

## ***Code Generation Network Interview with Axel Uhl***

**CGN:** Thanks for agreeing to this interview Axel. I wondered if we could begin by asking about your background, how long have you been involved in software development? What types of project have you worked on?

**Axel:** I first got my hands on a C-64 around 1984 or 1985, probably like many of us. Playing with the BASIC interpreter, extending it and deciding I'd need to write my own assembler. That's when I upgraded to my first PC. My first money I made with IT—while still in school—by doing some MS Office-based consulting. During my compulsory military service I joined the programming center of the German Air Force for air defense where we did some cool stuff on a DEC PDP 11/70 and on HP-UX workstations. I decided I wanted to know more and studied computer science in Karlsruhe.

In 1995 I got my first “real” job at Asea Brown Boveri (ABB) Corporate Research in Heidelberg. Besides introducing good software engineering processes and practices into ABB's business divisions, one of the more intriguing projects certainly was one of the first industrial three-tiered Java applications. We used the technology for a distributed logistics application that enabled ABB to better track their parts for power plant constructions. ABB sponsored a PhD programme that I was happy to join.

In late 1999 I decided that I wanted to see a not-so-huge company from the inside, one with more of an IT focus. I went to Interactive Objects Software GmbH where we did a lot of work on model-driven software development. Besides finishing my PhD work around application architectures for Internet search, I believe I can say that our team achieved some remarkable advances with regard to the process and technologies for the engineering of model transformations. We understood the development of model transformations as a problem that equaled that of developing other software applications in various ways. Hence we decided to “eat our own dog food” and use a model-driven approach for the development of model transformations. This turned out to work very well and gave us an edge over competing approaches. However, the overall business model around tooling unfortunately did not work out as well as we had hoped. The case proved that tooling is generally considered a commodity that—if not supported by an application platform or other services—is a tough sell by itself.

So back to the large enterprise, but this time with a clear software focus: SAP is Europe's biggest and most successful software firm.

**CGN:** You're now part of SAP's Office of the CTO. Can you tell us a little about your work with SAP? How and when did you get involved? What's your specific responsibility within the team?

**Axel:** I took the job in Walldorf in June 2004. I guess I got hired mainly because of my background in model-driven development, which already was a hot topic at that time for SAP. My first observation after I arrived was that over the decades the company had grown a substantial heterogeneity in their development tools and repositories environment. This had led to a number of challenges especially with tool integration. It also caused some redundancies in the set of infrastructure components,

especially in the area of repositories and common services on top such as model transformation and code generation support. We therefore launched a project; called “Modeling Infrastructure” (MOIN) that aims to provide a consolidation platform for SAP’s modeling tools.

Meanwhile, as you mentioned, I’m with the Office of the CTO. Our mission is to understand, consolidate and evolve the architecture of our suite of products. As you may imagine, already *understanding* what SAP developers have created since 1972 is a rough ride at times, not to mention the consolidating and evolving part.

Given my history and background, my natural focus inside the Product Architecture group is on tools, repositories, development methodologies and programming models. This results in an interesting mix of shorter-term and mid to long-term projects. In the short term we are mostly trying to help the various tools development groups in the MOIN and Eclipse alignment. In the medium-term we believe that there is room for improvement especially around the programming models that developers see and use when they target the SAP platform. The platform is evolving from what came to be known as the good old R/3 kernel with its ABAP programming language into a world of service enablement, business objects and process and task integration modeling. Obviously, you want to grow your programming model in lockstep with your platform, but that requires good tool workbench infrastructure. Our short-term projects are essential to support the medium-term plans.

In the longer run our group is looking into more visionary things that have to do with our runtime architecture. We need to understand better how we can leverage innovations in the area of enterprise architectures to further improve from where we are with a rock-solid and scalable, yet somewhat aged and entirely proprietary ABAP application server. With this aspiration, we are looking into advances in hardware, virtualization, database technologies and several other hot areas and strive to factor them into our long-term architectural vision.

**CGN:** What’s the history behind the MOIN project? How and when did development start?

**Axel:** That was in the fall of 2004. It first started in research mode but now has substantial manpower behind it and has turned into a regular development project. We originally contemplated using EMF, also because we aim to move our development tools to Eclipse over time. However, we identified several architectural issues with EMF that make its application to large-scale enterprise modeling difficult. At that time we were—mostly for legal reasons—unable to sign an Eclipse contribution sheet and hence could not contribute to the development of EMF to make it suit our needs. That was one of the main reasons to build MOIN ourselves and use open standards such as MOF, XMI and JMI. Now, two and a half years into the project, I believe we have done a good job in providing such a complex piece of technology in a usable form to our tools developers. MOIN is by far not finished, but in certain areas we believe that it already provides us with much better services than we could have got from EMF.

Very recently, SAP has signed an Eclipse contribution sheet. Now we can start contemplating helping in EMF development to let EMF and MOIN converge in the long run, although technically that will not be an easy exercise. There are some

fundamental mismatches between the two architectures that make bridging them a true challenge.

**CGN:** Who is using MOIN and what sort of problems are they tackling?

**Axel:** First and foremost we have a large internal customer base. SAP has about 10,000 developers, and they constitute a very important audience for us. If we manage to improve their development efficiency significantly, this will help SAP to get faster and better in rendering business understanding in software.

At the same time, as SAP moves towards providing a business software *platform*, providing first-class tools to our partners, ISVs and customers who want to build on top of this platform is key to success for this kind of strategy. The good news is that if we manage to get our own developers' efficiency improved then that same set of tools will also help our external customers. "Eating your own dog food" is very important in the tools area.

**CGN:** Can you share the immediate and longer-term plans for MOIN?

**Axel:** The tooling infrastructure work in some sense parallels what SAP had to do in the application server space. SAP's main focus was in selling the world's best business software. The question was how much of the stack SAP would have to and want to own. For the classical R/3 the decision was mainly to use existing operating systems and databases, abstract from them adequately and build the application server according to a proprietary programming model. This clever choice certainly facilitated SAP's incredible success story. In the future the game will be more about leveraging commoditization in the IT stack and knowing where to differentiate.

For the tooling part I see many analogies to the overall picture. We leverage Java and Eclipse because we think they have matured sufficiently. However, for the repository technology we believe that EMF is not (yet) capable of carrying our load and so we build our own infrastructure in this area. For some period of time this may give SAP an edge over competitors struggling with the shortcomings of today's repository commodities. In the long run I see this tooling infrastructure becoming a commodity also for enterprise-class use and nothing that SAP will particularly differentiate on. Our challenge then will be to anticipate the development of things. We have to play the Open Source game right, especially in those areas that we expect to become commoditized. This has to do with striking the right balance between keeping a proprietary edge, aligning with standards, co-innovating jointly with the rest of the industry and managing the company's intellectual property in the right ways. Personally, I believe that we should soon start to contribute technologies such as MOIN to the Open Source communities.

**CGN:** What's unique or controversial about your approach? Are SAP's competitors working on similar products?

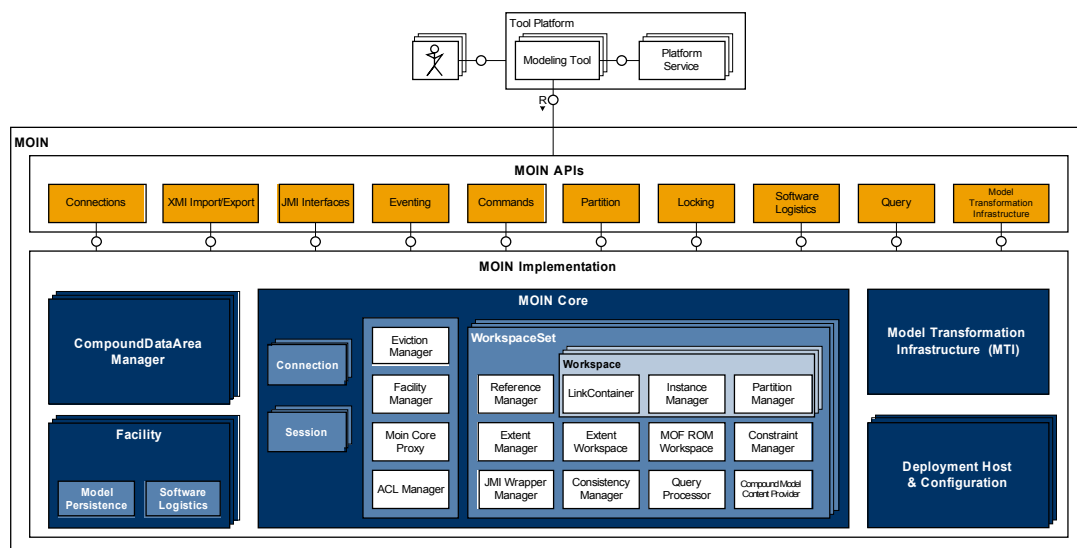
**Axel:** Certainly, the fact that it's not EMF and does not use EMF interfaces is the most controversial part of MOIN. We know that many people in the industry are exploring EMF and have come to similar findings about EMF's shortcomings. Some try to alleviate some of the pain by replacing the way EMF does model persistence,

e.g., introducing Hibernate or TopLink, doing an O/R mapping for the model elements. We believe that won't cut it. Several existing tools that use EMF implement their own way of doing model persistence (e.g., the Eclipse Web Tools Platform, WTP), conflicting with the O/R mapping approach.

Interestingly, IBM recently acquired Unicorn, maker of a model repository with an architecture that is not really close to that of EMF. This may serve as an acknowledgement that EMF is not the answer to all questions on model management. On the other hand, the initial setup procedure of MOIN and the process for developing and deploying metamodels still needs improvement. EMF is very good at that and makes it very easy for developers to get used to it. We believe, though, that we can achieve a similar degree of usability soon.

**CGN:** Can you walk us through the Modeling Infrastructure Architecture?

**Axel:** Figure 1 gives an overview of the architecture. The central component is an in-memory model cache, arranged in sets of workspaces that can be layered and share parts between different user sessions. The cache funnels model data coming from one or more so-called *facilities* and dispatches queries to them.

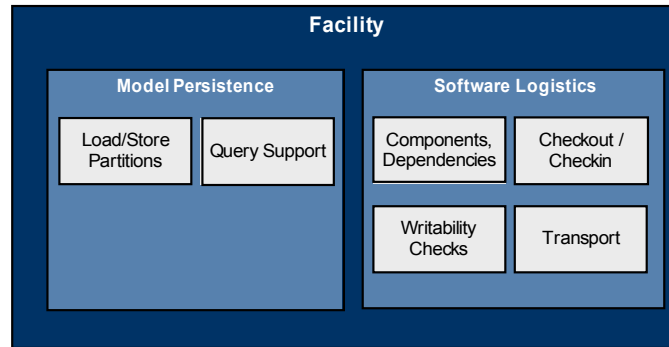


**Figure 1: Architecture outline for SAP's Modeling Infrastructure**

Some aspects of those facilities are outlined in Figure 2.

The so-called *deployment host* decouples the repository's core from its environment and allows for various deployment scenarios, such as an Eclipse runtime or a Java EE 5 application server. You could even run MOIN in a standalone Java VM and still benefit from most of its services.

From the choice of services we have defined on top you can see that we haven't just built a repository backend for server-side operations. Instead, we have ensured that tools can build right on top of them. At the same time, the core architecture allows it to scale very well and run in various different setups.



**Figure 2: Pluggable facilities for MOIN**

**CGN:** How would I get started using the product? How could I migrate to the product from a legacy code base?

**Axel:** Keep in mind that we have not yet made MOIN available to the general public, even though there are some thoughts on doing so in the not too far future. But assuming you had a copy of MOIN, today you would start drawing your metamodel using the MOF 1.4 standard and good old Rational Rose with the Unisys add-in for MOF export. You import what you get out of Rose into MOIN as a set of model partitions from which MOIN generates the respective JMI interfaces and implementations. With a single line of code of the form

```
repositoryFacade.importMetaModel("moin/meta/yourmetamodel");
```

you're in business. The metamodel gets deployed and registered with the MOIN infrastructure and from that point on becomes usable in your application or tool. You can then start to explore the MOIN APIs by obtaining the first package proxy for your metamodel package:

```
RefPackage refPackage=repositoryConnection.getPackage("ngpm");
```

I know that a lot of EMF metamodeling is still being done based on Rose for which EMF has a decent importer. Migrating an EMF Ecore metamodel over to MOIN can be done with little effort. It's the syntactical differences in interface conventions, model element identity management, life cycle characteristics, command and event frameworks used by the tools on top that can make migration of existing tools difficult.

MOIN does offer support for importing EMF metamodels also from the EMF XMI files, without going through Rose. Additionally, in our lab we already have XSD interoperability comparable to that of EMF where you can create an annotated metamodel from an XML schema and have the XMI import/export interpret the metamodel annotations and read and write XML documents compliant to the original XML schema. One of the challenges in this, obviously, is maintaining model element identity upon repeated document imports.

**CGN:** How long does it take to master the tools? Does it involve a radical change of mindset on behalf of the developers? Are developers the main users?

**Axel:** Keep in mind that so far we are building a tools factory. The tools built with that factory are only now emerging internally and won't hit the market before later this year. Our partners and customers will see these technological advances in two guises. For one thing, they will see better-integrated tools emerge over the coming years that will be easier to use, more consistent in the way they look and behave and tuned to the programming models useful to the creation of software for the SAP business software platform. On the other hand, ISVs and more technical folks will find it attractive to see a standards-compliant repository come out of SAP that not only hosts all of the important SAP application metadata but additionally is easy and convenient to extend and offers its powerful set of services to add-on tools and services.

I think the mindset of everyone who got the gist of model-driven development will find his way around in the landscape we are working on. Therefore, no major mindset change should be required for them.

**CGN:** Are you working on any other projects that may be of interest to our readers?

**Axel:** Most significantly, I'm also involved in some interesting SAP research activities where we explore different programming models and language styles that we deem adequate for our "Enterprise SOA by Design" platform. We believe that finding the right level of abstraction is really key to the efficiency and productivity of the developers and to the quality of the software we produce. Innovation in this area, however, means that we have to account for our massive legacy on the one hand and at the same time factor in the future architectural changes we can already anticipate today. Combining this with a powerful, emerging infrastructure that can carry those new programming models seems a very promising area of work to us.

On the other hand, I'm following some of the stuff that is going on in what many people would call the *Web 2.0* space, in particular the people-networking aspects in the music industry. Having written my own little MP3 player in Java that connects to the last.fm online services to exchange my listening habits I can now benefit from last.fm's profiling and correlation services and have my player suggest playlists for me based on my "musical soulmates" profiles. Being all convenient, useful and yet toy stuff, it does, though, give me a good handle on where this may be going. iTunes, Napster, the iPod craze, Pandora, last.fm, ... I guess this is only the beginning and a tiny pocket of the vast innovation that we see happening out there these days around what started as a wiring of a few computers in some defense research labs in the U.S. Looking at it this way, I think my generation does not have to be too sad about not having graduated in the seventies when the whole buzz started. It may have been easier to become a billionaire, true, but those guys also miss out on some fun parts because most of them by now have already retired. The combinatorics and darwinisms have created a truly fascinating mix of opportunities out of what they sowed, many of which only a few of those forward-looking people at Apple may have anticipated. I'm curious what the next years will bring, but I'm pretty sure that it'll be an interesting ride.

**CGN:** Thanks for the interview Axel.

Axel is a keynote speaker at [Code Generation 2007](#) where he will talk about “*Model-Driven Development in the Enterprise*”.

© Code Generation Network, Axel Uhl 2007

