A tool approach for supporting integration projects

Stefan Kühne, Maik Thränert

University of Leipzig
Augustusplatz 10–11, 04109 Leipzig, Germany
{kuehne|thraenert}@informatik.uni-leipzig.de

Abstract: Integration projects involve expert groups with different backgrounds and skills. Coordination mechanisms that manage the close collaborative work are given through the concepts of procedure models. An effective use of procedure models requires tool support that covers easy-to-use documentation aspects as well as sophisticated run time support for the complete development life cycle. Therefore we introduce a multi-layer tool architecture capable of supporting all stages of the application of procedure models. In particular we focus on the transformation of models between different abstraction layers. The practicability of our approach is discussed on the base of a prototypical implementation of its tool components.

1 Introduction

Business application integration is critical for organizations because it has strong influence on the flexibility of organizations to react to changing business requirements, e.g. to deliver new IT-enabled services. Existing applications must be combined with new technologies to create an integrated information system structure that supports flexible, dynamic, cross-organizational business processes [Lin01, Bus03]. Methods, tools and techniques for the development of adequate solutions are the topic of the so-called integration engineering that combines aspects of classical software engineering with specific approaches to integration problems.

According to [OFA01] business application integration can be covered on three different layers. On the top layer, the strategic layer, business solutions are adjusted to the specific needs of different customer segments. On the underlying business layer collaboration processes including e-services are defined, analyzed and optimized according to their specific business requirements. The mapping of these processes to an integrated IT-structure is done on the technical layer. Here it is considered which technical systems can be used for the implementation and how their integration can be reached.

Integration projects have strong cooperative characteristics [SDT05]. The development process involves different expert groups with different backgrounds and skills (e.g. customers, business analysts IT architects, integration specialists and software developers) that closely have to work together. Development tools (e.g. business process modeling tools, integration tools and software development tools) also differ from each other. This leads to several kinds of problems that tend to increase project risks. *Political aspects*

arise when design decisions, e.g. the decision whether a legacy system should be used for the implementation of an IT-enabled service, is affected by interests of political influences. *Communication problems* arise when different terminologies affect the communication between people with different backgrounds. *Methodical aspects* cover the effective combination of business- and technical-oriented methods and techniques for an adequate integration solution. These are supplemented by *technical aspects* that enable a seamless integration of development tools.

Methodical and technical aspects of a comprehensive support for an integration project are covered by concepts of the so-called computer aided integration engineering (CAIE) [SDT05]. This term is inferred from the sustained discussion about computer aided software engineering (CASE) with respect to the specific characteristics of the integration engineering domain. CAIE includes functional requirements like the management of the engineering process, the management of development artifacts as well as the management of resources.

Aim of this paper is to describe a CAIE tool that is capable of supporting the main aspects of integration engineering processes. Our approach assumes that methodical aspects are modeled on an abstract level in terms of a domain-specific, environment-independent procedure model. From this abstract representation an environment-specific process schema is derived that is executable by a runtime engine. The remainder is organized as follows. Section 2 gives an overview about the CAIE tool, its components and the relationships between them. Section 3 describes the modeling component in more detail. Section 4 introduces the general architecture of the CAIE tool runtime environment and describes concepts and implementation issues of its transformation component in particular.

Our CAIE tool approach is principally generic. It can also be applied in other engineering domains with strong cooperative characteristics, e.g. the IT service engineering [SS04] that combines aspects of classical software engineering and business oriented service engineering.

2 The CAIE tool approach

2.1 General tool approaches

Computer aided integration engineering can be classified into two general approaches – passive and active tool support. Passive support is focused on descriptive aspects, i.e. it covers the documentation of project issues independent of the state of the engineering process. This includes general guidelines, static project guides, implementation aids, glossary support and access to documentation of similar projects. An active support is focused on engineering process dependent issues, e.g. the delivery of artifact templates and sample documents for running activities, resource and capacity planning, process monitoring, integrated document management, quality management as well as support for user communication and cooperation.

As mentioned in section 1 engineering processes in the integration engineering domain involve specific development tools. This includes task-specific development tools (vertical tools), e.g. analysis tools, modeling tools and validation tools, as well as task-unspecific tools (horizontal tools), e.g. configuration management tools and documentation tools. Active tool support approaches can further be subdivided as to whether these domain-specific tools (horizontal and vertical) are integrated into one comprehensive development environment or not.

According to [Was89] there are five levels of tool integration in software engineering environments. The first level is *platform integration.* It stands for a virtual operating environment for tools, which provide distributed services. *Presentation integration* means a common look-and-feel of the tools from a user's perspective. *Data integration* has its focus on the sharing of data among tools and the managing of their relationships. The idea of *control integration* is that tools are able to notify other tools as well as to perform notifications. The most sophisticated integration level is *process integration.* It includes the track of software development activities as well as the management and improvement of the process in general.

It supports well-defined engineering processes, including the track of software development activities as well as the management and improvement of the process in general. The process based tool integration requires well-defined engineering processes. As will be shown later this could be done via a top-down approach by using procedure models.

A procedure model is a generic model of the development process and the maintenance and modification of systems. It consists of general guidelines, e.g. quality recommendations and "values", a development process model as well as implementation guidelines and aids. The definition of the development process model contains a set of activities, techniques, roles and results and describes the relationships between them.

2.2 The process of procedure model application

A procedure model is not intended as a direct support for real projects. It is abstracted in such a way that it is independent of organizational or project-specific conditions. Therefore it comprises the main aspects and is reusable in different problem scenarios.

To be applicable to a real problem scenario a procedure model must be adapted to the specific organizational and project-specific environment. The adaptation of a procedure model involves some transformation steps that break down the procedure model to an executable process model (see figure 1). In a first tailoring step the relevant procedure model components are selected according to a project type that is determined by project-specific property combination. The result is a project type specific procedure model. In a second tailoring step project specific instantiations and adaptations are made, e.g. a generic placeholder for a business process modeling tool is instantiated by a concrete tool reference. The next step transforms the project specific procedure model into an executable procedure model. This model is loaded into a runtime engine, where it is instantiated and applied to a specific integration engineering project. The reverse direction includes the

propagation of problems and optimizations to higher layers. These aspects are left out of the scope of this paper.

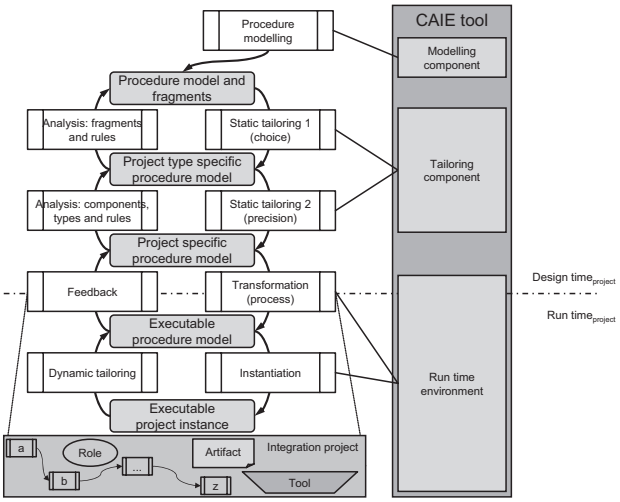

Figure 1: Application of a procedure model.

A comprehensive tool approach requires support on all layers. With the help of a modeling component procedure models or rather fragments of it are modeled. A tailoring component supports the adaptation to specific organizational or project-specific boundaries and requirements. The runtime environment supports the engineering process during its execution.

3 Representation of integration engineering procedure models

3.1 Related work

One of the main aspects of procedure models is the representation of the contained development process model or rather in the area of software engineering the contained software process model. These models are expressed by software process modeling languages (SPM). In the last few decades many SPM languages were developed. According to their language primitives they can be classified into textual and graphical languages.

Textual approaches are mostly extensions of programming languages and therefore can be classified very similary to them. Rule-based languages, e.g. SPELL [JLC92], represent activities with rules with pre- and post-conditions. Functional approaches, e.g. HFSP [Kat89], represent activities as functions with input and output objects. Procedural languages, e.g. APPL/A [SHO95], describe the control-flow with command statements.

Object-oriented approaches, e.g. PML IPSE [Rob88] are focused on organizational aspects, e.g. the message flow between artifacts.

Graphical SPM languages are more user friendly than the textual ones. Data model oriented approaches, e.g. SOCCA [EG94], are extensions of classical data modeling approaches. They represent processes through pre-defined object types. Data flow oriented approaches, e.g. Improvise [BKC95], represent the data flow and describes the transformation processes of data entities. Petri net based languages are the most expressible graphical approaches. Examples of this category are SLANG [BFG94] and FUNSOFT Nets [DG94].

Other characteristics that can be used for a classification of the mentioned SPM approaches are the existence of a well-defined language syntax, a formal semantic and constructs that enable extensibility. They can be further differentiated according to their expressible power, usability, tailoring flexibility and analyzing capabilities. The choice of a language depends on the aim of use.

A comprehensive tool support requires an approach that covers usability as well as execution-oriented aspects. Usability is focused on a graphical representation of the process model, e.g. the generation of process documentation. Furthermore an intuitive semantic that enables abstractions understandable to people without any technical background, e.g. customers or business experts, will increase the acceptability of the procedure model. The process of procedure model application mentioned in section 2.2 establishes a framework for a combined approach. On the top layer an abstract, documentation-oriented approach can be used. After refinements during a tailoring process an executable process represented in a language with a formal syntax and a formal semantic is derived from that.

3.2 Modeling procedure models with the ARIS Software Engineering Scout

The Software Engineering Scout [IDS03] by IDS Scheer AG describes a procedure model for software engineering projects. It contains general guidelines, a static process guide (the so-called Software Engineering Scout Assistant) as well as implementation guidelines and aids. The Software Engineering Scout Assistant contains a detailed description of the most important activities, results, milestones, roles and their dependencies. It is used for the direct support of software engineering projects (planning and execution). The content can be published, e.g. in the form of HTML-content in an organizations' intranet. In the following we want to describe the meta-model of the Software Engineering Scout.

On the top level the procedure model is modeled with a value added chain diagram (VAC). The VAC diagram is used to structure the model into phases and milestones. Each phase is detailed by an additional VAC diagram. On the second level a phase is structured into working packages and activities. Activities are more detailed in additional diagrams with respect to different views. Timing dependencies (the successors of an activity) are modeled in an additional VAC. Input and output artifacts as well as responsible and involved roles are modeled in an additional function allocation diagram (FAD). Connections of activities with external tools or data entities are represented in an additional product/service

exchange diagram. The hierarchical structure of artifacts is modeled in product/service tree diagrams. The organizational structure, e.g. organizational units and roles, is represented in organizational charts.

The Software Engineering Scout meta-model is available as modeling conventions in the form of a method filter for the ARIS Design Platform [IDS05] by IDS Scheer. Through this it can be used by ARIS modeling tools, like the ARIS Web Designer or the ARIS Toolset. Figure 2 shows two screenshots of the ARIS Web Designer – a definition of a phase in a second level VAC diagram and a detailed description of an activity in an assigned FAD diagram.

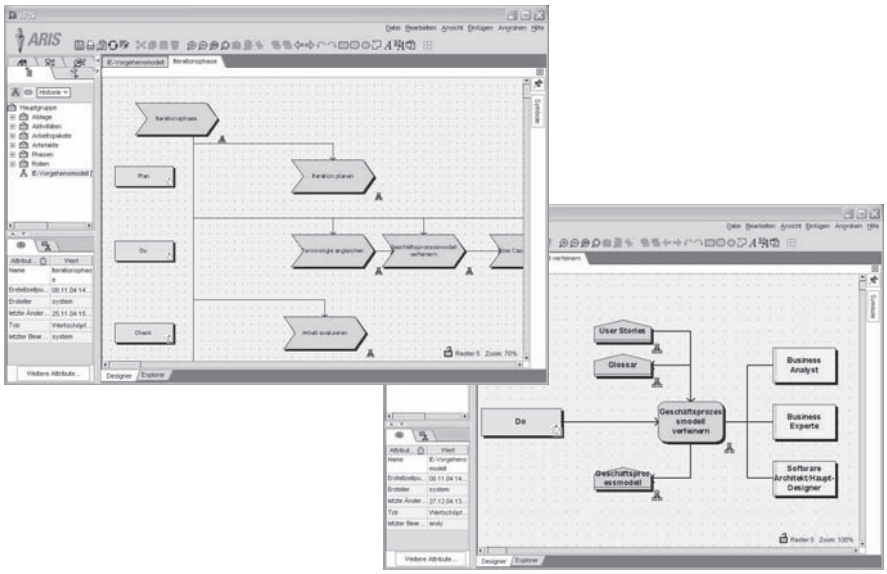

Figure 2: Software Engineering Scout.

The Software Engineering Scout by IDS Scheer has several advantages. It enables formal representations of procedure models. The modeling conventions are structured clearly, so the practice efforts are limited. With the help of the ARIS Process Platform there are matured tools available to model procedure models. Furthermore it is possible with the same tools to make project-specific adaptations in a tailoring process. The result can be exploited by different kinds of reports, e.g. the Scout Factory report enables the publishing of the procedure model as HTML content (see figure 3).

The Software Engineering Scout has several disadvantages, too. The process view is modeled in VAC diagrams. Through this the expressible power of control-flow dependencies is limited. An evaluation on the base of workflow patterns [AHK02] shows that basic constructs (Sequence, Parallel Split, Synchronizing Merge) can be modeled, but in contrast advanced workflow patterns [AHK00] (e.g. Deferred Choice, Multiple Instances Patterns) cannot. The hierarchical refinement of activities is also limited.

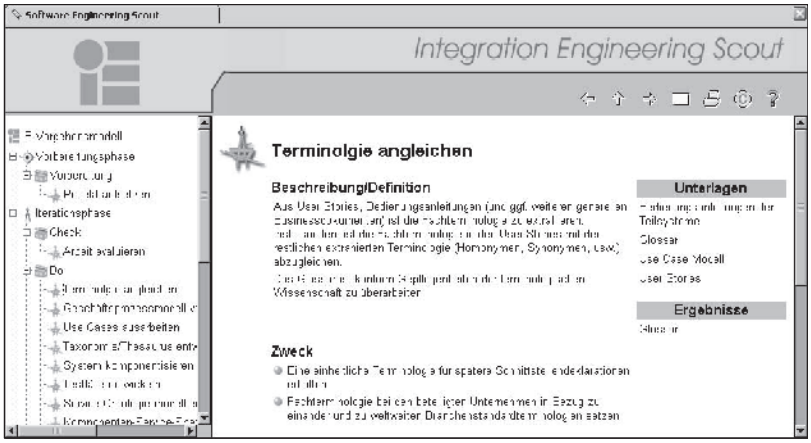

Figure 3: Integration Engineering Scout HTML export.

The workflow patterns represent control-flow aspects that are required on an execution level. The representation of procedure models takes place on a higher level of abstraction. Therefore not all patterns are required. Figure 4 shows the result of an evaluation according to their relevance in the area of integration engineering procedure models. Two advanced patterns with a high relevance have been identified – the Deferred Choice and the Multiple Instance with priori runtime knowledge.

The Deferred Choice enables the exclusive choice of two (or more) action alternatives offered to the environment. The decision is not made explicitly, e.g. it is independent from data that is available to the workflow system. This pattern enables a kind of late binding flexibility for the decision base. In the domain of integration engineering this is important for agile processes.

In engineering processes tasks often have to be processed several times for different inputs, e.g. in the integration engineering domain a unit test must be designed for different sub-components. Of course, the number of sub components is not known at design time. Therefore only at runtime can one determine how many unit tests have to be designed. To express this issue the workflow pattern Multiple Instances with priori runtime knowledge can be used. Here the number of instances of a task is defined at runtime before the instances are created.

To be able to express the workflow patterns Deferred Choice and Multiple Instances with priori runtime knowledge the modeling conventions of the Software Engineering Scout were extended to the so-called Integration Engineering Scout modeling conventions. The Deferred Choice can be expressed by setting the XOR-split attribute to true. The use of the Deferred Choice implies the introduction of an XOR-join attribute to synchronize exclusive execution paths. Both, the XOR-split and the XOR-join are expressed graphically by “xor”-marks (as later shown in figure 6). The pattern Multiple Instances with

Pattern	Rel.	Pattern	Rel.
Sequence	●	Multiple Choice	◐
Parallel Split	●	Synchronizing Merge	◐
Synchronization	●	Multiple Merge	○
Exclusive Choice	◐	Discriminator	○
Simple Merge	●	N-out-of-M Join	○
Arbitrary Cycles	○	Cancel Activity	◐
Implicit Termination	○	Cancel Case	◐
MI, without synchronization	○	Deferred Choice	●
MI, priori design time knowledge	◐	Interleaved Parallel Routing	◐
MI, priori runtime knowledge	●	Milestone	◐
MI, no a priori runtime knowledge	◐		
● .. relevant, ◐ .. useful, ○ .. not relevant			

Figure 4: Relevance of patterns according to procedure models.

priori runtime knowledge can be expressed by setting a MI attribute to true. This is shown graphically by an “MI”-mark.

4 The CAIE tool runtime environment

4.1 Architecture

Figure 1 shows the CAIE tool components according to the application process of procedure models. Having described the modeling component the following part will focus on the runtime environment and the part of transforming graphical models into executable processes. Figure 5 shows the architecture of the runtime environment in more detail. It is designed as a web application which enables user and tool integration in distributed cooperative engineering processes. The architecture of the runtime environment corresponds to a hub and spoke architecture, which enables a seamless integration of external components. The hub, named CAIE core, integrates external components like a workflow engine, a data repository and others. The CAIE core is based on Apache Struts [Apa05a], a framework for Java web applications. Through this a strict model view controller architecture (MVC) [KK00] is enforced.

The controller is divided into the layers application logic and integration logic. The integration layer consists of generic interfaces for different types of external tool components, e.g. a workflow engine, as well as specific stub implementations. The application layer combines connected external components and implements the business processes of the CAIE tool. The result is an integrated solution that can be seen as a realization of a so-called cockpit concept. The runtime environment enables a comprehensive user guide through the complete engineering life cycle. A workflow engine is responsible for the process execution. It computes what should be done, when and by whom. The workflow engine also integrates underlying engineering tools. A document management component manages the artifacts that are created in the engineering process. It also enables the ac-

cess to artifacts produced in similar projects. Integration projects are knowledge intensive processes. In addition to project-specific information, e.g. task descriptions or document templates, the developer needs problem-specific information according to his specific context. These pieces of information can be provided by a process-oriented knowledge management system, e.g. the PreBIS engine [BH05]. Other functionalities, like analyzing aspects, can be easily added by integration of other components.

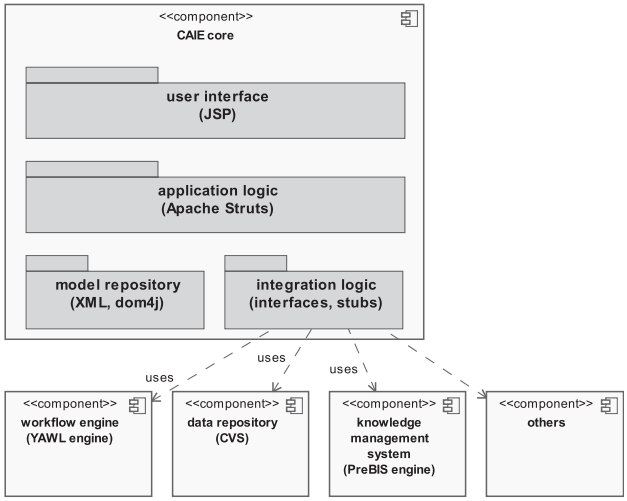

Figure 5: Architecture of the CAIE runtime environment.

In the following we want to concentrate on the choice and the integration of an appropriate workflow component. The connection requires a transformation of project-specific procedure models to the workflow language used by the workflow engine. This is described afterwards.

The design of the runtime environment enables the integration of workflow engines, which provide an interface for remote communication. There are many workflow systems available, commercial and non-commercial ones [AAD03]. A classification of workflow systems according to the kind of processes they support is given in [RRA03]. Explicitly structured processes are supported by classical production workflow systems. Case handling systems extend the application domain to implicitly structured processes. Ad-hoc structured processes are supported by ad-hoc workflow systems. The CAIE tool approach assumes structured engineering processes; therefore production workflow and case handling systems can be applied.

Amongst others we evaluated the Oracle BPEL Process Manager [Ora05], the YAWL system [AAD03] and the jBPM system [JBP05]. We selected the YAWL system because it provides several advantages that facilitate an integration. It is open source and runs like the CAIE core under a Java Servlet and JavaServer Pages (JSP) compliant servlet container, e.g. the Apache Jakarta Tomcat [Apa05b] or the JBoss Application Server [JBo05]. Fur-

thermore the YAWL system was designed to support the workflow language YAWL (Yet Another Workflow Language) [AH03]. According to the workflow patterns YAWL is a very expressible language. Except for Arbitrary Cycles it supports all workflow patterns mentioned in [AHK02, AHK00]. Therefore the relevant patterns in the area of procedure models (see figure 4) are supported, too.

4.2 Transformation

A workflow-based execution support for procedure models requires the mapping of tailored project-specific procedure models to process models executable by the runtime environment. The mapping includes the transformation of the control-flow perspective, the data perspective, the organizational perspective and the functional perspective. In the case of the CAIE tool the control-flow perspective and the data perspective are handled by external tool components, i.e. the connected YAWL engine and the document management component respectively. In the following we want to focus on the transformation of the control-perspective, which requires a mapping from the modeling conventions of the Integration Engineering Scout (the extended Software Engineering Scout) to the YAWL language.

A process model in YAWL, defined as a YAWL specification, consists of a set of extended workflow nets (EWF-net). Each EWF-net may contain atomic and composite tasks. A composite task refers to unique EWF-net. A YAWL specification has exactly one EWF-net, called top level workflow that is not referred to by a composite task.

The concept of a tree-like structure of EWF-nets is similar to assignments of the top level VAC in the Integration Engineering scout. Therefore an intuitive mapping is the transformation of each VAC diagram to an EWF-net and the contained VAC symbols to atomic tasks. Each VAC symbol that has an assignment to another VAC diagram (according to the modeling conventions: the VAC symbols in the top level VAC diagram) is transformed to an atomic task followed by a composite task that refers to the corresponding EWF-net of the assigned VAC diagram.

For the mapping of the control-flow perspective of a VAC diagram the framework of workflow patterns (see [AHK02, AHK00]) is used again. They are applicable for this purpose because they represent the full range of control-flow aspects in a language independent manner. The identification of a workflow patterns representation in the “source” and the “sink” language is a direct hint to a transformation rule. Figure 6 shows the relationships between VAC diagrams (according to the modeling conventions of the Integration Engineering Scout) and YAWL specifications on the base of workflow patterns. The Parallel Split has two representations in VAC diagrams. The first one is the connection of a VAC symbol with two (or more) successors with the help of “is predecessor of”-relationships in an assigned VAC diagram. This is mapped to a YAWL-task that is connected with successors by an “AND”-split. The second representation is the set of VAC symbols that don’t have any predecessor. This set represents process lines that are processed in parallel. This variant of Parallel Split is transformed to an unnamed YAWL-task that is connected with

successors by an “AND”-split. The Parallel Join also has two representations. The first one is the connection of two (or more) VAC symbols with the same successor. This is mapped to YAWL-tasks that are connected with a YAWL-task that is decorated by an “AND”-join. The second representation is the set of VAC symbols that don’t have any successor. This is mapped to YAWL by an unnamed task that is decorated by an “AND”-join.

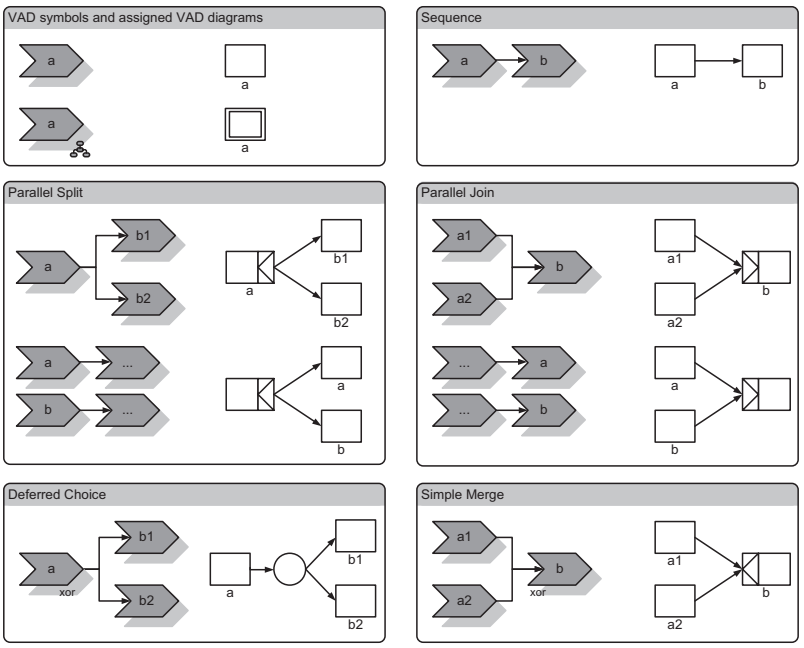

Figure 6: Mapping rules of VAC constructs to YAWL constructs.

The transformation algorithm is straightforward. It starts on the top level VAC diagram, applies the mentioned transformation rules and repeats the processing for each assigned second level VAC diagram. The technical realization of the mapping is done via an XML transformation. The project-specific procedure model is exported as an XML document, which is processed. The result is a valid XML document that can be directly uploaded to the YAWL engine. In a first version the transformation was implemented in the form of an XSLT stylesheet [W3C05]. The functional programming style of XSLT proved to be too slow and memory consuming, e.g. the computation of VAC symbols without any predecessors in VAC diagrams has a recursion depth that is linear to the number of VAC symbols. Therefore in a second version the transformation algorithm was implemented in Java using the open source library dom4j [dom05] for XML processing.

4.3 The presentation layer

The view consists of JavaServer Pages (JSP), data containers and static HTML content. The functionality of JSP is enlarged through custom tag libraries.

Figure 7 shows two screenshots of the CAIE runtime environment. The left one presents the control-flow perspective. Here the user gets a list of available and checked-out work items according to his permissions. The right one shows the functional perspective, where detailed pieces of information of the selected work item is presented, e.g. a description of what should be done, predecessor and successor activities, and input and output artifacts.

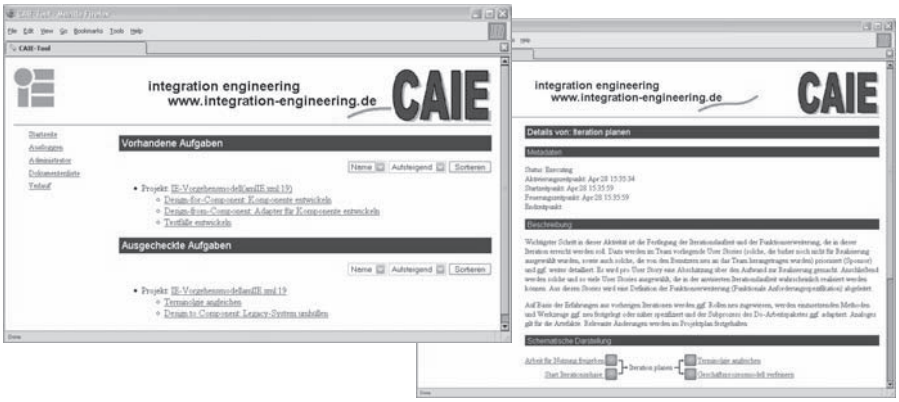

Figure 7: Screenshots of the CAIE runtime enviroment.

5 Conclusion and outlook

This paper presented a tool approach for computer aided integration engineering. It is based on the assumption that the engineering process should be derived from an abstract, project independent procedure model. With the Integration Engineering Scout we suggested a practical modeling approach for the representation and adaptation of procedure models. With the example of YAWL we presented a mapping from those models to executable workflow models. These workflow models can be loaded into a workflow based runtime environment that leads the user through the whole engineering process.

A focus for further research lies in the data perspective. There are high dependencies between artifacts of an engineering process, e.g. a platform independent business process model can be the base for several platform dependent implementation models [SDT05]. Here it has to be considered which relationships between artifacts exist and what notation independent concepts can be used for which transformation approaches. Another focus lies on the knowledge perspective. Here it has to be considered what knowledge based information retrieval concepts can be applied in the CAIE tool approach.

References

[AAD03] van der Aalst, W. M. P., Aldred, L., Dumas, M., ter Hofstede, A. H. M.: Design and Implementation of the YAWL System. QUT Technical report, FIT-TR-2003-07, Queensland University of Technology, Brisbane, 2003.

[AH03] van der Aalst, W. M. P., ter Hofstede, A. H. M.: YAWL : Yet Another Workflow Language (Revised Version). QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane, http://is.tm.tue.nl/staff/wvdaalst/Publications/p198.pdf, 2003.

[AHK00] van der Aalst, W. M. P., Barros, A. P., ter Hofstede A. H. M., Kiepuszewski, B.: Advanced Workflow Patterns. In: 7th International Conference on Cooperative Information Systems, volume 1901 of Lecture Notes in Computer Science, Springer, Berlin, 2000; 18–29

[AHK02] van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P.: Workflow Patterns. QUT Technical report, FIT-TR-2002-02, Queensland University of Technology, Brisbane, http://is.tm.tue.nl/staff/wvdaalst/publications/p159.pdf, 2002.

[Apa05a] The Apache Software Foundation: Struts. Website: http://struts.apache.org/, 2005.

[Apa05b] The Apache Software Foundation: Apache Jakarta Tomcat. Website: http://jakarta.apache.org/tomcat/, 2005.

[BFG94] Bandinelli, S., Fuggetta, A., Ghezzi, C., et. al.: SPADE : An Environment for Software Process Analysis, Design, and Enactment. In: Finkelstein, A., Kramer, J., Nuseibeh, B. (eds): Software Process Modelling and Technology, Research Studies Press, 1994; 223–247

[BH05] Karsten Böhm, Jörg Härtwig: Prozessorientiertes Wissensmanagement durch kontextualisierte Informationsversorgung aus Geschäftsprozessen. In: 7. Internationale Tagung Wirtschaftsinformatik, 2005; 943–962

[BKC95] Barghouti, N. S., Koutsofios, E., Cohen, E.: Improvise : Interactive Multimedia Process Visualization Environment. In: Proceedings of the 5th European Software Engineering Conference, 1995; 28–43

[Bus03] Bussler, C.: B2B integration : concepts and architecture. Springer, Berlin et. al., 2003.

[DG94] Deiters, W., Gruhn, V.: The FUNSOFT Net Approach to Software Process Management. In: International Journal of Software Engineering and Knowledge Engineering, 4(2), 1994; 229–256

[dom05] dom4j: dom4j. Website: http://www.dom4j.org/, 2005.

[EG94] Engels, G., Groenewegen, L.: SOCCA : Specifications of Coordinated and Cooperative Activities. In: Finkelstein, A., Kramer, J., Nuseibeh, B. (eds): Software Process Modelling and Technology, Research Studies Press, 1994; 71–102.

[IDS03] IDS Scheer: ARIS Software Engineering Scout. Website: http://www.ids-scheer.com/international/english/products/aris_scouts/23227, 2003.

[IDS05] IDS Scheer: ARIS Design Platform. Website: http://www.ids-scheer.com/international/english/products/aris_design_platform/23247, 2005.

[JBo05] JBoss: JBoss Application Server. Website: http://www.jboss.org/products/jbossas, 2005.

[JBP05] jBPM.org: Java Business Process Management. Website: http://www.jbpm.org/, 2005.

[JLC92] Jaccheri, L., Larsen, J.-O.; Conradi, R.: Software Process Modeling and Evolution in EPOS. In: Proc. Fourth International Conference on Software Engineering and Knowledge Engineering, 1992.

[Kat89] Katayama, T.: A hierarchical and functional software process description and its enaction. In: Proceedings of the 11th International Conference on Software Engineering, 1989; 343–352

[KK00] Kassem, N., Kassem, N., et. al.: Designing Enterprise Applications : Java 2 Platform. Addison-Wesley, Boston et. al., 2000.

[Lin01] Linthicum, D. S.: B2B Application Integration: e-Business-Enable Your Enterprise. Addison-Wesley, Boston et. al., 2001.

[OFA01] Österle, H., Fleisch, E., Alt, R.: Business Networking, Shaping Collaboration between Enterprises, Springer, Berlin et. al., 2001.

[Ora05] Oracle: Oracle BPEL Process Manager. Website: http://www.oracle.com/technology/products/ias/bpel/index.html, 2005.

[Rob88] Roberts, C.: Describing and acting process models with PML. In: Proceedings of the 4th international software process workshop on Representing and enacting the software process, 1988; 136–141

[RRA03] Reijers, H., Rigter, J., van der Aalst, W. M. P.: The Case Handling Case. In: International Journal of Cooperative Information Systems, Download: http://is.tm.tue.nl/staff/wvdaalst/Publications/p212.pdf, 2003.

[SDT05] Specht, T., Drawehn, J., Thränert, M., Kühne, S.: Modeling Cooperative Business Processes and Transformation to a Service Oriented Architecture. In: Proceedings of IEEE Conference on E-Commerce Technology, 2005.

[SHO95] Sutton, S. M., Heimbigner, D., Osterweil, L. J.: APPL/A. ACM. In: Transactions on Software Engineering and Methodology (TOSEM), v.4 n.3, 1995; 221–286

[SS04] Scheer, A.-W., Spath, D.: Computer Aided Service Engineering : Informationssysteme in der Dienstleistungsentwicklung. Springer Berlin et. al., 2004.

[Ver98a] Verlage, M.: Vorgehensmodelle und ihre Formalisierung. In: Kneuper, R. (ed.): Vorgehensmodelle für die betriebliche Anwendungssystementwicklung, B.G. Teubner, Stuttgart, Leipzig, 1998; 60–75

[Ver98b] Verlage, M.: Modellierungssprachen für Vorgehensmodelle. In: Kneuper, R.: Vorgehensmodelle für die betriebliche Anwendungssystementwicklung, B.G. Teubner, Stuttgart, Leipzig, 1998; 76–94

[Was89] Wassermann, A. I.: Tool Integration in Software Engineering Environments. In: Long, F. (ed.): Software Engineering Environments, Volume 467 of LNCS., Springer, Berlin et. al., 1989; 137–149

[W3C05] W3C: XSL Transformations (XSLT). Website: http://www.w3.org/TR/1999/REC-xslt-19991116, 2005.