An Architecture Description Language for Mobile Distributed Systems

Volker Gruhn, Clemens Schäfer

University of Leipzig
Faculty of Mathematics and Computer Science
Chair for Applied Telematics / e-Business*
Klostergasse 3, 04109 Leipzig, Germany
{gruhn,schaefer}@ebus.informatik.uni-leipzig.de

January 12, 2004

Abstract

Mobile software applications have to meet new requirements directly arising from mobility issues. To address these requirements at an early stage in development, an architecture description language (ADL) is proposed, which allows to manage issues like availability requirements, mobile code, security, and replication processes. Aspects of this ADL, Con Moto, are exemplified with a case study from the insurance sector.

1 Motivation

We can observe in our daily life, that existing and emerging communication technologies and the growing availability of powerful mobile communication and computing devices are the enablers for software systems with mobile parts. Regardless whether we look at fleet management systems or point-of-sale systems, all applications with mobile parts have in common that the point where the information technology is used is moved towards the point where the business process is really executed.

Although these mobile applications bring new requirements with them, we usually apply development methods and modeling methods used for 'conventional' applications. In this work, we will show how the development of mobile applications can benefit from an approach explicitly addressing mobility issues.

The remainder of this paper is organized as follows. In section 2 we motivate the necessity of an architecture description language (ADL) for mobile systems. Section 3 contains an application of this mobile ADL to a case study from the insurance sector. Finally, the conclusion in section 4 rounds out the paper.

2 Con Moto as Mobile ADL

As discussed in the previous section, more and more software systems are determined by some sort of mobility. In the following we call a software system part which is used at different locations (without these locations being known in advance) a "mobile component". We call a business process mobile, if its execution depends on a mobile component. For a more precise definition of the term "mobile" we refer to [RPM00].

If a software system contains mobile components, its management has to deal with a number of tasks and challenges which do not occur in the management of systems without mobile components. Examples of such tasks and challenges are:

- A functionality which is crucial for a number of mobile business processes has to be available

*The chair for Applied Telematics / e-Business is endowed by Deutsche Telekom AG.

on client systems. Usually it is not sufficient to keep it running at a central site and to allow mobile access to it. Thus, the availability requirement for this functionality may clash with service level agreements which can be obtained for telecommunication infrastructure.

- If huge amounts of data are needed in a mobile business process, then it is usually necessary to provide a only readable copy of these data at the mobile location. This immediately raises the question whether or not these data can be downloaded when needed or if they should be installed at the remote location in advance.

- If certain functionality are to be handled confidentially, then this may collide with mobility requirements, simply, because components implementing this functionality should not be made available at local devices.

- Updates and patches of mobile components demand either precautions from a software update process point of view (e.g. to register which versions of which components are installed where) or they require that mobile components are not made persistent at mobile locations.

Further challenges exist in the context of compatibility between mobility on the one hand and other non-functional requirements (robustness, scalability, security, traceability, etc.) on the other hand. Summing this up, requirements with respect to the mobility of software systems determine not only substantial parts of the software processes for the development and maintenance of these systems, but they also have an important impact on the architecture of these systems.

Our approach to an architecture centric software development of mobile software systems is to express all aspects of software systems' mobility in a software architecture. This software architecture is described from different views (application view, software view, system view [GT00]). All these view onto the architecture should clearly describe which components and which business processes are mobile, which kind of mobility applies to them (ranging from mobile web-based access to central systems to mobile code in the sense of [HBSG99, WBS02]) and which side effects a change of mobility properties could have.

Our approach is based on the architecture description language Con Moto. Describing all three views onto a software architecture in terms of Con Moto means to grasp all aspects of mobility. A Con Moto-based description of a software architecture is used for different purposes:

- understanding the architecture of the system, paying particular emphasis on mobility issues

- using the architecture description for configuration management and change management tasks

- scheduling work including tasks and packages related to infrastructure support enabling mobility

- analysis of the architecture, in particular for the identification of performance bottlenecks and incompatibilities between mobility requirements and non-functional requirements.

In the next section we will introduce the key elements of Con Moto along the lines of a concrete example. This examples covers a point-of-sale software system for insurance agents. We put emphasis on those elements of Con Moto which focus on mobile components and mobile business processes. We are aware of the fact, that this introduction does not give a complete overview about Con Moto, but it hopefully will provide sufficient insights into the role a Con Moto based architecture description plays in an architecture centric development of mobile system.

During the introduction of Con Moto we refer to other ADLs (a good overview can be found in the paper of Medvidovic and Taylor [MT00]) wherever appropriate in order to point out what the benefits of considering mobility as first class property of software systems look like.

3 A Case Study: uniPOS

In the insurance sector, insurance and financial products are more and more sold actively to the customer instead of being bought by the customer. From the viewpoint of insurance agents, this produces new requirements like rapidity, actuality, and

demand for diversity (for services as well as for products), which have to be fulfilled by sales supporting systems.

Existing sales support systems are mostly realized as fat client applications, which are extensive and allow autarkic (offline) execution. However, their architecture implies a number of problems (e.g. replication issues) and fails to meet the needs described above.

Since a mobile system is predestinated to provide the above-mentioned sales support, we decided to create the case study *uniPOS*.

The uniPOS system supports different target groups. Tied intermediaries, brokers, multiple agents, policy holders as well as staff members of the insurance company are provided with an optimized view on stock data and support for their daily business processes by

- integration of different lines of business in one view,
- support for different sales channels, and
- online deal- or case-closing transactions.

Additionally, uniPOS provides fast introduction of new products and avoids multiple plausibility checks.

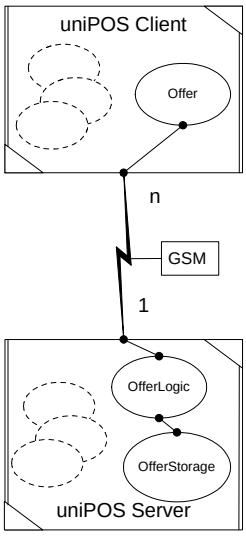

Figure 1: Offer Component in uniPOS

Figure 1 shows a part of the uniPOS system, expressed in Con Moto. There are two devices, namely the *uniPOS Server* and the *uniPOS Client*, both represented by rectangles with triangles in their edges. On these devices, there are several components, namely *Offer*, *OfferLogic* and *OfferStorage*, which are represented by ovals. The lines between the components and the components and the devices depict connectors, meaning that e.g. *Offer* can use services from *OfferLogic*, since the two devices are connected by a attributed connector representing a GSM telecommunications channel.

With this representation (and the formalism behind) it is possible to judge about the operational availability of e.g. the *Offer* component. If we request a high availability, a Con Moto checker can show the clash between the GSM channel (which is not always available) and the requirement of availability. Hence, availability requirements for systems and subsystems can be answered by analyzing a system depicted in Con Moto.

Since we want to use Con Moto during the component based design of mobile systems, the need for a graphical representation is clear. Although the ADL *Wright* [AG97] can model distributed systems, it lacks a graphical representation. Other ADLs allow the graphical modeling of distributed systems. *Darwin* [MDEK95] is based on the pi calculus. From these precise semantics, Darwin's strength lies in modeling hierarchically constructed systems and distributed systems. *Rapide* [LV95] provides comprehensive modeling, analysis, simulation and code generation capabilities. However, as it is strictly event based, it clashes with the service-oriented nature of component-based systems. *C2SADL* [MRT99] is an ADL also suitable for dynamic systems.

However, none of these ADLs implies constructs similar to our notion of the device and the attributed channel. These ADLs also do neither support such constructs directly nor support non-functional properties, which could be used to extend the existing ADLs in a way to make them suitable for mobile systems.

Weaves [GR91] addresses issues that seem to be related to mobile systems like dynamic rearrangement. However, the concept of the stream-like weaves do not fit into the component-based approach that is used for mobile systems today. A recent contribution to the field of ADL research is *xADL 2.0* [DvT02]. It is an XML-based ADL that is highly extensible. Although it does not support

the mobility issues we want to point out, that ideas from it can be valuable contributions to Con Moto.

We now continue with our example. Before, we detected a system which fails to meet the availability requirements. Con Moto also supports the modeling of a system variant that is superior at this point.

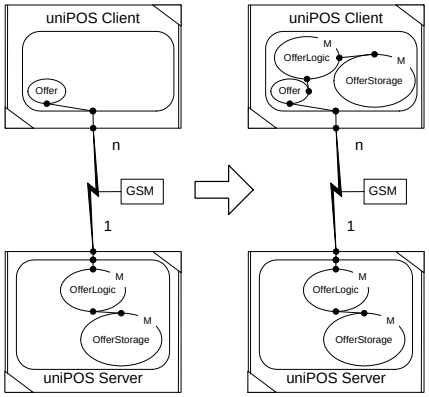

Figure 2: Mobile Code

In Figure 2 two equivalent variants of uniPOS are depicted that make use of mobile code. The components that shall be mobile have been marked with an 'M'. But to check whether a system configuration with this kind of mobile component is possible, we need to introduce the notion of *execution environment*, depicted by a rounded rectangle. This indicates that there are different execution environments like application servers, lightweight application servers etc., that may be compatible in a way that they allow mobile components to move around the system in order to cope with availability problems. With the specified mobile components, the availability requirement for the component *Offer* can be satisfied, as the mobile components may travel from the server to the client.

Furthermore, with this representation Con Moto allows the discussion of security aspects as it is able to determine which components on which execution areas and hence on which devices a component will be available.

However, just making a component mobile is not sufficient: in our example, the *OfferStorage* component has to work on some data from a database. This directly leads to replication problems in the mobile case. Hence, we have to express in our architecture, where replication has to be performed. Figure 3 shows two equivalent architectures this for the database with offer data. The replication areas have been drawn as dotted rounded rectangles. The dotted line between them indicates that they belong together and that there is a replication relationship between them.

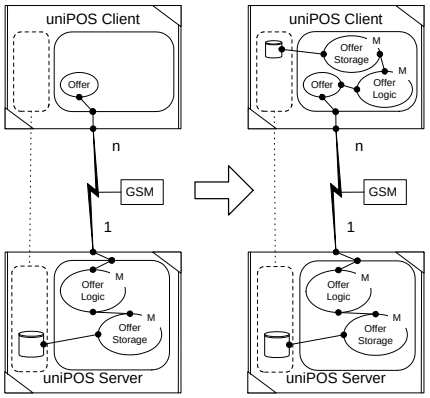

Figure 3: Replication

In Figure 2 and in Figure 3, two equivalent variants of the same system have been displayed. In both cases, the left variant (with the components *OfferLogic* and *OfferStorage* on the server) is sufficient, as the "mobile" cases (the right sides of the figures) are covered as well due to the specification of the execution environments and the replication areas.

4 Conclusion

In this paper we discussed the idea of an architecture description language matching the requirements of mobile architectures. Due to the increasing mobility and distribution of business processes and supporting software systems, the mobility aspect deserves emphasis and this emphasis should be reflected in the architecture description. Even though our ADL Con Moto is no formally defined

yet and even though the language itself is still subject to minor updates, it turned already out to be useful in managing the complexity of mobile architectures. We consider this and—even more—the strong demand for architectures whose mobile aspects are easy to recognize as encouragement on the further way to full syntax and semantics definition for Con Moto.

References

[AG97] R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.

[DvT02] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. An infrastructure for the rapid development of xml-based architecture description languages. In *Proceedings of the 24th International Conference on Software Engineering*, pages 266–276. ACM Press, 2002.

[GR91] Michael M. Gorlick and Rami R. Razouk. Using weaves for software construction and analysis. In *Proceedings of the 13th international conference on Software engineering*, pages 23–34. IEEE Computer Society Press, 1991.

[GT00] V. Gruhn and A. Thiel. *Komponentenmodelle* (in German). Addison-Wesley, 2000.

[HBSG99] Ophir Holder, Israel Ben-Shaul, and Hovav Gazit. Dynamic Layout of Distributed Applications in FarGo. In *Proceedings of the 21st international conference on Software engineering*, pages 163–173. IEEE Computer Society Press, 1999.

[LV95] D. C. Luckham and J. Vera. An Event-Based Architecture Definition Language. *IEEE Transactions on Software Engineering*, 21(9):717–734, September 1995.

[MDEK95] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying Distributed Software Architectures. In W. Schafer and P. Botella, editors, *Proc. 5th European Software Engineering Conf. (ESEC 95)*, volume 989, pages 137–153, Sitges, Spain, 1995. Springer-Verlag, Berlin.

[MRT99] Nenad Medvidovic, David S. Rosenblum, and Richard N. Taylor. A language and environment for architecture-based software development and evolution. In *Proceedings of the 21st international conference on Software engineering*, pages 44–53. IEEE Computer Society Press, 1999.

[MT00] N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, January 2000.

[RPM00] Gruia-Catalin Roman, Gian Pietro Picco, and Amy L. Murphy. Software Engineering for Mobility: A Roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 241–258. ACM Press, 2000.

[WBS02] Yaron Weinsberg and Israel Ben-Shaul. A Programming Model and System Support for Disconnected-Aware Applications on Resource-Constrained Devices. In *Proceedings of the 24th international conference on Software engineering*, pages 374–384. ACM Press, 2002.