Experiences from an Industrial Software Architecture course

Even-André Karlsson
Q-Labs
S-22 370 Lund, Sweden
Even-Andre.Karlsson@q-labs.se

Abstract

Software architecture is seen as one of the key elements for achieving a high degree of reuse in product lines. To get more focus on architectures, many organizations want to improve their ability to find and maintain a successful architecture. To achieve this they have to involve their most senior designers. In this paper we discuss experiences from an industrial course on software architectures for experienced software designers. We present the motivation for the course, the material selected and the experiences. The course was built up with several exercises where the participants could utilize the covered material on their own system experience.

1. Introduction

Software architectures is a relatively new academic field where there exists a rather limited body of knowledge and courses. It is also an old field in that all software systems have an architecture, and the practitioner's knowledge is often more developed, but probably not so well articulated, as the theoretical knowledge. Thus when building up an industrial course on software architecture it is important to tie the course to the experience of the participants, and also ensure that they can share their experience. We did that through the exercises where the participants applied the material on their own system and organization.

Participants to these courses have been experienced (5+ years) software designers from organizations developing technical software, e.g. telecom systems, where they have a long lived product with new enhancements and variants, i.e. a product family.

The rest of this paper is organized according to the content of the course. Each section starts with the motivation for including this topic in the course, the material selected, the connected exercise and some experiences. The different topics covered were:

- Why focus on architectures?
- Tools for describing architectures
- Architectural requirements and evaluation
- Frameworks and patterns

- Object brokers
- Working with architectures

In the end we discuss some general conclusions and further development. We do not present in detail any of the material covered, but rather discuss the experiences from including this material in the course.

2. Why focus on architectures?

Motivation

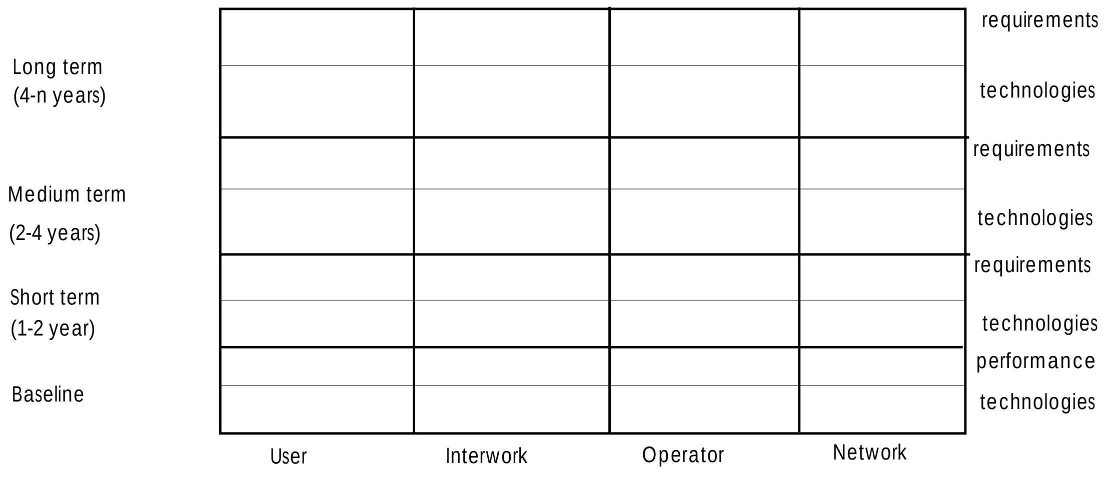
It is important to couple the software architecture to the business strategy, i.e. product strategy, of the company, as the architecture of long lived system can dramatically impact the opportunities of a company. We also motivated software architectures from a reuse point of view, leading to higher productivity and profit.

Material selected

Coupling architectures to product evolution:

- Successful products will evolve, we will sell it to new markets with new requirement
- A good architecture supports product evolution
- Architectures should be determined from product strategies

The following matrix was suggested as one “tool” of analysing the product strategy. Each



column represents a major user or interface to our current system. The requirements of these stakeholders will evolve, which will mean that our system will be impacted if we are going to provide added value to our customers. The matrix can be used to predict what will impact our system in the future. The matrix is divided into rows according to different time periods. The division between requirements and technology are between what the user wants to use the system for, e.g. video conference (for telecom systems) and what type of technology will be available for us to use, e.g. ATM.

We also stated several questions related to the product strategy:

- Who determines product strategies in your company, e.g. marketing, product management, system management?
- How and when are product strategies determined, used and updated?
- Is there a product life cycle plan with the major stakeholders?
- Are requirements and technologies distinguished?
- Who is responsible for technology watch?

Exercise

There are two exercises in this section:

- Make a product strategy for your system.
- Analyse and suggest how to your organization should work with a product strategy.

Experiences

The purpose of this section was to put the software architecture into a bigger perspective, as we make an architecture for specific purposes, and if we don't have those purposes clear it is difficult to come up with a suitable architecture. The experience is that few companies have a clear product strategy (except for the next release), and that the participants found both exercises difficult, but necessary. Also the audience for attacking these issues needs to be broader than just software designers, as they usually only take in requirements from other parts of the organization.

3. Tools for describing architectures

Motivation

This section covers how to describe architectures.

Material selected

In this section we mainly used two sources. One was Soni's four views on architecture [Soni95], i.e.

- Conceptual

- Module interconnection (functional decomposition and layers)
- Execution
- Code

The other source was Shaw and Garlan’s [Shaw96] with

- Architectural styles
- Component types
- Interaction
- Distribution and concurrency

We also added a discussion on other architectural “features” like:

- Fault tolerance
    - Duplication
    - Suspicion
    - Isolation
    - Recovery, e.g. restarts
- Dynamic change, i.e. changing the software, either code or data, during operation as is done in telecom systems
- Execution priority for lie-critical systems, e.g. during melt-down for a nuclear reactor.

Exercise

Also in this part we had two exercises:

- Apply the four different views on your system
- Describe your architecture in terms of style, components and interaction.

Experiences

These exercises were simpler for the participants than the first two. They discovered that for their system the different architectural views were not clearly separated, in particular the functional decomposition, execution and code architecture were the same in one example. Also some systems did not have a unique architectural style, e.g. it could be mainly data-flow but with elements of data-centric. In this case it was important to clearly understand how to use the different styles when implementing new functionality.

4. Architectural requirements and evaluation

Motivation

To be able to come up with a good architecture we need to formalize the requirements on the architecture from the product strategy and to evaluate different alternatives. The reason for having this module here is that it ties together the two previous ones, which can stand by themselves.

Material selected

In this part we discussed:

- Architectural drivers from [Clements95], where we discussed that the non-functional requirements were found to be the determining factors in choosing the software architecture.
- Gilb's attribute specification as a method to specify non-functional requirements, [Gilb88].
- How to evaluate architectures,
  - Determine architectural requirements, both technical and from product strategy
  - Prepare a set of concrete scenarios with which to assess the architectural requirements, e.g. new functionality, non-functional requirements, changes in technology
  - Evaluate the degree to which each architecture provides support for each task
- Two cases of architectural evaluation (KWIK [by Parnas and Mobile Robotics by Schumacher, both from [Shaw96].

Exercise

The exercises for this area were:

- Specify the driving non-technical requirements for your system using Gilb's attribute specification.
- Prepare a set (3-4) of test cases (scenarios) to assess your current architecture based on your product strategy.

Experiences

This section was also more difficult for the participants, as they were not used to formalizing the requirements on their system in this way. The architecture was in most cases just something that appeared based on an informal process, without any good coupling to a product strategy or thorough analysis of non-functional requirements. Note that this does not mean that any of the discussed architectures were bad, nor that they did not take most of these things into account, but the discussion had never been explicit.

5. Frameworks and patterns

Motivation

The section on frameworks and patterns was included to give more concrete ideas on how object-oriented concepts could be used to realize architectures.

Material selected

For framework we covered what the purpose and idea of a framework is, how we use object-oriented concepts to realize it, and two example frameworks in some detail.

On patterns we used some examples from Gamma's book also focusing on how to describe patterns, [Gamma94].

Exercise

The two exercises in this section was:

- Determine a small part of your system where a framework could be useful and sketch a possible framework.
- Which of the design patterns discussed are applicable to your system, why and why not? If possible try to find some reoccuring design patterns in your system not covered by the discussed patterns and describe them as patterns.

Experiences

Since this part presupposes a certain knowledge of object-oriented concepts, this might have to be covered first. We have also tried to make a non-object-oriented version of this section, i.e. not referring to specific object-oriented terminology, but more general on indirection of function calls, and selecting less object-oriented patterns.

6. Object brokers

Motivation

Object brokers are another concrete technology that influence many system architectures.

Material selected

We briefly described the purpose of Corba and DCOM discussing the different basic object services and common facilities. We also covered the distinction between these two object brokers.

Exercise

The exercise in this section was to try to analyse their system in terms of the Corba architecture and see what the different basic object services and common facilities could give their application.

Experiences

This part required substantial time for the participants to understand the object broker architecture and what it provided, and mapping that to their system on the fly is not easy.

7. Working with architectures

Motivation

If a company is going to be successful with it's architectures over a longer time, they need to establish a way to develop, maintain, enforce and enhance the architecture and the knowledge about using it, i.e. a process and an organization.

Material selected

In this section we discussed issues around organization, competence, process and documentation for:

- Architectural requirements
- Design and evaluation
- Evolution and control
- Sales, manufacturing and maintenance structure

The later question is particularly important as a product architecture should impact the way a company market and sell their products.

In addition we cover competence management, e.g.:

- How are new "architects" recruited, developed and maintained?

Exercise

The exercise in this section was to try to analyse their own organization's current status on these areas, as well as come up with suggestions for how it could be improved.

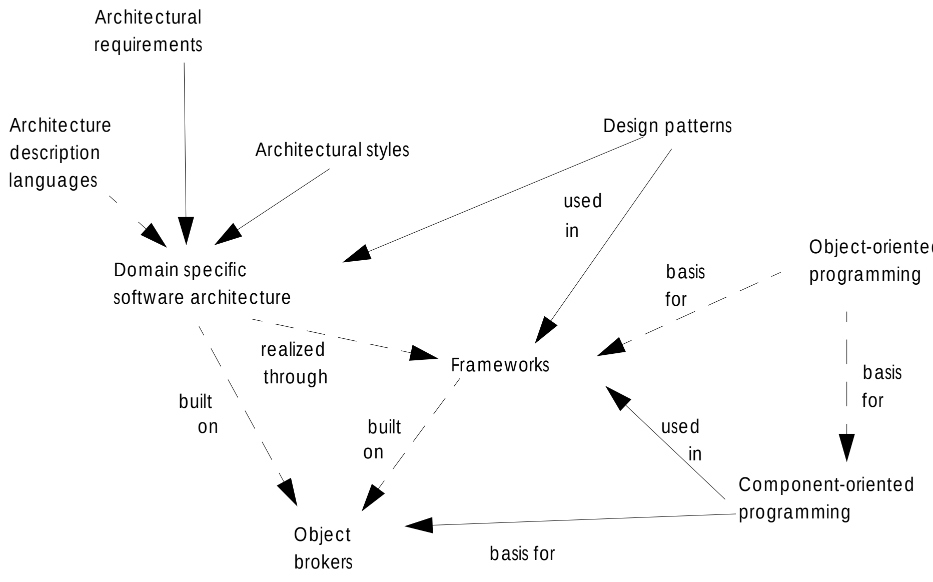
Experiences

Since this courses were mainly for software designers, they had not first hand experience from some of the impacted areas, but in any case they found the exercises interesting, and many good suggestions tend to come up.

8. Conclusion

We have not included anything on how to make an architecture, as this is currently not well understood. Usually people have several meetings where they present and discuss different alternatives. What we have tried to provide is a set of tools that can be used to structure this discussion, and to ensure that all aspects are properly analysed and covered.

We used the following figure which tried to put the different parts in relation (partly adapted from Jan Bosch):


Architectural requirements
Architecture description languages
Architectural styles
Design patterns
used in
Object-oriented programming
basis for
Domain specific software architecture
realized through
Frameworks
basis for
built on
built on
used in
Component-oriented programming
Object brokers
basis for

In general the courses have been well received, although sometimes the amount of material covered can be too much depending on the background of the participants. The exercises where the participants try out the ideas on their own systems and organizations are also seen as very useful. During the exercises we let the participants work in groups of 2-3 people, and we relay heavily on supporting them through the exercises. At the end of each exercise each group present their result with a brief discussion.

References

[Clements95]	“Understanding architectural influences and decisions in large system projects”, Paul C. Clements, Proceedings of the First International Workshop on Architectures for Software Systems, ICSE 17, Seattle, April 24-25, 1995
[Gamma94]	“Design Patterns - Elements of Reusable Object-Oriented Software”, Gamma E., Helm R., Johnson R., Vlissides J., Addison-Wesley 1994
[Gilb88]	“Principles of Software Engineering Management”, Tom Gilb, Addison Wesley, 1988
[Shaw96]	“Software Architecture - Perspectives on an emerging discipline”, Shaw M., and Garlan D., Prentice Hall, 1996
[Soni95]	“Software Architecture in Industrial Applications”, Dilip Soni, Robert L. Nord and Christine Hofmeister, ICSE’95, pp. 196-207