Towards Semi-automated Design of Enterprise Integration Solutions

Pavol MEDERLY*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
mederly@fiit.stuba.sk

Our aim is to reduce the effort needed to create integration solutions, i.e. specialized software systems that enable cooperation of disparate business applications (services), either within an enterprise or across enterprises. We are trying to achieve this goal by fully or partially automating the process of *technical design* of such solutions.

More specifically, the technical design of integration solutions includes choosing the overall architecture style (e.g. the one based on Process Manager or Pipes and Filters pattern), selecting an appropriate integration platform (e.g. a concrete Enterprise Service Bus, or ESB, product), and designing the solution components so that all the functional and non-functional requirements are met. Functional requirements are typically given by business analysts in the form of an abstract description of the workflow and data transformations the solution has to implement. On the other hand, typical non-functional requirements cover areas of availability, reliability, performance, security, logging and auditing, and maintainability of the solution, as well as ensuring compatibility with message formats, protocols and application programming interfaces (APIs) used by individual systems being integrated.

We concentrate primarily on *messaging-based integration solutions*, i.e. those that utilize a standardized message-oriented middleware infrastructure for communication between solution components. So far we have developed two methods that create designs of such integration solutions, taking into account a subset of the non-functional requirements categories described above – namely throughput, availability, logging, message ordering, choosing correct message format and content, and duplicate message handling. The methods are largely platform-independent and describe the solutions they create using enterprise integration patterns [1], the de-facto standard language in the area of messaging-based integration.

The first method [2] uses an *action-based planning* approach, representing properties of message flows present in the integration solution as the planner's states of the world, and potential solution components (i.e. business services as well as

* Supervisor: Professor Pavol Návrat, Institute of Informatics and Software Engineering

Spring 2010 PeWe Workshop, April 18, 2010, pp. 75–76.

integration services ensuring e.g. message format conversions, load balancing, fault tolerance, or message logging) as planning operators. Each operator has a set of preconditions (checking presence of specific literals in the state of the world) and effects (removing and adding some literals from/to the state of the world). These preconditions and effects correspond to laws inherent in the domain of messaging-based integration solutions. Our method encodes an integration problem as an action-based planning problem, executes a planner, and then interprets the plan found by the planner as a description of the integration solution.

The second method achieves similar goals using *constraint programming*. It encodes an integration problem as a Constraint Satisfaction Problem (CSP), representing properties of business and integration services and message flows between them as CSP variables, and the laws of messaging-based integration as constraints over these variables. Then it executes a CSP solver and interprets the solution found as a description of the integration solution (see Fig. 1).

Fig. 1. *A schema of the second method.*

Current results show that these methods, especially the second one, are able to find solutions for practically-sized integration problems within reasonable time (seconds). Yet what is needed is to broaden the set of design issues tackled by the methods: In the future we would like to deal with the problem how to place and manage logical data elements in messages, how to resolve security issues, how to support diverse message transport protocols, and how to deploy solution components into ESB containers. We also plan to more precisely define the notion of solution optimality. We expect that in order to achieve these goals we would have to involve the developer in the solution-finding process. Finally, we plan to provide a code-generation module that would generate partially or fully executable code for selected integration platforms.

Acknowledgement. This work was partially supported by the Slovak Research and Development Agency (contract No. APVV-0391-06) and by the Scientific Grant Agency of Republic of Slovakia (grant No. VEGA 1/0508/09).

References

[1] Hohpe, G., Woolf, B.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Pearson Education, Inc., Boston, MA (2004)

[2] Mederly, P., Lekavý, M., Závodský, M., Návrat, P. Construction of Messaging-Based Enterprise Integration Solutions Using AI Planning. In: *Preprint of the Proc. of the 4th IFIP TC2 Central and East European Conf. on Software Engineering Techniques*, CEE-SET 2009, (2009), pp. 37-50.