraou, B. Combemale, X. Cregut, and M. P. Gervais, "Definition of an Executable SPEM 2.0," in *APSEC*, pp. 390–397, 2007.

[17] B. Polgar, I. Rath, Z. Szatmari, A. Horvath, and I. Majzik, "Model-based Integration, Execution and Certification of Development Tool-chains," in *Workshop on model driven tool and process integration*, pp. 36–48, Jun. 2009.

[18] G. Karsai, A. Lang, and S. Neema, "Design patterns for open tool integration," *Software and Systems Modeling*, vol. 4, no. 2, pp. 157–170, May 2005.

[19] A. Balogh, G. Bergmann, G. Csertán, L. Gönczy, Horváth, I. Majzik, A. Pataricza, B. Polgár, I. Ráth, D. Varró, and G. Varró, "Workflow-driven tool integration using model transformations," in *Graph transformations and model-driven engineering*, pp. 224–248, 2010.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1 