Global Software Development Process Research at Siemens

Matthew Bass, Daniel Paulish
Siemens Corporate Research, Inc.
Princeton, NJ 08540
{ Matthew.Bass, Daniel.Paulish } @Siemens.com

Abstract

Siemens Corporate Research (SCR) is the research and development unit of Siemens USA. The Software Engineering department of Siemens Corporate Research spends much of its time doing consulting for Siemens Business Units. As a result, we have been involved in a large number of software development projects varying in size, complexity, and domain. Many of these projects were developed with globally distributed teams. Over the years, we have identified best practices, and begun to organize these practices into more cohesive set of development processes focused on issues related to global development. This paper describes our experience with experimentation, lessons learned from one specific project, and suggests future steps for global software development (GSD) within Siemens.

1. Introduction

Siemens is one of the largest developers of software intensive systems in the world. With a presence in over 190 countries, it is also one of the most globally distributed. As software products are growing in complexity and the organizations that develop them are also growing in staff size, Siemens business managers are seeking new approaches to get new software products quicker to market, while reducing their overall development investments. One of the strategies that Siemens has adopted is to move some of its software development to low cost countries. The implications of such a decision are not entirely known. The associated risks, required changes in the development process, needed infrastructure changes, and required modifications to the management practices for successful global development are not fully known.

2. Data Processing System 2000

The Data Processing System 2000 (DSP2000) is a software system for acquiring and processing meter data, from electrical, gas, and water meters. The meter data is stored and processed so that billing determinants can be calculated for periodic transfer to a billing system. The billing system generates the bills for energy or resource consumers.

The development for DSP2000 was done at four sites in three countries. SCR staff acted as project manager and lead architect, and developed one component for this project. The product is currently being successfully sold and distributed. This section describes our experience with the DSP2000 project as it relates to GSD, section 3 then highlights some of the lessons learned from our experiences with GSD projects, and section 4 describes planned next steps towards improving the state of the practice of GSD within Siemens.

2.1 Global Analysis

Global Analysis (GA) [1][2] is a technique for analyzing, categorizing and documenting factors that influence the architecture and project management of a system. In the DSP2000, GA was completed early on during the high-level design for the DSP2000 project. Three types of factors were considered; organizational factors, technological factors, and product related factors.

2.1.1 Organizational Factors

Organizational factors may apply only to the project at hand (as in the case of schedule and budget), or can impact every product developed by that organization (as in the case of culture, development site(s) location, and software development process).

Two examples of organizational influencing factors in the DSP2000 project were:

- Technical skills were in short supply, prior products were Unix-based with local user interfaces, and marketing required new products to be Windows-based with web-based user interfaces.
- Time to market was critical. The market was rapidly changing, and it was viewed as critical to quickly get some limited features of the product to potential users so their feedback could be solicited.

Two strategies were adopted to address these organizational factors. In order to mitigate the lack of technical expertise, it was decided that this project would exploit expertise located at multiple development sites, and to invest in training courses early in the development. As a result of the criticality of time to market, it was decided that the product would be released incrementally. In this way, release dates could be met even if some features were missing. Additionally, a design strategy was followed to reuse the current data-acquisition system, and attempt to use third-party components wherever possible.

2.1.2 Technological Factors

Technological factors may limit design choices to the hardware, software, architecture, platform, and standards that are currently available. Technology, however, changes rapidly, and so if it is the case that the architecture has even a reasonably significant lifetime, then it should be designed with this in mind.

Two examples of technological factors that influenced this project were:

- An object broker was necessary for meeting the scalability and availability requirements within a distributed hardware configuration.
- The database system was expected to change over time. Oracle 8 was initially specified, but it was known that new versions would become available, and some customers would prefer other vendors.

Microsoft COM was selected to as the object broker, and a layer was designed in the architecture to abstract the database in anticipation of future database changes.

2.1.3 Product Factors

Product factors include features of a product as well as qualities like performance, dependability, security, and cost.

Two examples of product factors that influenced this project were:

- This product was to be designed as a product line. In order to support a product line architecture, the graphical user interface (GUI) had to be able to accommodate many different types of users for different applications.
- The required scalability and anticipated performance requirements of the system were another influencing factor. The DSP2000 was intended for industrial and commercial applications where thousands of meters would be handled. While it wasn't originally specified for the residential market, where millions of consumers would be required, it was known that this might be a future possibility.

A web-based GUI was select to address the needed flexibility. In order to allow for potential unknown market performance requirements, we anticipated that a scalable distributed platform was necessary.

2.2 Project Planning

The DSP2000 software development was planned as a sequence of incremental engineering releases, the first of which consisted of a "vertical slice" of the architecture, which functioned as a prototype of the architecture. The last planned release was the first set of functionality that was sold as a package to a customer.

We found that a six to eight week cycle time for each iteration worked best. Some of the release dates were driven by trade shows, at which time a new release with the latest functionality was required. Particularly in light of our global development, we found that one of the best means of communication was via the system itself. It was difficult to fully understand and discuss the explicit and implicit requirements without an appropriate prototype. The system itself turned into the common language for all involved, facilitated by the web-based GUI.

The planning process itself was complicated by the distributed nature of the project. What ended up working well was to distribute drafts of the proposed schedule and task assignments for each incremental release to the team members. Often, we would get feedback in the form "This feature cannot be achieved in the time frame provided", or "I am planning a vacation during these weeks". A second version of the schedule committing the release dates and feature sets would then be distributed.

Another item that is useful in global project teams is an explicit statement of the overall project goals. An example of such a statement is "Quality will have a higher priority than schedule, which will have a higher priority than functionality." Such an explicit statement helps project managers make the inevitable trade-offs that must be made right before a release. We have found in the past that cultural bias exists that will influence such trade-offs at a local level, unless such an explicit statement of goals exist.

2.3 Project Management

Each development site had a local manager to manage the team members at that site. There was also an overall project manager, and a project manager for each software application package development. As a result there was overlapping management responsibility for achieving the project goals. These managers had to negotiate individual work assignments. In practice, however, most potential conflicts were resolved when the proposed development plan was distributed for feedback.

The chief architect was responsible for decision making and resolving technical conflicts for the application package. Analogous to the overall project manager, the chief architect was the overall technical manager. In practice, both the technical manager and the project manager reviewed key technical decisions.

An engineer was assigned responsibility to each subsystem. This engineer was responsible for the detailed design and implementation of this subsystem.

Project status tracking was done during weekly teleconferences. Each team member was encouraged to report on his development progress and to raise information or issues to be shared with other team members.

3. The Influence of Global Development

While the decision to develop DSP2000 across multiple sites was primarily motivated by the lack of resources with the required technical skills, the implications of that decision were felt in the project planning, project management, architecture, and design of the system.

Communication is a key issue in most projects, but additional barriers to effective communication exist in globally distributed projects. Several strategies were found to be useful in the DSP2000 project in overcoming the communication barriers. Those strategies include:

- Explicitly documented project goals – in the absence of clear direction, local cultural and personal biases are going to influence decisions. The resulting choices may not be in line with the overall goals of the project.
- Incremental development – an incremental release schedule with fairly short cycles helps to facilitate communication, and highlight ambiguities and misunderstandings. While this can be useful in many projects, co-located teams may have options that are not available to a globally distributed team.
- Internationally aware calendar – it was important that weekly teleconferences take place to monitor status, and highlight issues. It was important (and often difficult) that time zones and local holiday schedules be taken into account when scheduling these meetings.
- Well-partitioned architecture – in order to facilitate work break down across multiple sites, the architecture needed to reflect the organizational structure of the project. There needed to be well-defined components or subsystems with understood dependencies for each site. These components or subsystems also needed to take into account the technical skills of the staff at the responsible development sites. As it turned out, the decision to distribute the development globally had a large impact on the architecture.
- Communication of progress – in the DSP2000 project, the Uniform Resource Locator (URL) for the test system was made available for all the team members and their management. This was a big morale boost for the team, since everyone was aware of the rapid progress being made. The result was a much greater sense of team then would otherwise have been possible in a globally distributed project.

4. Current Research Focus

We were pleased with our experience on the DSP2000 project. We feel that many of our approaches were validated based on the success of this project. Ideally the decision to use distributed development teams would result from influencing factors relating to the project in question. More and more, however, this is not the case. Distributing development to low cost countries has become a cost-saving strategy for many organizations. Siemens is no different. It is not clear what the impact of such an approach has on the bottom line. While the hourly development cost may be reduced, extra effort is likely to be spent on project management, architectural design, requirements engineering, and so forth.

SCR is currently in the process of codifying past experience in the form of questionnaires, checklists, processes, and other decision aids to assist in the successful application of global software development. We are attempting to correlate project characteristics with proven strategies in order to better establish criteria for success for given projects.

One area where we are planning additional work is the experimental application of a reference process for GSD. Our process includes best practices from requirements engineering, software architecture design, and organizational patterns. Engineering rules of thumb are used to plan projects, specify the size of software components, the division of responsibilities between a central product management team and remote component development teams, metrics, tools, and operational procedures. The experimental projects are used as case studies to further support the identification of best practices.

We feel that we have a good start in understanding some of the issues related to successfully managing a global software development project. We now need to further substantiate, refine, and transfer our approach to the Siemens operating companies.

References:

[1] Hofmeister, C., Nord, R., and Soni, D., *Applied Software Architecture,* Addison Wesley, Massachusetts, 2000.

[2] Paulish, D.J., *Architecture-Centric Software Project Management,* Addison Wesley, Massachusetts, 2002.