Preface

Somewhere towards the end of the second millennium the director of Vision Consort bv, Hans Brands, came up with the idea to do research in the field of embedded software architectures. He was particularly interested in the process that leads to a successful embedded software architecture. While sharing his ideas with colleagues and other people in the field of embedded software, he became more and more convinced that the result of this research could be useful. The only problem was that he was told that such an investigation would take a lot of man-years. Man-years that were not available to him. Stubborn as he was, he decided to try anyway and have one person do this assignment, this person being a student.

This is where I came in. Being a student of business administration at the Catholic University of Nijmegen I had reached the point where I could start my final thesis. Vision Consort and I found each other and I decided to accept this assignment. This was in June of the year 2000. Now, half a year later, the result is here. I hope a majority of you will find the results useful.

Now, I wish you lots of pleasure while reading this report. In case of questions or remarks please contact me at:

Phesen@visionconsort.nl

Paul Hesen
Eindhoven (The Netherlands), January 2001

Contents

Management summary

INTRODUCTION

Electronic devices have to be put on the market in an ever increasing speed. Before the product can be put on the market it has to be developed first. Before it can be developed an architecture has to be made. The problem is that although this is understood the skills concerning an architecture are still scarce (Muller page 30). At the moment creating an architecture is still an art, no clear definitions of what an architecture should look like exist (Muller, page 30).

At Vision Consort we also saw that this was a problem and we wanted to contribute to the solution of this problem. Therefore we started an investigation. Before starting we first limited the scope to the creation of embedded software architectures. From that scope we came up with the following goal:

"Stating the factors of success at the creation of an embedded software architecture by doing a literature study, interviewing stakeholders from the process and testing the findings in a case study"

Seeing a software architecture like this:

" A software architecture is the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution" (Reifer, page 630).

THE MODEL

The research delivered a large amount of success factors, so many that some structure is needed. This structure is derived from the theory about the fuzzy front end of new product development. Researchers Khurana and Rosenthal divided the elements that play a part in the fuzzy front end into two categories (Khurana and Rosenthal, page 104). The first concerning the activities that cut across projects and form the basis for project specific activities, naming them the foundation elements. The second concerning the individual project, naming them the project specific elements.

Analogically the software creation process can be categorized. The difference is that the foundation elements are split in two. The first set of foundation elements concerns the organization. The second one concerns the competencies of the software architect. The project specific elements are not divided. In figure M 1 the basic structure of the model is shown.

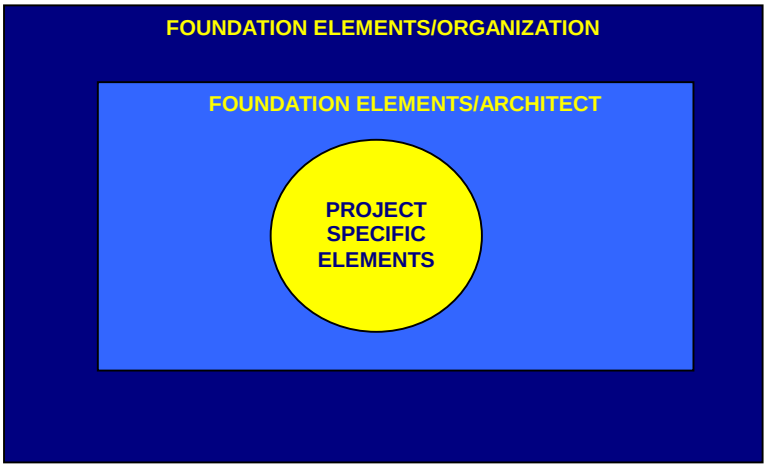

Figure M 1 *Basic structure of the model of the creation of a successful embedded software architecture*

FOUNDATION ELEMENTS/ORGANIZATION

The foundation elements concerning the organization consist of giving management support, using the capability maturity model, using quality assurance, guaranteeing the competencies of the software architect and providing feedback.
Following can be concluded about the foundation elements concerning the organization:

The management of the organization must support the process with resources and political influence. They must make sure that the software architect can do his job to his or her best possible effort. This also means that the competencies of the architect must be guaranteed. Beside the support they must also stimulate feedback given by the organization to the software architecture and to the software architect. Finally, the organization must be characterized as an organization that is mature and makes use of quality assurance.

FOUNDATION ELEMENTS/SOFTWARE ARCHITECT

The software architect has to perform two roles during his work as an software architect. The first role is a conceptual role. He uses this role when he is actually putting the software architecture together. The second role is a more managerial role. He uses this role when he is implementing the software architecture.
Beside the roles there are a lot of competencies the software architect must have. In the following figure (M 2) you can see which competencies and see their importance for the conceptual and the managerial role. The data come from a case study performed during this investigation. In the model 0 stands for not important at all and 10 for essential.

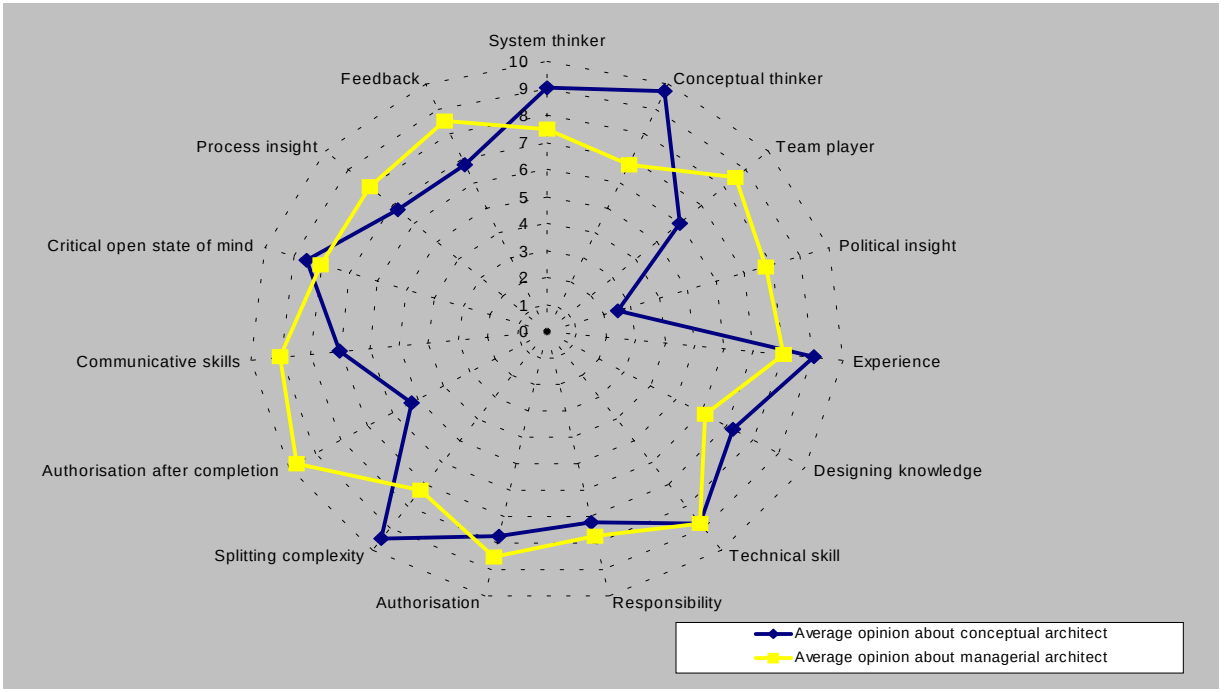

Figure M 2 *Kiviat diagram of the comparison between the conceptual architect and the managerial architect*

Following can be concluded about the foundation elements concerning the software architect:

A software architect performs two roles. In the first role he is a conceptual figure, in the second role he is more of a managerial figure. A lot of competencies can be attributed to the software architect. These competencies can be structured in knowledge based competencies (experience, technical knowledge), human contact based competencies (communicative skills, being a team player) and competencies concerning his place in the organization (status, responsibility).

PROJECT SPECIFIC ELEMENTS

The factors of success concerning the individual project are placed under the project specific elements. They consist of:

- Deriving the software architecture from the business goals;
- Using iteration;
- Having an eye for functional AND non-functional requirements;
- Influence of the software architect during the implementation of the software architecture;
- Using and creating documentation.

Following can be concluded about the project specific elements:

A software architecture must be driven by business goals and must be made in an iterative manner. The requirements in general and the non-functional in particular are very important to the software architecture. The presence of the software architect during the implementation of the software architecture is also crucial and so is the documentation during the whole project. The last two factors of success are both based on a common ground, which is learning from others.

THE FINAL MODEL

The final model combines all factors of success named in the previous three sections. In figure M 3 the final model can be seen.

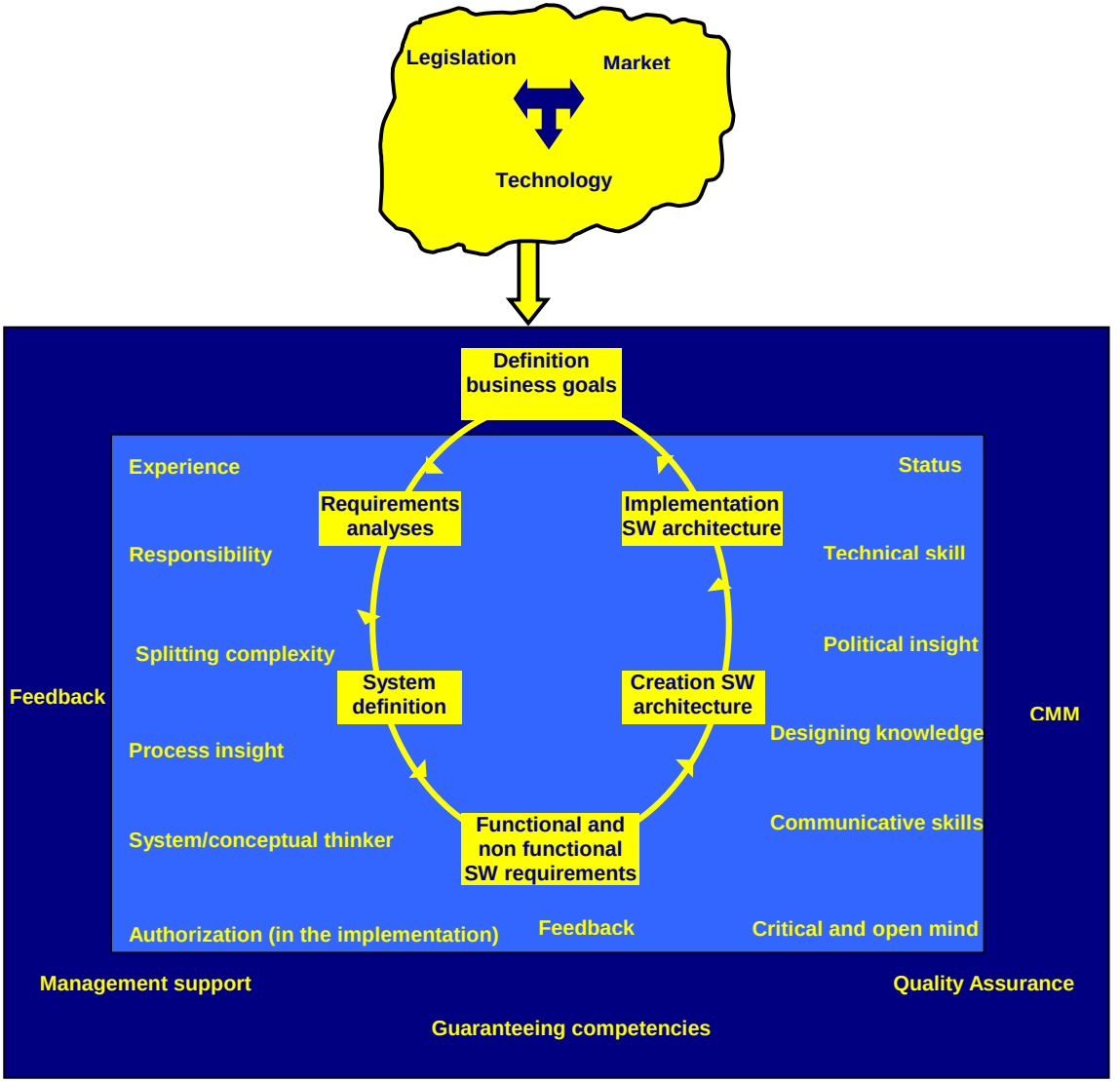

Figure M 3 *The final model*

Chapter one Introduction

1 GOAL

Electronic devices have to be put on the market in an ever increasing speed. Before the product can be put on the market it has to be developed first. Before it can be developed an architecture has to be made. The problem is that although this is understood the skills concerning an architecture are still scarce (Muller 30). At the moment creating an architecture is still an art, no clear definitions of what an architecture should look like exist (Muller 30).

At Vision Consort we also saw that this was a problem and we wanted to contribute to the solution of this problem. Therefore we started an investigation. Before starting we first limited the scope to the creation of embedded software architectures. From that scope we came up with the following goal:

"Stating the factors of success in the creation of an embedded software architecture by doing a literature study, interviewing stakeholders from the process and testing the findings in a case study"

2 THE INVESTIGATION

The investigation consisted of three parts. In the first part literature in the field of innovation management and new product development was studied. The results of this were a couple of topics that were used during the first 8 interviews. The results of the literature and the first interviews were the end of part 1.
In part 2 more interviews took place. The preliminary view from part 1 was discussed in another 4 interviews. These interviews had to bring out the mistakes in the preliminary findings.
Part 3 of the investigation was about seeing if and in what way the topics, which were discovered during the investigation, were used in an existing project.

3 DEFINITIONS

We are talking about architectures or embedded software architectures. But do we really know what an architecture is? An overview:

The IEEE comes up with the following definition of an architecture (IEEE, page 10):

"An architecture is the organizational structure of a system or component"

The IEEE also has a definitions for architectural design (IEEE, page 10)

"Architectural design is the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a (computer) system"

In paragraph 1 we saw that the investigation involves software architectures. Below you will find a definition specifically for software architecture. It is a definition by Donald Reifer (SW management, page 630).

" A software architecture is the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution".

This definition has two novelties in comparison with the other two. The first one is that it recognizes that principles and guidelines are also a part of a software architecture. The second one is that the software architecture is not only made for the design of a product but also for the evolution of that product.

In the investigation we have seen the creation of a software architecture broader than putting the software architecture together on paper. The creation can not be done via a phase model

but instead must be created in an incremental way. Therefore when we say creation we mean putting the software architecture together on paper as well as implementing the software architecture.

4 THE STRUCTURE OF THIS DOCUMENT

In the following chapters a report of the investigation will be given. The report consists of six chapters. In chapter two an introduction of the model will be given. You will notice that the model basically consists of three parts. Also in chapter two the content of the three parts will be discussed in general. Chapters three, four and five will give a more thorough view of each part. At the end of each chapter the findings of that chapter will be placed in the model. The last chapter, number six, will give an overall conclusion. At the end of the report you will find a bibliography.

Chapter two The model

1 INTRODUCTION

In this chapter the basic structure of the model will be discussed. You will see that the model consists of three parts. The elements that are present in each part will also be summarized in this chapter. A more in-depth look will be given in chapter three, four and five.

2 STRUCTURE

The research delivered a large amount of success factors, so many that some structure is needed. This structure is derived from the theory about the fuzzy front end of new product development. Researchers Khurana and Rosenthal divided the elements that play a part in the fuzzy front end into two categories. The first concerning the activities that cut across projects and form the basis for project specific activities, naming them the foundation elements, the second concerning the individual project, naming them the project specific elements (Khurana and Rosenthal, page 104).

Analogically the software creation process can be categorized. The difference is that the foundation elements are split in two. The first set of foundation elements concerns the organization. The second one concerns the competencies of the software architect. The project specific elements are not divided. In figure 2.1 the basic structure of the model is shown.

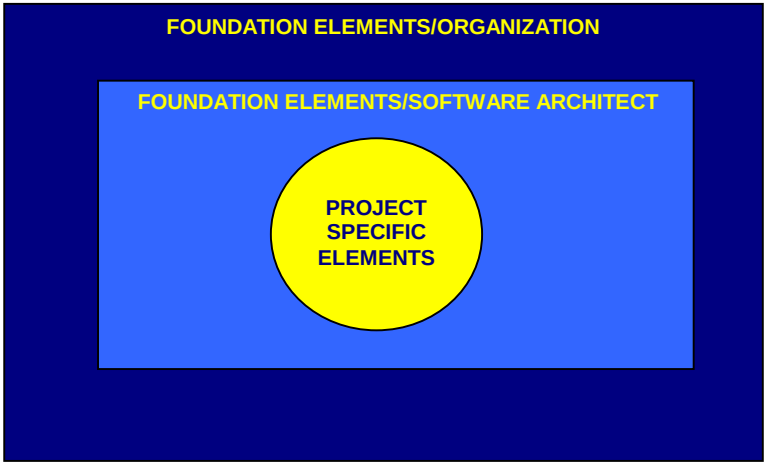

Figure 2.1 Basic structure of the model of the creation of a successful embedded software architecture

2.1 foundation elements/organization

The foundation elements concerning the organization involve the competencies of the organization. These competencies make the creation of an software architecture easier, this because they must lead to a process where good software architects can do the job they are hired for and do this in a way which benefits the organization to the highest possible level. The competencies are:

- Management support;
- The use of CMM;
- The use of quality assurance;
- Guaranteeing the competencies of the software architect;
- Feedback from the organizations towards the software architecture.

2.2 foundation elements/software architect

The foundation elements concerning the software architect involve competencies of the software architect. Some of these competencies are more useful in the creation stage and some are more useful in the implementation stage. The competencies are:

- Being a conceptual thinker;
- Being a system thinker;
- Being a team player;
- Having political insight;
- Being experienced;
- Having knowledge of designing techniques;
- Having technical skills;
- Having responsibility;
- Being authorized;
- Being able to split complexity;
- The extent to which he is authorized in the implementation stage of the software architecture;
- Having communicative skills;
- Having a critical, open state of mind;
- Having process insight;
- Being able to cope with feedback;
- Having a certain status in the organization;

2.3 project specific elements

Project specific elements involve activities that are actually performed during the creation and the implementation of the software architecture. The first rule here is that a software architecture always has to be derived from the business goals. From there all kinds of stakeholders provide functional and non-functional requirements. Finding a balance between often conflicting requirements is a job in itself. After the requirements have been analyzed the software architecture can be made. After it is made, it has to be implemented in the organization. The presence of the software architect in this stage is crucial.

Chapter three Foundation elements/organization

1 INTRODUCTION

In chapter 2 we already saw that the organization plays a part in the chance of success of an embedded software architecture. Five factors were given which can be put under this cause. In this chapter we will take a closer look at management support, the presence of the software capability maturity model (CMM), the presence of quality assurance, the competencies of the software architect and feedback.

2 MANAGEMENT SUPPORT

The fact that management support is needed to create a successful embedded software architecture might not surprise you. But why is it that although we know that it is necessary the support is not always there? Maybe this has to do with the gap that exists between the technical and the managerial world. Let me explain this. In order to have support someone must either be already convinced that something is necessary or he has to become convinced. Simply assuming that management is already convinced is not very sensible. Management often has to be convinced. And when someone has to be convinced they have to be told a story in their own language. In practice this means that the benefits of a software architectural approach have to be translated into business terms.
When that is achieved, the creation and implementation will be easier. This is because of potential additional budget that management can provide, the buffer they can be to critics of the approach and the fact that important software architects can be made available for the project.

3 THE USE OF THE CAPABILITY MATURITY MODEL

In the industrial environment it is very common to use the CMM. Although CMM does not provide many guidelines for software architecture creation it can be a helping factor. This factor will surface when the software architecture is implemented, for a couple of reasons.

The first one is that when an organization is already used to a few practices learned while applying CMM, these practices can also be used for the software architecture.

Also, very simple, when certain issues are solved by the CMM there is potentially more attention for the issues concerning the software architecture. A last factor concerns multi site approaches. When one set of components is developed at one site and the other set at another it is very useful when the two sites have an equal level of CMM. It seems as if the bound with software architecture does not exist here, but that is not true. Part of the reason of making a software architecture is that it must enable separation of concern. There is a point in time when the separated components have to be brought back together. When this is problematic because of a gap between the level of maturity, the software architecture has not reached its goal and is not successful.

4 THE USE OF QUALITY ASSURANCE

Quality assurance is also a helping factor for creating a successful embedded software architecture. Knowing what you are going to do before you do it is in many cases a factor for success. When you create a software architecture there are a few steps you need to follow. Knowing which steps you must take, knowing what the output of each step should be and also checking if they are done, contributes to the chance of success. An important factor is that the steps you must take and the output that has to be delivered is a derivative of the business goals.

5 GUARANTEEING THE COMPETENCIES OF THE SOFTWARE ARCHITECT

Although the outcome of this investigation will help the process, people are still the most important factor. Especially in highly qualified work, like software architectural creation, the competencies of a software architect are crucial. Simply relying on the fact that someone should have the competencies is not enough. In the next chapter you are going to see what the competencies are. You will notice that a lot of them are not purely technical skills. Software architects on the other hand are always technically educated. The other competencies therefore need to be trained.

6 FEEDBACK FROM THE ORGANIZATION TOWARDS THE SOFTWARE ARCHITECTURE

Creating a software architecture is a theoretical process. Theory made for practice. Feedback from reality is therefore crucial. Feedback is used to ensure that the actual direction corresponds with the desired direction (Muller, page 25) Human beings learn from their mistakes, *provided that they are aware of them*. Feedback is the starting point of the learning process, because it provides the detection of mistakes (Muller, page 26).
The higher the uncertainty or the larger the duration of an activity, the more important feedback becomes (Muller, page 26).
There are of course two sides to this story. The first one is that the organization provides feedback. The second one is that the software architect does something with this information. In the next chapter more will be told about the software architect and feedback. In this chapter the organization was the scope. So the people in the organization must come up with feedback. This can be done by implementing certain feedback loops.

7 PRELIMINARY MODEL

We have now spoken about the foundation elements concerning the organization. This means we can complete this part of the model. The model no longer has got an empty outer shell. This is presented in figure 3.1.

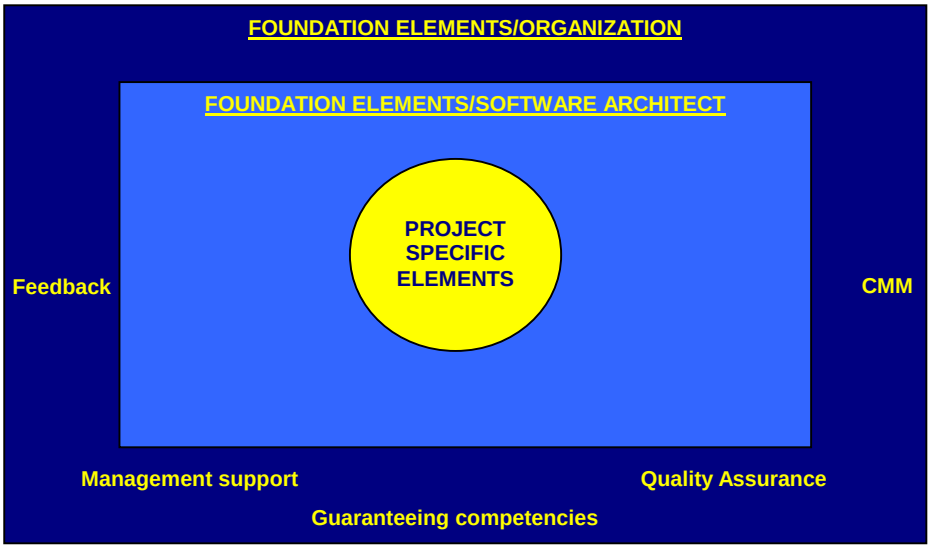

Figure 3.1 *The foundation elements/organization*

Chapter four Foundation elements/software architect

1 INTRODUCTION

In chapter two we saw that the competencies of the software architect form one of two foundation elements. A list of competencies was shown. In this chapter we take a closer look at the competencies. It will become clear that a software architect performs two roles. Some competencies are more important to one role than to the other. In a Kiviat diagram the importance of each competence for a role is pointed out. The data come from the opinion of persons who participated in a case study.

2 THE DIFFERENT ROLES OF A SOFTWARE ARCHITECT

As pointed out before the software architecture creation process consists of the actual creation and the implementation. Although the software architect can not create the software architecture from an ivory tower, there is far more human contact during the implementation phase than during the creation phase. On the other hand creating something requires far more knowledge about the object created than it does to implement it.
Some competencies required are therefore more needed during the creation phase, others are more necessary during the implementation phase. The terminology chosen is the conceptual software architect for the first phase and the managerial software architect for the second.
This does not mean that the role of the software architect must be divided. On the contrary: seeing the role as a whole will be advantageous.

3 THE COMPETENCIES

3.1 being a conceptual and a system thinker

The first competence the software architect must have is being a conceptual and a system thinker. Conceptual because he is working on a concept and not on a physical product. System because what he is going to create is a system on its own but also part of a larger system, that larger system being the product or the roadmap. Thus knowledge of thinking in systems and knowing what consequences of decisions can be to the system is very important.

3.2 being a team player

Because of the size of the projects making a software architecture is not something one person can do. A diversity of people with certain knowledge that complete each other work on the project. The software architect must contribute to this process positively.

3.3 having political insight

During the software architecture creation there are several conflicts of interest between stakeholders. The software architect must go for that interest that benefits the business most. Therefore he must be able to see if someone is talking out of their own interest or out of the interest of the business. He must then be able to take the discussion out of the political light and put it into business terms.

3.4 being experienced

The job of a software architect needs a wide scope. This wide scope can not be learned from books only. Having experienced what the consequences of some decisions are can help the software architect in building a software architecture. Several experienced people all agreed that five years of experience is the lower limit for a software architect.

3.5 having knowledge of designing techniques

This is an item that can be categorized under not re-inventing the wheel. Knowing that there have been studies about how software architectures can be made and knowing what the tenor of those studies is, can help the software architect in building a better software architecture.

3.6 having technical skills

Making an embedded software software architecture is a process that takes place in a technical domain. Knowing and understanding the constraints and the possibilities of this domain is crucial for creating a successful software architecture.

3.7 having responsibility and being authorized

Without responsibility there would be no professional reason for the software architect to make a software architecture. The authorisation can guarantee that what the software architect proposes will be done. If the software architect does not have authority, the software architecture will not have it either. And this breaks down the whole bottom line of the software architecture: being a concept for a product or a part of a product.

3.8 being able to split complexity

The goal of a software architecture is that it must make the problem easier after it is made than before it was made. This means the software architect has to split the total complexity into parts that can be solved autonomically.

3.9 authorisation during the implementation stage of the software architecture

The importance of implementing the software architecture in the development cycle will be discussed in the next chapter. For now it can be said that the software architect must have authority in the development stage. He must explain why certain decisions are made and see if the developers really bring the software architecture into practice.

3.10 having communicative skills

During the process the software architect uses a lot of different communicative actions. He or she must listen to the feedback, convince people with different opinions, disappoint some stakeholders, explain why a decision is made and work together with other individuals. Since all these things are crucial to the process the software architect must have sufficiently mastered them all.

3.11 having a critical, open state of mind

Clear diagrams, tables with facts and smooth presentations give the impression of high quality and increase the confidence. However, these same diagrams, tables and presentations conceal forgotten, misinterpreted, or underestimated facts. The software architect must always be alert.

- Are we addressing the right problem or requirement?
- Is this design adequate?
- Does the input data consist from facts, wishes or ideas?

(Muller, page xv)

3.12 having process insight

During the implementation phase insight into processes becomes important.

3.13 being able to cope with feedback

We saw that feedback was important. Feedback is a form of critic. Not all people can cope with that, but because feedback is important for the final result, being able to cope with feedback is very important.

3.14 having a certain status

Having a certain status can help with the acceptance of ideas. Status can be seen as informal authority. So the story told at 3.7 can also be told here.

4 A KIVIAT DIAGRAM

During the case study 4 players were asked what they thought about the items from paragraph 3. In the next diagram the average scores are presented. The importance of an item for the conceptual role is put off against the importance for the managerial role. In the model 0 stands for not important at all and 10 for essential.

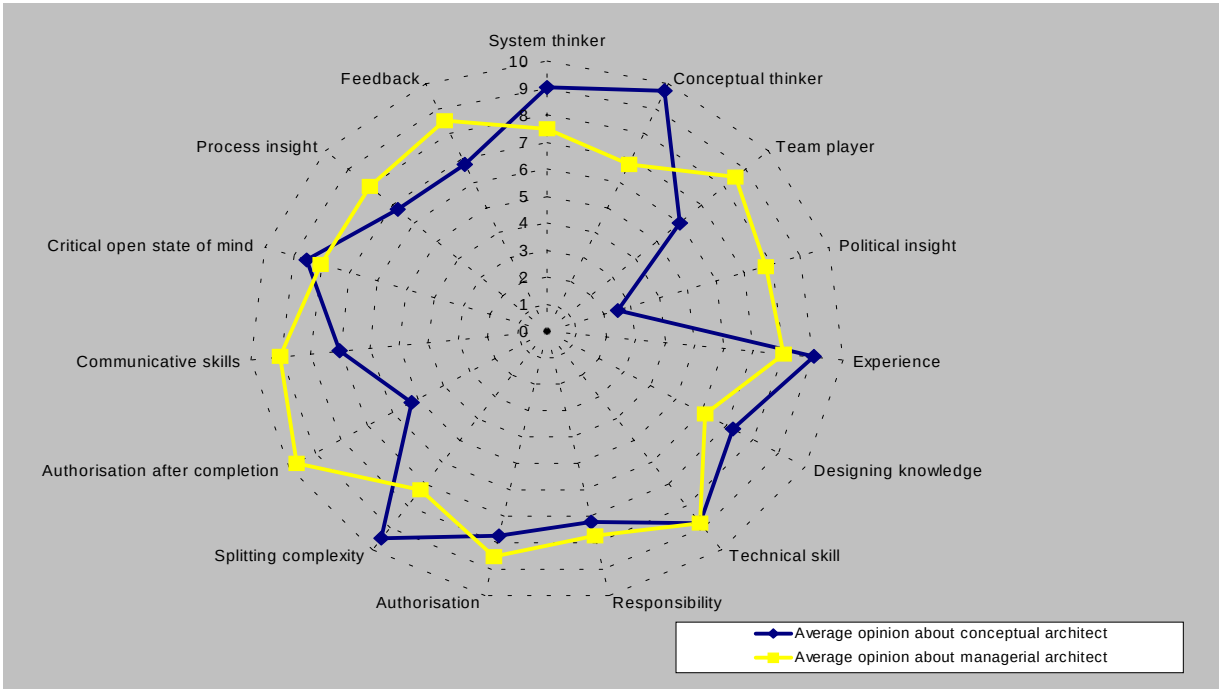

Figure 3.1 *Kiviat diagram of the comparison between the conceptual architect and the managerial architect*

As you can see the item status (3.15) is not present in the diagram. This is because this item was discovered during the case study and could therefore not be measured during the case study.

5 PRELIMINARY MODEL

We have now spoken about the foundation elements concerning the software architect. This means we can complete this part of the model. The model no longer has got an empty middle shell. This is presented in figure 3.2.

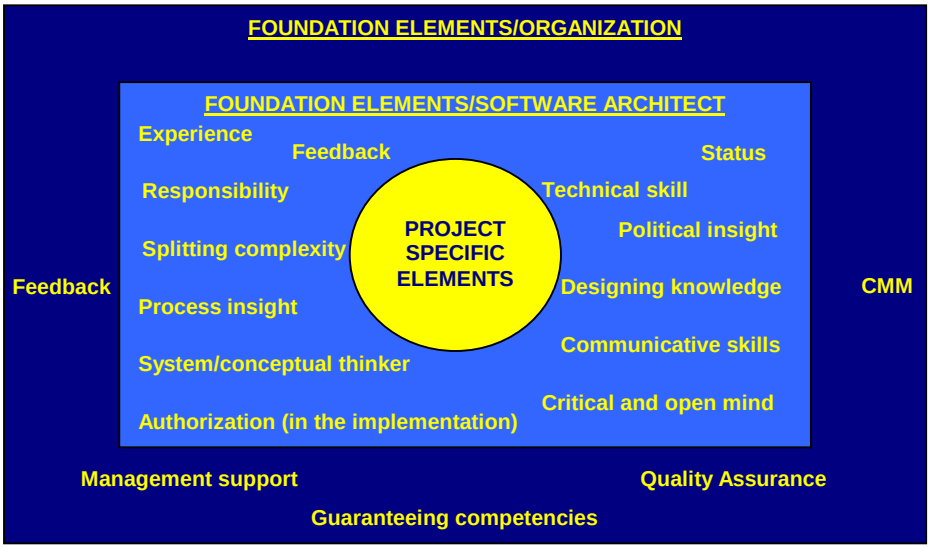

Figure 3.2 The foundation elements/ software architect

Chapter five Project specific elements

1 INTRODUCTION

In this chapter the project specific elements will be discussed. These are the elements that you will come across during the project in which a software architecture is made. At the end of this chapter the total model will be presented.

2 SOFTWARE ARCHITECTURE DRIVEN BY THE BUSINESS GOALS

The first thing that can be said about the project specific elements is that the software architecture should be driven by the business goals. Therefore business goals should be deployed into the organization. The business goals should be the starting point of any software architecture. Without this, building a software architecture is purely a goal in itself.

3 ITERATION

Already made clear at the topic of feedback. The total project of creating a product or a part of a product can no longer be done first time right. Since putting together and implementing a software architecture is also a part of that project, there has to be some sort of iteration involved. How many steps is not clear. What is clear is that in any step the goals that should be reached during that step must be made explicit.

4 REQUIREMENTS

Requirements can be functional and non-functional and can be delivered from outside the organization as well as inside. Maybe the difference between functional and non-functional requirements is not clear.

By functional requirements I mean statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do (Sommerville, page 119).

Non-functional requirements are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, standards an so on. (Sommerville, 119) The non-functional requirements concern e.g. the product or the process and can be driven by external motives (Sommerville, page 131)

The non-functional requirements are playing a role that is becoming more important. Let me explain why. Companies are forced to reuse software components and to develop product families which are based on standard platform software architectures. The lifetime of these generic platforms is much longer than the lifetime of each individual product. This results into very strong demands on non-functional requirements, like the ability of the platform to cope with the fast evolution of standard technologies. (Postema, page 3)

So seeing that the non-functional requirements are important and treating them accordingly becomes an increasingly important factor of success.

5 THE IMPLEMENTATION STAGE

After the software architecture is determined in theory it should be implemented. This is to prevent that the software architecture dies after it is being produced. The software architect should play an active role is this implementation. The first reason is because he has to explain what he meant by the software architecture. A second reason can be that feedback from developers leads to improvements. The third reason is that the software architect now is pushed to step out of his ivory tower. He can see in practice, which the direct implications of his theoretical work are. This way he learns from his mistakes for the simple reason that he can now see what he has done wrong.

6 DOCUMENTATION

In practice it shows that a lot of software architects do not stay in the organization forever. So without documentation the knowledge that the software architect carries leaves the organization together with the software architect. Therefore a thorough documentation of the software architecture and also the process that led to the software architecture is a pro.

7 FINAL MODEL

After the foundation elements concerning the organization and the software architect this chapter gave an inside view of the foundation elements. Therefore it is now time to present the final model. You can see the final model below in figure 4.1.

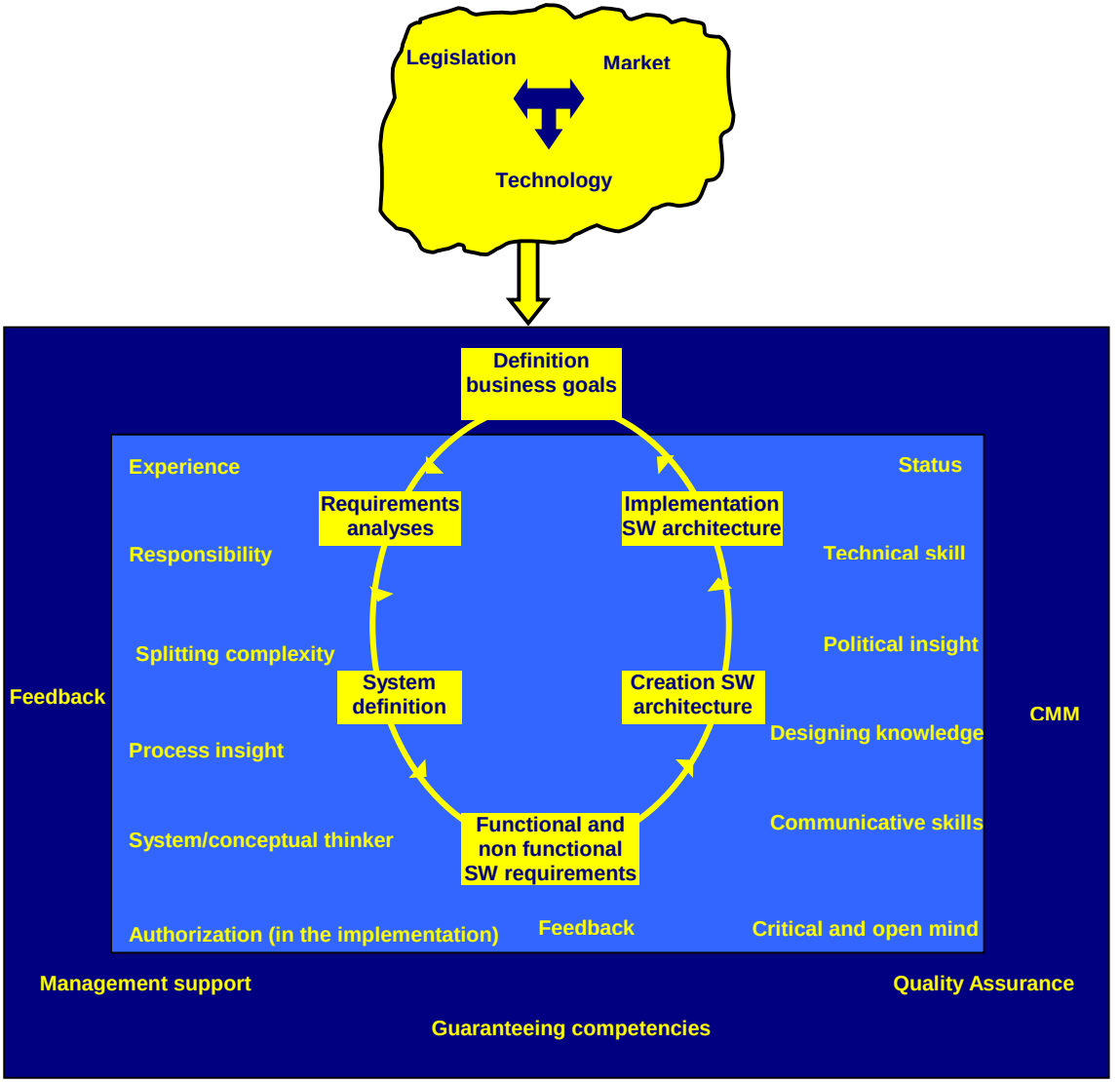

Figure 4.1 The final model

Chapter six Conclusion

The goal of this investigation was to state the factors of success at the creation of an embedded software architecture. I did this through the study of literature, interviewing stakeholders from the process and testing the findings in a case study. Now, at the end of the investigation, it is time to see if I have reached that goal.
To be fair and critical to myself I can state that I did, but I must say that not every factor is elaborated thoroughly. Since this was not the goal of the investigation this is not a real problem for me. However it is a problem for the industry. So I think more research must be done.

Practically I am thinking of research in the field of the competencies of the software architect. Not what these competencies should be, but how we can achieve that the software architects are trained to become the software architects described in chapter four.
Another topic that needs further elaboration is that of the project specific elements as a whole and specifically the topic of iteration and the importance of non-functional requirements. At the moment there are some methods which have a central position for iteration. I have not seen them being used in the interviews or in the case study I did though. Furthermore I also think that the importance of non-functional requirements has grown since roadmaps became more important. Then again, a difference in the approach towards these requirements is hard to see with the naked eye.

Enough about the conclusions about the investigation itself. To you the most important are the conclusions that can be derived from the investigation.

We have seen in this report that the factors of success can be structured in three segments. This is the first conclusion. The segments are the foundation elements concerning the entire organization, the foundation elements concerning the software architect and the project specific elements.

For the foundation elements concerning the entire organization the following can be concluded:

The management of the organization must support the process with resources and political influence. They must make sure that the software architect can to his job to his best possible effort. This also means that the competencies of the architect must be guaranteed. Besides the support they must also stimulate feedback given by the organization to the software architecture and to the software architect. Finally the organization must be characterised as an organization which is mature and makes use of quality assurance.

For the foundation elements concerning the software architect the following can be concluded:

A software architect performs two roles. In the first role he is a conceptual figure, in the second role he is more of a managerial figure. A lot of competencies can be attributed to the software architect. These competencies can be structured in knowledge based competencies (experience, technical knowledge), human contact based competencies (communicative skills, being a team player) and competencies concerning his place in the organization (status, responsibility).

For the project specific elements the following can be concluded:

A software architecture must be driven by business goals and must be made in an iterative manner. The requirements in general and the non-functional requirements in particular are very important to the software architecture. The presence of the software architect during the implementation of the software architecture is also crucial and so is the documentation during the whole project. The last two factors of success are both based on a common ground, which is learning from others.

Bibliography

IEEE, *Software Engineering*, 1994 edition, Institute of Electrical and Electronic Engineers

A. Khurana and S.R. Rosenthal, *Integrating the Fuzzy Front End of New Product Development*, Sloan Management Review, Winter 1997, pp. 103-119

G. Muller, *System Architecture*, Philips Research, www.extra.research.philips.com/natlab/sysarch, version 0.2, May 2000

H. Postema, *Architecture Assesments: Approach and Experiences*, Proceedings SPI'98, the International Congres on Software Process Improvement, Monte Carlo, 1998

D. J. Reifer, *Software Management*, fifth edition, 1997, IEEE computer society

I. Sommerville, *Software Engineering*, fifth edition, 1996, Addison-Wesley