Reducing the complexity of process-based integration using model-driven technologies

Stefan Kühne

Business Information Systems, Universität Leipzig
Johannisgasse 26, 04103 Leipzig, Germany
kuehne@informatik.uni-leipzig.de

1 The complexity of process-based integration of application systems

Integration engineering [1] is a sub discipline of software engineering which aims at the development and management of integrated business information systems. This includes the development of integratable sub systems as well as the development of integration solutions, e.g. by composing existing systems. One specific focus is the process-based integration of application systems expressed by workflow languages such as WS-BPEL or YAWL [2]. A common starting point for this type of integration are high-level process descriptions, e.g. based on languages such as EPC or UML AD, from that executable process models are derived.

Even comparatively simple business processes lead to complex workflow models. They are repetitive according to concerns such as error handling, logging and event signaling. The complexity of integration process models decreases their maintainability and adaptability.

2 Application of model-driven approaches and techniques

Model-driven approaches, such as MDE [3], MDSD [4] and GSD [5], are promising in tackling the complexity and inflexibility of current process-based integration. They propose formal problem descriptions in terms of (domain-specific) models and the usage of model operations that operate on these models to automate common development tasks. The application of a model-driven approach in a specific domain requires the definition of a domain architecture which includes modeling languages, generators, transformators and platform specifications.

Aim of our work is to develop a domain architecture to support and to automate common development tasks in the area of process-based integration of application systems. This architecture applies abstraction mechanisms to executable process languages according to aspects (e.g. error-handling, logging, monitoring) and patterns (e.g. workflow patterns, distributed communication patterns, data-flow patterns) expressed in domain-specific modeling languages. Furthermore it provides transformational mappings to workflow engines and model operations such as comparison, evolution, metrication and validation.

The development of such an architecture arises several questions according to modeling aspects and model operations that require domain-specific solutions.

– How can complexity reduction mechanisms, such as abstraction or separation of concerns, be applied to develop domain-specific process modeling languages?
– What methods and techniques can be applied to map high-level business process models to executable workflow models? Which refinement and weaving approaches should be applied? What kinds of model-to-model and model-to-text transformations have to be combined?
– What mechanisms can be applied to manage manual refinements of integration process models in incremental and iterative development cycles?
– How should dynamic aspects in models be validated across different abstraction layers? How should model evolutions be organized in this area? How can delta models of process models be exploited?
– How should the project-spanning domain engineering process and its interconnections to integration projects be organized?

To be applicable in different business domains the architecture is delivered as a framework, i.e. its artifacts are defined in an extendable manner. The usability (including its extensibility) of the domain architecture should be evaluated in two different business domains (e-government, e-commerce) and should be compared to other approaches provided by generic BPM tools, e.g. the ARIS SOA Architect [6].

This work represents work in progress. Some results are available in [7, 8].

References

1. Rautenstrauch, C.: Integration Engineering: Konzeption, Entwicklung und Einsatz integrierter Softwaresysteme. 1. edn. Addison-Wesley, Bonn ; Paris (1993)
2. Aalst, W., Hofstede, A.: YAWL: Yet Another Workflow Language. Information Systems **30**(4) (2005) 245–275
3. Favre, J.M., Nguyen, T.: Towards a Megamodel to Model Software Evolution Through Transformations. Electronic Notes in Theoretical Computer Science **127**(3) (2005) 59–74
4. Stahl, T., Völter, M.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons (2006)
5. Czarnecki, K.: Overview of Generative Software Development. In Banâtre, J.P., ed.: Unconventional Programming Paradigms (UPP) 2004. LNCS 3566, Mont Saint-Michel, France (2004) 313–328
6. IDS: ARIS Business Architect – Web-based Enterprise Architecture and Business Process Management. `http://www.ids-scheer.com/international/english/products/aris_design_platform/50277`, Saarbrücken (2007)
7. Kühne, S., Thränert, M., Rotzoll, W., Lehmann, J.: Model-Driven Integration Engineering in der E-Government-Domäne Meldewesen. In Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler, B., eds.: eOrganisation: Service-, Prozess-, Market-Engineering – 8. Internationale Tagung Wirtschaftsinformatik. Number 1, Karlsruhe, Germany, Universitätsverlag Karlsruhe (2007) 109–126
8. Kühne, S.: Modellgetriebene Entwicklung im Kontext des Integration Engineering. In Steffens, U., Addicks, J.S., Streekmann, N., eds.: MDD, SOA und IT-Management (MSI 2007), Workshop, Oldenburg, April 2007. (2007) 47–57