# Embedding a Light Weight Process into a "Heavy Weight" Project

Christa Schwanninger, Siemens AG
christa.schwanninge@mchp.siemens.de

## *Our Organisation*

We work in the research and development department of a big company in Germany that produces electronic equipment ranging from houshold appliances to power plants, in all of which software plays a major role nowadays. Thus we work with customers that are always inhouse only, but they nevertheless vary in the quality of their software development process. Our department's goal is to help our clients cope with all problems that arise in software development. We help on different levels, starting from short-term consulting and writing studies to developing prototypes or join in the development of software.
The latter two are always also an opportunity, or even include the demand to influence the client's development process. Sometimes we are activly asked to do so, sometimes it is useful to accept an existing process. And sometimes we have to accept an existing process but nevertheless try to change it

## *The Customer*

In the project I'm writing about we actively took part in product development. We started helping to define the architecture and went on implementing a small part of the system on our own, in a team of two.
The complete project counts more than 100 developers in 4 different towns (Regensburg, Munich, Dresden and Vienna). The customer produces embedded systems for cars. Programming languages used to be assembler and C, developers were mostly electrical engineers with good hardware related knowledge, the process was rather chaotic. Several groups developed similar products for different customers in parallel. In 1999 they decided to develop a common product line architecture, an OO framework.
For the managers this was a completely new approach and they wanted to assure the success of the project by picking a very strikt „havy weight" process. Despite they made a big leap forward with new technologies and a defined process, compared to what we were used to they were rather conservative. They had to deal with new technologies, new developers and fellow managers, because the project was bigger than the ones they did before. This resulted in some distrust for new, too revolting ideas. The need of management to keep pace with what happened in this big project and the slight distrust made them even more concerned about the process, that couldn't be strict enough.

To modularize the big project into rather independent subprojects was easy because of the chosen architecture. Nevertheless it was impossible to suggest a more light weight process for the single sub projects. But we were sure that this heavy weight process was not our way of working. And we wanted to smoothly transport this message into the project.

We had our own part of the software, which of course had to use other parts of the framework and was used by others, but both usage scenarios were restricted to using DLLs. Our part was to build an extension to a standard middleware to enable inter-process and remote communication.

## *How we proceeded*

Since we were only the two of us that worked on the middleware component we moved into one room and set everything up to be able to follow practices that are nowadays known as XP. But maybe it is not really XP since we didn't follow the description in Kent Beck's book in all the points. We had code ownership and practiced pair programming only in one third of the development time. The reason for this was the different expertise we entered the project with. We both are experienced developers. What we did all the

time was talking about our design and therefore learned a lot from each other and finally were able to understand each others code easily.
We tried to find the subproject that had to use our middleware first, because we regarded the people who worked on it as our first customer. We ourselves intended to play customer for the part we used, which was a tracing utility. We contacted our customers early and were delighted to find out that they were located in the same site as we. We started to invite them for coffee and explained that we wanted to work together with them closely and they could come with their problems and wishes and we would do our best to help quickly. We had very close contact for the rest of the project. They gave us their software to test our middleware and we had many fruitful discussions. They found errors in our code and we found errors in theirs. We managed to implement all the functionality they needed until their software was ready to integrate ours. And we had time enough to write all the documents the strict development process demanded, only not at the time this process expected us to do so.
Since we started implementing really early we had implemented a great deal of the functionality before we were supposed to write e.g. the requirements specification or the interface specification. These documents documented the nearly ready product.

What made it possible to proceed in an XP like style in an atmosphere of distrust and strict control towards the developers? Our advantage was the distance to the overall project leaders. We visited them several times to present our design and the fact that we contacted our customers to talk with them about the requirements calmed them down. The project leaders were overwhelmed by all the work they had to control their process and therefore soon had no time to care for project parts that seemed to run well. They were content to get the documents the process required on time. Only people who worked with us saw our process and told us that they liked it.

We implemented all the functionality expected before the project ended for us. We used the rest of the time to find extensions to our part that would be required after we left the project. Each of us implemented one of the new requirements with the people who had to maintain our part in pair programming. We hope that they will follow our process and carry it further into the organization.

We also tried to be good customers to the people implementing the tracing package. We volunteered to be early testers and use it in our software. We dealt with them as we expected our customers to deal with us. I don't know if they felt annoyed with us at the beginning, but we had a very good relationship at the end. They were not located at the same site, but we visited each other and communicated heavily by phone and email.

## *"Challenges"*

It may sound strange, but we had problems to find our first customer in the big project. The project leaders were completely overwhelmed by the extent of controlling and planning that the project seemed to be chaotic soon. They worked long hours every week, Saturday was a working day for most of them and also the developers who woked on their site. This was true also for other subprojects that followed the process, in which information always run from the subproject to the project leaders and from there back to other subprojects, which didn't really work. A sign for this was the heavy email traffic, which usually was addressed to nearly all the people working in the project and the demand for several status reports to different people nearly every week.
The really useful information flew between the people in the subprojects that got to know each other by instance when visiting the project leaders.

The other annoying thing was to extract and abstract the information needed for the different deliverables from our source code. Since the most documents should have been written before implementation we had some work to make it look like being before implementation. And the strict format of the documents made the writing no fun.

The project leaders demanded that every line of code in the product had to be documented in the source code. We didn't do this upfront since all the documentation seemed too much overhead and we regarded

the code subject to daily changes. For final delivery we had to go through all our code and comment it, which was very boring and we didn't see the use of it, since we were sure that the code was so clear and easy to read that many comments simply disturbed the reader.

### *What we liked*

We were able to build up a very good relationship to the customer. They were happy about our flexible reaction on demands for changes. Even if one of us was on vacation the other one was able to incorporate changes in all of the code. Since our software was integrated very early, integration itself was no problem. Integration was one of the hardest problems for other software parts in the project.
The project was ready on time, the customer was content with it and we had time enough to pass it on to our successors successfully.
We managed to follow our own process and convinced the project participants who worked with us that a software project doesn't need all this controlling to be successful. We tried to convince the project leaders by simply showing that our approach works and I hope we were successful in some cases.