Taming Orchestration Design Complexity through the ADORE Framework

Sébastien Mosser, Mireille Blay-Fornarino

Taming Orchestration Design Complexity through the ADORE Framework

Demonstration submitted to the "journées 2010 du GDR GPL"

Sébastien Mosser and Mireille Blay–Fornarino

University of Nice Sophia – Antipolis
CNRS, I3S Laboratory, MODALIS team
Sophia Antipolis, France
{mosser,blay}@polytech.unice.fr

Abstract. The Service Oriented Architecture (SOA) paradigm supports the assembly of atomic services to create applications that implement complex business processes. Assembly is accomplished by service orchestrations defined by SOA architects. The ADORE framework allows SOA architects to model complex orchestrations of services by composing models of smaller orchestrations involving subsets of services. The smaller orchestrations are called orchestration *fragments* and encapsulates new concerns. ADORE is then used to weave fragments into existing application models. This demonstration illustrates how the ADORE framework can be used to model a SOA using fragments composition. We illustrate it using several implemented case studies.

Work Context: SOA & Business Processes Design

An application in the Service Oriented Architecture (SOA, [1]) paradigm is an assembly of services that implements a business process. SOA applications can be defined as orchestrations of services [2]. A SOA application is typically defined by business specialists and can involve many services that are orchestrated in a variety of ways. Furthermore, the need to extend an SOA application with new business features (to follow market trends or adapt a normalized process from a company to another one) arises often in practice. Existing tools and formalisms (*e.g.* BPMN notation [3], BPEL industrial language [4]) are technologically–driven. They use a *design–in–the–large* approach and considerable effort can be expended when using them to develop and adapt large applications involving many services that are orchestrated in a variety of ways.

We propose a *design–in–the–small* driven framework called ADORE[1] to tame the complexity of orchestration design. Experts focus on the design of small process *fragments*, and let the complexity of composing all the fragments into a final application to dedicated algorithms.

Relations to Aspect Oriented Modeling Approaches

Several approaches fill the gap between orchestrations and AOP (*e.g.*, [5], [6]). These approaches rely on the BPEL language and impose to use dedicated BPEL execution engines to interpret the aspects. ADORE preaches technological independence and exposes itself as a *model* to support composition [7]. Instead of interpreting *aspectized* BPEL code, ADORE aims to generate complete orchestrations of services, executable in any industrial engine.

One of the strength of ADORE is to focus on the so–called *Shared Join Points* (SJP, [8]) interactions spawn. We develop a set of rules to identify conflicting interactions between orchestration fragment at composition time. Instead of *re–ordering* the aspects to deal with conflicts around a SJP, we use an *order–independent* composition process. When interactions are detected, the user

[1] *Activity moDel supOrting oRchestration Evolution*, http://www.adore-design.org

will enter knowledge at a fine–grained level (where coarse–grained is fragment re–ordering) to solve the conflict and then ease the interaction.

In [9], authors propose a way to weave multiple aspects in UML sequence diagrams. They propose a very precise way to identify join points and express pointcuts, but their weaving methodology relies on a sequential aspect composition, where ADORE uses an order independent composition process.

Inspired by grid–computing community, ADORE proposes an algorithm (fully described in [10]) to automatically enhance a process with *set* concerns. Considering a process p handling a scalar data d, the algorithm can automatically transform p into a process handling a set of data $d^\star \equiv \{d1, \ldots, n_n\}$.

Moreover, ADORE allows users or programs to extract metrics from its internal representation, inspired by well-known indicators like [11].

Underlying Implementation

ADORE user interface is implemented as an EMACS major mode. This mode hides in an user–friendly way the set of shell scripts used to interact with the underlying engine. The engine is implemented as a set of logical rules, using the PROLOG language. A dedicated compiler implements an automatic transformation between ADORE concrete syntax and the associated PROLOG facts used internally by the engine. As visualizing processes is important in design phase, ADORE provides a transformation from its internal facts model to a GRAPHVIZ code. It produces as a result a graphical representation of ADORE models. Visualization tools and graphs screenshots are available on the project website.

Relation to other industrial or research efforts

ADORE was partially funded (2005 – 2009) by the French Research Agency (ANR) through the FAROS consortium. The work of the FAROS consortium (including both industrial– Orange Labs, EDF & Alicante – and academic – IRISA, I3S & LIFL– partners) was to propose a model–driven methodology to build reliable SOA. The ADORE framework is one of the platforms targeted by the FAROS methodology.

References

1. MacKenzie, M., Laskey, K., McCabe, F., Brown, P., Metz, R.: Reference Model for Service Oriented Architecture 1.0. Technical Report wd-soa-rm-cd1, OASIS (February 2006)
2. Peltz, C.: Web Services Orchestration and Choreography. Computer **36**(10) (2003) 46–52
3. White, S.A.: Business Process Modeling Notation (BPMN). IBM Corp. (May 2006)
4. OASIS: Web Services Business Process Execution Language Version 2.0. Technical report, OASIS (2007)
5. Charfi, A., Mezini, M.: Aspect-Oriented Web Service Composition with AO4BPEL. In: ECOWS. Volume 3250 of LNCS., Springer (2004) 168–182
6. Courbis, C., Finkelstein, A.: Weaving Aspects into Web Service Orchestrations. In: ICWS, IEEE Computer Society (2005) 219–226
7. Mosser, S., Blay-Fornarino, M., Riveill, M.: Web Services Orchestration Evolution : A Merge Process For Behavioral Evolution. In: 2nd European Conference on Software Architecture(ECSA'08), Springer LNCS (September 2008)
8. Nagy, I., Bergmans, L., Aksit, M.: Composing Aspects at Shared Join Points. In Hirschfeld, R., Kowalczyk, R., Polze, A., Weske, M., eds.: NODe/GSEM. Volume 69 of LNI., GI (2005) 19–38
9. Klein, J., Fleurey, F., Jézéquel, J.M.: Weaving Multiple Aspects in Sequence Diagrams. Transactions on Aspect-Oriented Software Development (TAOSD) **LNCS 4620** (2007) 167–199
10. Mosser, S., Blay-Fornarino, M., Montagnat, J.: Orchestration Evolution Following Dataflow Concepts: Introducing Unanticipated Loops Inside a Legacy Workflow. In: International Conference on Internet and Web Applications and Services(ICIW), IEEE Computer Society (May 2009)
11. Laue, R., Gruhn, V.: Complexity Metrics for Business Process Models. In Abramowicz, W., Mayr, H.C., eds.: BIS. Volume 85 of LNI., GI (2006) 1–12