A System Family of Integrated, Distributed Embedded System Products and its Architecture

Ulf Cederling
Växjö University
Ulf.Cederling@masda.hv.se
and
Bengt Lennartsson
Linköping University
bln@ida.liu.se

1 Background

CelsiusTech Systems AB (CTS) has for several decades been active in the field of shipborne and land-based command, control and communication (C3) systems, fire control systems and electronic warfare systems for navy, air force and army applications.

In December 1985, CTS signed two contracts for the Danish and Swedish Navy respectively. This was the formal start of the Ship System 2000 program (SS2000), a product line approach for the development of integrated naval command, control and communication/weapon control systems. It was of vital interest for the company to design a family of distributed systems instead of two specific systems.

During the 80th it became clear for CTS that it was necessary to abandon the minicomputer-based solutions for integrated, distributed solutions. Since SS2000 was intended to constitute a platform for new system products towards the end of the nineties, and as these kinds of systems have very long life cycles, the management made the strategic decision to go for a technology shift in both hardware and software.

Previous C3 project experiences at CTS have shown continuous changes of the requirements, timely integration of system units and cost/schedule overruns. One important issue for CTS was therefore to increase the productivity and shorten time to market for their system products. Software reuse appeared here as a fundamental strategic principle.

In order to achieve their goals, the management decided to modify their software development process, adopt a new design methodology (object-oriented design) and replace the software development environment and programming language used in previous projects. Therefore, SS2000 was designed to meet increasing demands on functionality as well as on different cost aspects (the cost of operating a ship, system development cost, etc.).

The hardware architecture of SS2000 consists of a set of nodes connected through a dual Ethernet Local Area Network. The nodes can contain several processors, so the operational software runs in an environment of loosely coupled processors. This structure allows functions to be switched from one node to another and it also facilitate the handling of new and changing requirements.

Since the start of SS2000 program, new system products have been ordered from, among others, the Finnish, Australian and New Zealand Navies. All these products are very interrelated whether one prefers to talk about software reuse or parallel product

development. They are all based on the same framework and they have at the same time contributed to the maturity of a software architecture for this type of domain.

CTS has now reached one of their goals, i.e. to deliver large (million-plus lines of code), reliable system products within a couple of years with the performance and distribution behaviour known before the start of the project. It is now possible for them to predict cost and schedule based on known requirements for product line configurations.

2 Software Architecture

As systems are becoming more and more complex, increasing demands are placed on the overall software architecture, partly as a basis for satisfying customer requirements, partly as a technical base for development and enhancement of systems.

Architecture can be looked at from several different aspects such as:

- Layers of abstraction
- Layers of dependence
- Modularity
- Configuration
- Physical distribution of hardware
- Physical distribution of software

One important purpose for a software architecture is to offer a set of views of the system and to act as a bridge between different stakeholders. Recently, different view models addressing the aspects above have been described, [1] and [2]. The views of these models highlight properties which address needs related to the stakeholders of a system or system family being under development.

In SS2000, the conceptual and static views are covered rather extensively. One of the aspects above, layers of abstraction, highlights a number of important concepts. One is the replacement of software from any level downwards neatly and with minimum effort. Another demonstrates the need to keep applications separate so that exclusion of one functional area does not affect the functionality of other functional areas.

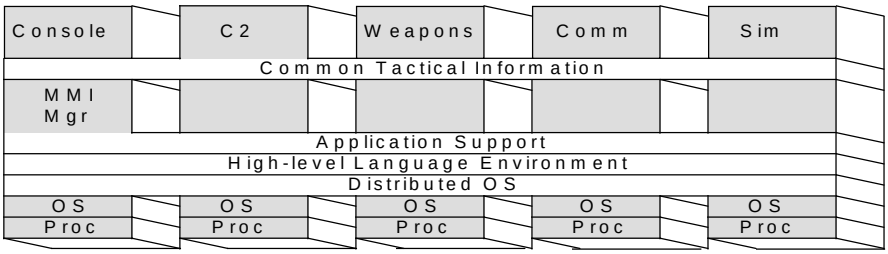

Figure 1: Levels of Abstraction

The different products developed from the platform, show large variation in size, functionality, etc. and they are expected to live over a long period of time, during which changes at the technology level as well as the functionality level will occur.

One such example is changes regarding the target platform and operating system. Most SS2000 systems have been built on Motorola 68 0X0 and OS2000 (OS9). However, it has been possible to port SS2000 software to RS6000 and UNIX with a minimum change required to the application software.

Another important issue for CTS was to establish a basis for future software reuse. In order to handle the complexity of the application area and to facilitate reuse, they organised the software components in a hierarchy and divided the System Family in System Function Groups, System Functions and Ada Units. System Functions are the primary software components and the units of integration and reuse, while a System Function Group is the level used for work assignment to a development team.

Furthermore, since the number of System Functions and Ada units is rather large in the SS2000 family, about 250 and 2 000 respectively, a more implementation-oriented software component level was also needed. The Rational Environment provided such a level in the form of a mechanism called a Rational Subsystem.

The System Functions are classified in different layers of dependencies, according to whether they are application-specific or not. The purpose is to establish a system hierarchy where general purpose components do not depend on services from higher layers in order to attain as much reusability as possible.

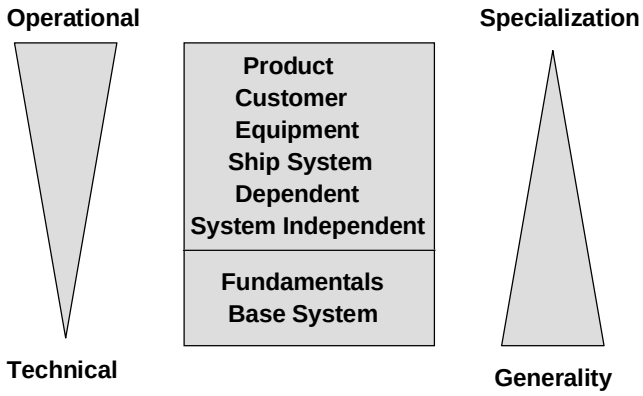

Figure 2: Levels of Dependencies

It seems that CTS has achieved another of its main objectives, i.e. to define a high degree of commonality among the developed system products. On an average, 70% of the system products are reused without any modification of code.

3 SS2000 Related Problems

Up to now a number of system products have been developed within the application domain of SS2000 since the delivery of the first release to the Danish customer at the end of 1989. Much of the architecture framework is unchanged over the years, but there are examples of modifications of the architecture due to different reasons.

For instance, rather early in the SS2000 family of projects, nested relationships between System Functions lying at different layers were discovered. The reason for this was traced back to the import/export feature of Rational Subsystem. A part of the solution to this problem, implied development of new System Functions and System Function Groups, but also transfer of System Functions from one System Function Group to another or splitting a System Function in two parts.

Interfaces have also been changed. Interfaces are supposed to have very long life-times. They are strongly related to questions about architecture stability and the life cycle of the system products.

Furthermore, performance requirements have caused a lot of architectural problems. A demand, not so easily satisfied, is the need of dynamic models as a complement to existing static architectural models.

Another interesting discovery is architectural disturbances due to the fact that customers belong to various cultures, in particular the influence on the software architecture of divergent decision hierarchies at different customers.

4 Team Learning

We believe that the most fundamental success factor is the establishment of an early shared understanding or vision in the design team on what should be done and how. This statement is based on observations from a number of large-scale industrial system design and development projects during a long period of time.

This early shared understanding or vision is something that evolves in the team of system architects during their meetings, something that exists in the union of their minds when they are sitting around the table together. They are in this situation able to discuss problems and each one of them has a different interpretation of and a different contribution to the shared vision. They learn to understand each other and to exploit each others contributions, but in this early phase they are not able to formalise their shared vision or to communicate it to others outside the team by means of some document.

The process that takes place among the different experts in the group is *team learning*. When the team has learned how the system should be built they have established a system structure or a system architecture, i.e. a shared vision of a system composed of interrelated system components or subsystems. At this stage the result may be possible to describe partially to persons outside the group.

However, much of the understanding is still residing solely in the minds of the initial team of experts, the system architects, in the sense that problems can be solved or questions answered only by means of the process taking place when these system architects are gathered around a table in the same room. The team learning process, where each of them can contribute inspired and encouraged by the presence of the others, is the mechanism producing the solutions and the answers.

This view of what is going on when a system structure evolves, when a complex system is born, is not in harmony with the established truths on what system/software design and development is about. The waterfall model, the belief in system development as a sequence of transformations from some requirement specification to a working implementation and integration, formal methods, quality assurance actions, and process improvement procedures etc., do not consider the essence of (non-trivial) system design and development, i.e. the team learning process in the design teams.

We suggest researchers interested in contributing to improved software engineering practise in industry to apply the team learning model when studying (successful) system development projects. Such research can be in terms of observations, case studies, in industrial projects, but also in terms of action research, where the team learning process is supported by different actions. Just as for different kinds of medical treatment, decisions should not be based on assumption and beliefs only. Empirical evidence from comparative studies of the impact from different models are urgently needed.

5 Research Topics

The research topics, inspired by large-scale industrial development projects in our neighbourhood, are the following.

One ongoing research activity concentrates on assessment of the software architecture established in the SS2000 program regarding its stability over time. The architectural platform has now reached the maturity level where experiences should be possible to collect, analyse and document.

Another activity has to do with identification of patterns at different levels of abstraction in the SS2000 system family supporting software reuse as well as system understanding.

The third, more extensive, but at the same time perhaps the most significant research activity, focus on how the system architects are able to achieve and communicate the general understanding of what the future system should offer its stakeholders. In our opinion, the system architecture is the framework or the skeleton where the knowledge and understanding achieved about the application domain is put in place. The architecture is the carrier of the views and concepts of the evolving system or system family, i.e. the means for communication between the people involved in the development of the systems.

References

[1] Kruchten P., “The 4+1 View Model of Software Architecture”, IEEE Software, November 1995

[2] Soni D. , Nord R. L., and Hofmeister C., “Software Architecture in Industrial Applications”, Proceedings of the 17th International Conference on Software Engineering, April 1995