Value-based case study #10

GEVA: re-engineering an already proven payments system onto JEE using a model driven software factory

Management introduction

GEVA Business Solutions (based in Aachen, Germany) is a software company which specializes in the development, integration and distribution of payment transaction solutions. It is primarily, but not exclusively, oriented to German, Swiss, Austrian and Luxembourg financial services companies for the exchange of payments between each other. Its 50+ customers include Commerzbank, BNP Paribas, DEXIA and the German Federal Ministry of Finance.

Most payments systems have been built in either COBOL or using something like the .NET Framework. GEVA's original solution was no different. In the late-2000s GEVA understood that it needed to reset its solutions if it was both to offer constantly improving capabilities to its customers and at the same time reduce the cost of servicing those customers with their many different interfaces and infrastructures while still delivering an utterly dependable and proven payments capability.

In this value-based case study Uwe Klatt, the General Manager (Geschäftsführer) of GEVA, discusses:

- *the decision processes*
- *the selection of a model-driven, methods-based development partner (Integranova) that generated code for JEE*
- *the experience*
- *lessons learned, plus best practices.*

The business problem

GEVA was founded in 1982 as an offshoot from a local University in Aachen. Initially it focused on mostly hardware solutions for the finance sector before it refocused in the 1990s on delivering software. In 2004 a management buy-out concentrated the company on the provision of payment software applications which GEVA has successfully been selling to financial institutions to support electronic payment (e-payment) transactions between banks.

Our solutions are not, however, for end user customers. They are for the interchange of e-payment instructions and transactions between banks. As such our solutions have to be wholly robust and reliable: our customer banks depend on the throughput, accuracy and availability of our software.

In the late 2000s we started to find banks beginning to ask for Java, especially J2EE (now JEE), based solutions for payments. Payment systems are, as you can imagine, a sensitive area for banks: if these do not work as expected the results can be costly, for all.

Therefore, when the banks were asking about Java, it was more than just about asking for the introduction of a new technology. No purchases by banks would take place unless and until JEE solutions could be shown to come from an established, dependable vendor — one that possessed the credibility the banks would need in order to buy. This 'concern' is one reason why most banks are so slow to change any of the core aspects of their systems.

Technology requirements and implications

At the time that this interest in Java-type solutions emerged, most payment solutions were written in either IBM COBOL or using a framework like Microsoft's .NET. Furthermore, each bank had, and still has, its own distinctive combination of interfaces and infrastructure.

In the past this meant GEVA had, essentially, had to support at least one 'programming technology' (COBOL or C++/.NET) and sometimes both. Often these would be used in different combinations with different interfaces and communicating with different infrastructures within each customer bank, never mind between banks. As you may imagine, this was expensive and complex — which was not what customer banks wanted long term.

We listened to what our banking customers were saying. We thought about what they were suggesting and decided we could do what they asked. But to achieve this we decided that we wanted something like a model-driven software factory where we could

introduce a common application base framework that would generates JEE-based applications which could cover 90% of our existing core e-payment functionality.

In addition we wanted to improve our capability to deliver customizable solutions, ones where functions would be changeable or alterable very fast with a minimum of development (and a minimum of re-proving that what had changed was functionally and operationally reliable). If we could do this we would introduce a modern, JEE-based e-payment core product while reducing GEVA's exposure to complexity to only the minimum that was unavoidable — the interconnection to each customer bank's specific interfaces and IT infrastructure.

We also decided that we would:

- use our own domain expertise in payments and e-payments, basing this on our existing functional capabilities
- not to try to develop a new framework-based JEE core on our own: we concluded that we needed a partner that had both the Java/JEE experience but, as importantly, offered a model driven software factory development capability to deliver a descriptor framework that would be functionally provable, re-usable and facilitate change.

Selecting a partner

Finding a partner capable of meeting our requirements — this was late 2007 and early 2008 — was not straightforward. In Central Europe at that time there were few possible partners even supporting JEE at all, never mind understanding the financial services domain: most continued to prefer what the financial institutions already used — COBOL, .NET, C++, etc (this 'deafness' to what the banks were asking for we saw as an opportunity for GEVA).

We did, however, finally identify three possible partners with a potentially suitable, model driven, software factory approach. We performed the usual due diligence and held a 'beauty contest'. We decided that a Spanish company — Integranova, based in Denia and with a close research affiliation with the Universidad Politécnica de Valencia — best suited our needs.

In part this was because its MES (Model Execution System — Figure 1) was already oriented to JEE. In part it was because MES brought to us the structured factory approach we desired, where we could:

- rapidly capture detailed payment process descriptions in the MES descriptor tool (and keep this within the model)
- use those process descriptions to drive the automated generation of JEE code which

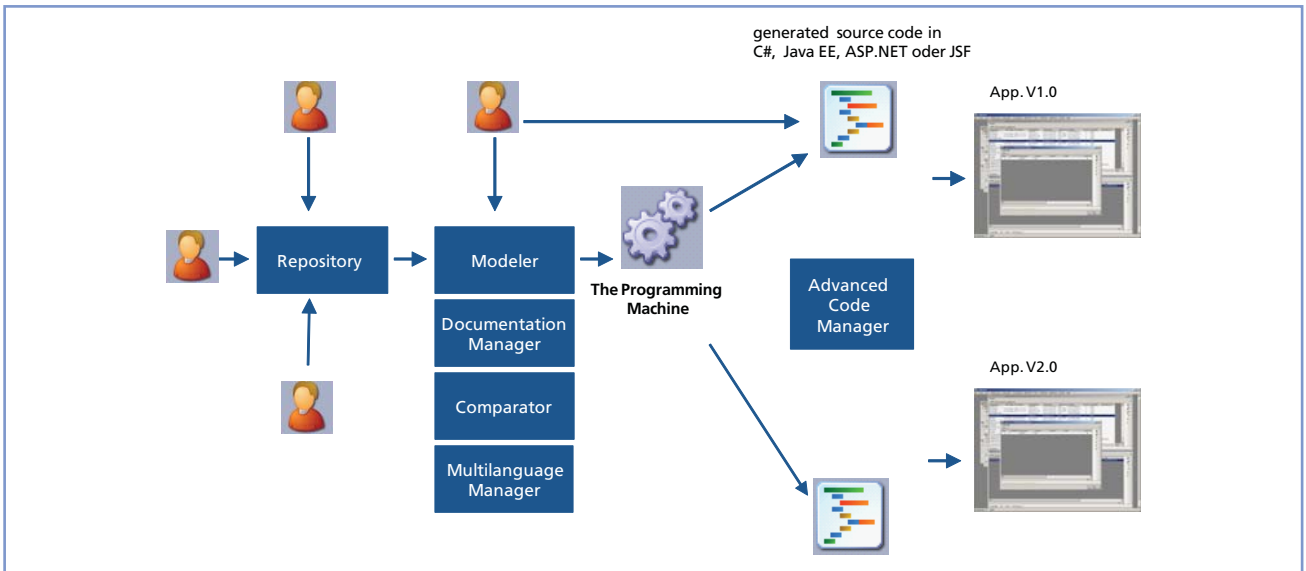

Figure 1: MES — a model driven software factory

would then become the basis of the new GEVA core e-payments platform.

But as much was to do with our decision to choose Integranova was its management willingness to be a partner and to work with us to re-engineer our existing, as well as new, e-payment capabilities into the MES. This willingness was clear from the start — and it has continued.

Progress and customer deployment

In early 2008 we started our partnership with Integranova. We now have about 80% of our original e-payment functions (originally written in C++/.NET) working in JEE via the MES. We hope to have the remaining 10% not yet done (remember 10% is bank-specific and has to be individually written) finished by the end of this year or early 2012.

We already have three banks using our new JEE-based payments platform, even though it is not wholly complete. These banks are using the JEE applications in production and this is providing the necessary confirmation of the needed stability and reliability.

We have many more banks waiting for us to complete, so that they can make similar transitions. Indeed we expect to move most, if not all, of our customers to the new platform in the next 2-3 years.

Going to JEE is, without doubt, giving us the competitive edge we hoped. We are now much further forward in satisfying the banks' wish to have Java-based solutions than the competition. We have also found that the 'agile' development approach works well in the new environment that we have introduced with the MES.

In addition, one of our previous business constraints has been removed. Before working with the MES our e-payments platform was pretty monolithic. Using the MES and working with Integranova we have re-architected and re-engineered our core applications/platform. In so doing we introduce a new flexibility. We have the possibility to 'break out' specific functions and to sell these to new customers, and even to new markets, as separate from the whole platform: this we could not do before.

Let me give you an example: we can, if we want, now separate the SEPA payment function and offer this to corporate customers (a new market for us) so that these corporates can make SEPA payments more easily. This has the potential to be a substantial commercial step forward for us.

Lessons learned

Moving from C++/.NET was not trivial. While GEVA had the e-payments domain knowledge, it did not have JEE experience when we started. This is why Integranova's tools plus Java and process modeling knowledge have been so important.

We have also learned that making JEE perform to the levels our customers need is not trivial with JEE (some customers are processing 1m-1.5M e-payments per banking day). Such volumes stretch JEE. For those relatively small number of our functions that demand such performance we have had to do coding by hand rather than use the MES model generated JEE code (the process descriptions are still in the MES). We are making good progress, however, and we now believe we understand the issues. But what we have learned — both GEVA and Integranova — is that it is not always simple to resolve complex, high performance transaction handling and performance issues in JEE.

On the less positive side, we have always aimed for a single JEE framework. Today, however, we have to have three different implementations of our platform — because of differences in vendor implementations of JEE on the main application server platforms. These different 'implementation flavors' are for:

- the IBM WebSphere application server
- the jBoss application server
- the Oracle application server.

We do not like this. We will have to address it in future development steps. We can hope that Java developments will help us. But we are going to try to identify the key differences between the three JEE application servers and then have Integranova reflect these in the MES so that each specific flavor can be nominated when we go to generate code.

An area where we do want the MES to improve (and Integranova is working on this) concerns how one moves from the modeling process descriptions to the generated JEE code. This works. We must, nevertheless, be able more easily to go from the process flow definitions described in the model to building up the

classes and the interfaces that our customers require. This would significantly improve our effectiveness and productivity for the future.

A fundamentally positive point has been the impact on developers. Compared to the learning curve that developers had for COBOL or C++/.Net before they became productive, it is now much faster and easier for new developers to contribute. The MES has significantly assisted and reduced the time needed for developers to start to contribute. This definitely improves our ability to integrate new people into our business.

Best practices

The most important best practice I can immediately think of may seem obvious. It is that understanding what a customer really wants, especially in terms of process flows and interfaces, is vital. If the understanding is not deep enough, mistakes can happen which are then costly to revisit and correct. Putting sufficient effort into this is essential if you are not to waste the effect that something like MES offers. Looking back we probably did not put as much effort to start with into this as we should have. Now we know we must always do this.

Related to this is that we have found we must spend more time with each customer to understand their flows and interfaces. When we do this properly, the development of that last 10% (the integration to each customer) is both faster and better. It is this last 10% which takes the greatest time (and cost), now that most of our core functionality has been implemented in JEE. This last 10% is the most difficult part to control. We think the best practices that the MES model-driven basis enforces will also help us here eventually (though not as much as is already happening with in our core functionality).

Another best practice that comes with the MES approach is understanding how the emphasis moves from development to focus more on business analysis (the description of the customer's payment process flows) and on architecture (which combines the e-payments core functions with the needed interfaces at each customer bank). The degree of involvement by developers has changed, helped by the automation of code generation. The best practice is to make sure your people change (and do not retain 'old habits') and that you encourage process analysts and architects.

My last best practice is also, in some ways, obvious. Choose your partner with care. While not everything has been perfect with Integranova (what relationship is ever 'perfect'?), Integranova has pretty much done all we wanted. It has grown good at being close to us and at assigning good people to be in Aachen with us. This makes a big difference to how you work together. Distance too often leads to misunderstanding or incomplete understanding — which both partners have managed (mostly) to avoid. The result is proving productive for GEVA over what is now a 3+ year-old relationship.

Management conclusion

GEVA selected Integranova because it offered a model driven software factory approach that generated JEE code. This made sense, not least because GEVA did not have the Java/JEE experience/knowledge necessary to re-engineer its e-payments platform on its own.

By 2012 GEVA will have a JEE-based, dependable core e-payments platform that is readily customizable for individual customers. In converting to the MES, GEVA obtains the ability to reduce (though not necessarily to eliminate) traditional coding.

For GEVA, the model driven factory approach enables it to work at higher levels internally (and also with its customers) to capture its own and its customers' payments processes and interfaces. Though the full commercial benefits have yet to arrive for GEVA, they are imminent as GEVA introduces its re-engineered JEE-based e-payments platforms to its existing customer base and starts to open new business opportunities.