Four Metaphors of Architecture in Software Organizations: Finding out The Meaning of Architecture in Practice

Kari Smolander
Lappeenranta University of Technology, Telecom Business Research Center (TBRC), Department of Information Technology, Laboratory of Information Processing, P.O. Box 20, FIN-53851 Lappeenranta, Finland
kari.smolander@lut.fi

Abstract

Current literature, research, and practice provide ambiguous meanings for the concept of architecture in the context of software and systems development. This qualitative and grounded theory based study delves into the practice of architecture design and description in three software-producing organizations. Nineteen architects, designers, and managers are interviewed and the general meanings of architecture in practical real-life situations are distilled and analyzed. The ambiguity of the concept of architecture receives its explanation. Architecture emerges as a plastic concept including diverging and simultaneous connotations for different stakeholders. The research process produces four general metaphors for architecture, "architecture as blueprint", "architecture as literature", "architecture as language", and "architecture as decision". These metaphors and the research process are presented and discussed in detail.

Keywords: software architecture, metaphor, software organizations, grounded theory, qualitative research

1. Introduction

The concept of software architecture has generated lots of research, writings, and applications during the last decade. However, as Bosch [1] notes, there is a considerable difference between academic perception of software architecture and the industrial practice. The academia has focus on explicit definition of architecture with special purpose description languages and tools, whereas the industry uses mostly informal descriptions and sketches determined by the needs of the situation at hand. In addition, the academia has this far approached architecture mostly and almost solely from the perspective of a technically advanced software engineer. In practice, however, architecture must be communicated and understood by a diverse set of stakeholders with varying skills and experience, including for instance data administration departments and production organizations, different stakeholders at customer side, other suppliers interfacing the systems, hardware vendors, salespersons, and the general management of the company. Without sufficient communication and common understanding about architecture among these stakeholders, a systems development project is probably doomed to fail technically and organizationally, as shown by analogous situations in many studies on systems development failures [e.g. 2, 3].

This research focuses on software and systems architecture design and description processes. The aim is to bring structure to the meaning of architecture in practice and clarify the role of different stakeholders, products, and businesses when creating and describing architecture. Instead of defining technological imperatives in the form of a conceptual theory or a constructive work, we shall observe how architecture is designed and described in industry and try to reach to a comprehensive picture of the meanings of systems and software architecture, including its organizational and inter-organizational connotations. The conceptions of different stakeholders (including architects, software designers, managers, and customers) will be analyzed and the meanings of architecture will be interpreted and summarized in the forms of four metaphors describing the essence of the associated meaning. As the research method, we use grounded theory [4, 5], which suits well for studying new and unexplored areas without well-defined theories.

The paper is organized as follows: first, the research area, hypotheses, and motivation for the study are reflected in more detail in Chapter 2. After that, the research process and grounded theory as a research method will be represented in Chapter 3. Chapter 4 explains the observed meanings of architecture in practice and forms a picture of the phenomenon, introducing four basic metaphors for architecture and its description. In Chapter 5, the implications of the results are discussed.

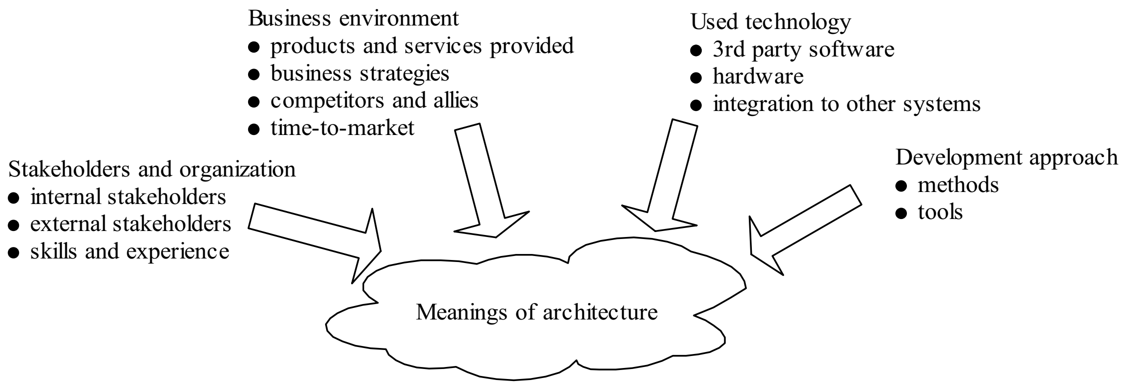


Figure 1. Hypothesis: the creation of the meanings related to architecture

2. Research area and motivation for the study

The software architecture community has been unable to provide a commonly agreed and practically useful definition of software architecture. The definitions are either too general to be useful in practice [as in 6], giving limited guidance to practitioners and researchers, or the definitions are too narrow, emphasizing only some aspects of architecture and leaving others out of the scope. This fact is referred for example in [7] and exemplified in Software Engineering Institute's web page [8], which, at the time of writing, contained 11 textbook definitions, 18 definitions from articles, and 78 "visitor definitions" made by individual readers of the page.

A plausible explanation to the vagueness of architecture as a concept is that in practice, situations vary so much that a general and usable definition is not possible. The variation includes differences, for instance, in technology, products, development methods, business environment, organizational roles, and external stakeholders. Any strict definition of the concept of architecture would probably prove unsatisfactory at some situation. Our hypothesis is that in practice, architecture in the context of software development is more a working concept having many meanings among the stakeholders involved in the process. The creation of the meanings involves the factors presented in Figure 1, including stakeholders and organization, business environment, used technology, and the development approach.

Our aim is to analyze the meanings of architecture in software producing organizations. We wish to formulate a description of architecture as a concept and to give an explanation about the usage of the concept in practice. Most of the available analyses explaining the purposes of architecture are made using practical experiences and insight [6, 9, 10], presented as tentative ad-hoc lists [11], or seen from the perspective of a software engineer only [12]. Our analysis differs from these mentioned so that in addition to the perspective of a software engineer, we see architecture and its description in its organizational and business environment.

The arrows in Figure 1 reflect the possible effect on the set of meanings related to architecture. When architecture is described and communicated inside a group of software engineers only, the set of meanings related to architecture and the rationale for architecture description become different than when a diverse set of customers, subcontractors, hardware vendors, and salespersons take part to the process. Similarly, the meanings change when the business environment changes from product-based to tailored software. In addition, used technology (like a component infrastructure or special hardware) may affect on how the meaning of architecture is interpreted and in many cases, development methods and tools have their own interpretations of architecture. Our hypothesis is that all these factors affect the meanings related to architecture and the emerging rationale for architecture description in the situation. We hypothesize that the question is not only technical and methodical as often expressed within software architecture literature.

The totality forming the meanings in Figure 1 is studied hardly at all in the software engineering community. Many researchers have studied one aspect at a time, like how a certain technology or application domain affects on architecture description [e.g. 13, 14], or how architecture should be described within a certain methodical framework [e.g. 15, 16]. Studies from an organizational or business perspective are harder to find. Some approaches take the existence of different stakeholders and their varying needs implicitly into an account when designing or analyzing architecture [e.g. 17, 18]. These approaches, however, use stakeholders mostly as a source of information and not as an active participant having its own needs and purposes for architecture. Bosch [1] presents a case study on the possibilities for organizing the production of software product lines, but does not reflect on rationale and needs of different stakeholders in needs of different stakeholders in architecture design and description. Though the different concerns of stakeholders are explicitly expressed in IEEE recommended practice for architecture description [19] and such works as [20], these works still have an implicit assumption that there exists a single architecture (with a single meaning) which includes mutually consistent views defined by viewpoints for the use of different stakeholders.

3. Research Process and method

3.1. Studied organizations

The study has been made in three Finnish software-producing organizations that were selected because of the differences in their products, organizations, and business strategies.

The first organization produces software for telecom services. The organization is specialized in producing operator-side services for mobile telecommunications and to some amount also platforms for supporting these services. Its customers come from inside the same corporation. Its products, telecom services, have a very large user base and therefore the users are practically anonymous to the developers. The number of personnel in the software development organization is about 200.

The second organization makes software-based products especially for mobile phones and handhelds. In addition to the product development of its own, it also operates as a subcontractor for telecom operators and electronics manufacturers. Its typical customers include manufacturers of mobile hardware. The size of the personnel in software development is about 200. The organization has recently made a strong strategic commitment to a certain mobile software technology.

The third organization acts as an IT solution provider in several sectors. It has a wide range of customer-oriented services and it works as a systems integrator delivering tailored applications and information systems to the customers, including telecom sector companies and public administration. When delivering systems, it must integrate to customer's existing IT infrastructure and legacy systems, and in many cases even take care of their operations. In this study, we interviewed personnel from two divisions of the organization, having 400 and 600 persons associated with systems and software development.

All the organizations are technologically advanced and, for instance, use extensively UML in their designs and make their implementations with the latest technology, including Java and component architectures. Although all the organizations are somehow dealing with the telecom industry (a service developer of an operator, a software developer for telecom devices, a developer of tailored information systems – also for other industries), we do not see this as a serious source of bias in this study. On the contrary, the significant differences in products, organizational structures, and practices between the organizations facilitate the execution of a comparative analysis.

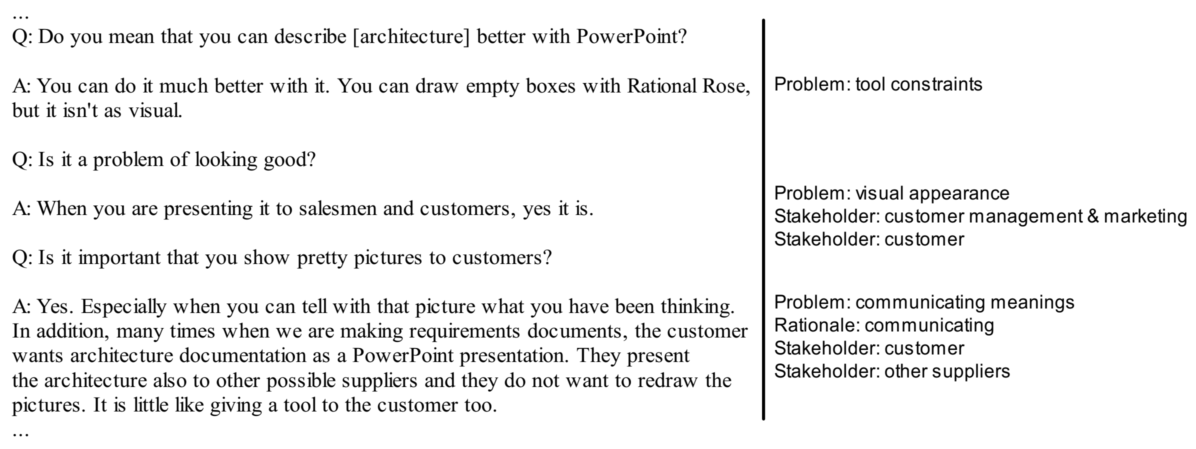
3.2. Pre-study: used and needed architectural viewpoints in practice

The research started as a pre-study with a general investigation on architectural practices and used architectural viewpoints in the three organizations. The current state-of-the-art was discussed with chief architects in these organizations. The status of architectural practices was resolved by using both questionnaires and prepared presentations by the chief architects. The purpose of this phase was to achieve background information and understanding about the situation, and to serve as the basis for interpretations. After the general investigation, the chief architects collected experiences from available projects and identified the explicit and implicit architectural viewpoints [as in 19, see also 21] that were typically used or needed in the architecture development in their organizations. After resolving the lists of viewpoints, the author of this paper participated in analyzing and interpreting the rationale for the viewpoint selection. The pre-study is reported in [22].

The mentioned factors affecting on the selection of architectural viewpoints included both technical aspects (like performance and concurrency requirements) and non-technical aspects (like business and communication needs or the organization of the software production). It seemed that the importance of non-technical aspects was higher than the importance of technical aspects, but we could not make any definite arguments based on the data available at that time. We felt that more research was needed before claiming any further arguments of the factors, their interrelationships, and their relative importance. We realized soon that software architecture as a concept includes more than what is included in the conception of the programming-in-the-large [23]. The pre-study left several questions open, like what is the actual practice in the organizations compared to the conceptions by the chief architects. In addition, if non-technical aspects were more important when selecting the viewpoints, then what was the rationale behind architecture description in practice? For plain programming and technical implementation, non-technical viewpoints are not necessarily needed. We felt that more research was required to reveal the concept of architecture in its full richness.

3.3. Qualitative study on the rationale for architecture description

After the pre-study, it was clear that there will be no known theory or comprehensive framework available in



...	
Q: Do you mean that you can describe [architecture] better with PowerPoint?	
A: You can do it much better with it. You can draw empty boxes with Rational Rose, but it isn't as visual.	Problem: tool constraints
Q: Is it a problem of looking good?	
A: When you are presenting it to salesmen and customers, yes it is.	Problem: visual appearance Stakeholder: customer management & marketing Stakeholder: customer
Q: Is it important that you show pretty pictures to customers?	
A: Yes. Especially when you can tell with that picture what you have been thinking. In addition, many times when we are making requirements documents, the customer wants architecture documentation as a PowerPoint presentation. They present the architecture also to other possible suppliers and they do not want to redraw the pictures. It is little like giving a tool to the customer too.	Problem: communicating meanings Rationale: communicating Stakeholder: customer Stakeholder: other suppliers
...	

Figure 2. An example of open coding

the field of software architecture explaining the whole picture of architecture. We decided to use an exploratory, qualitative, and theory-forming strategy in our research. We use the word exploratory, because we could not fully anticipate where the results will take us; qualitative, because it is not possible to measure quantitatively a concept not defined exactly; and theory-forming, because we wanted to propose an explaining theory to a phenomenon not having such. For these reasons, we found *grounded theory* [4, 5] suitable for our purposes. It is a research method developed originally in social sciences, which uses qualitative content analysis for the construction of a theory grounded in the data about the phenomenon under study.

Grounded theory has proved its usefulness when dealing with new and unexplored areas related to processes and change in organizations. The role and meaning of architecture in systems development projects exemplifies such an area without scientifically established theories and concepts. Because the theory creation in grounded theory is strongly grounded to the data (instead of researcher's divine inspiration), the resulting theory is accurate and the research tends to produce useful and practically valid results [24]. Information systems research offers many examples of the application of grounded theory [for instance, 24, 25], and the field of software engineering also recognizes the need for qualitative approaches in the areas related to human behavior [26].

The study started with interviews in the first organization. First, six theme-based and tape-recorded interviews were made and completely transcribed. The interviewees in the first organization included two designers, two architects, one project manager, and one department manager. These roles were considered the most central in software development based on the knowledge we already had about these organizations. In the first organization after the first interviews, we however decided to extend the interviews to include also one internal customer, which seemed to be a central role in that case. Glaser and Strauss [4] call this dynamic process of data collection where the collection is extended and directed according to the emerging theory as *theoretical sampling*. The study continued with *open coding* [5] of the transcripts from the first organization – the conceptualization and categorization of the data. The transcripts were coded using ATLAS.ti [27], which is a software tool for qualitative analysis. Figure 2 shows an example of open coding of an architect's interview transcript. The coding started with the high-level "seed categories" [28] of stakeholders, problems, and rationale for architecture description (as in Figure 2), but continued with constant creation of new categories and merging of existing categories as new evidence and interpretations emerged. The open coding of all the 19 interviews in the three organizations produced altogether 179 different categories.

When browsing systematically through the data and coding the occurrences of architecture-related phenomena in the first interviews, the researcher was beginning to grasp the conception represented in this paper. The emerging conception or theory was confirmed and further developed by twelve more interviews made at the two other organizations. These included interviews of three more architects, three designers, and six managers, of which two were team leaders, two technology managers, one site manager, and one process development manager. The interviews served as a means to implement *comparative analysis* [4], in which more evidence is collected, and the theory is generated and verified against the additional data. In addition, the objective was to achieve *theoretical saturation* [4], where additional data would bring no new categories or properties to the emerging theory. Again, the interviews were fully transcribed and coded and, concurrently, the coding of the first organization was extended to *axial coding* [5], which included the determination of the causal relationships between the code categories. In axial coding, the properties of the categories and causal relationships between the phenomenon and its context were further elaborated. The analysis included also data from other sources than interviews, such as sample architecture descriptions and the software process descriptions from all the three organizations.

According to Strauss and Corbin [5], the coding phase of the grounded theory process concludes at *selective coding*, in which the core category is selected and related to other categories. The extraction of the four architectural metaphors presented below is considered as an application of selective coding here. The core category here is "the meaning of architecture in practice" and the story tells about the appearance of the meaning in the presence of categories like different stakeholders, systems development practices, and other significant observed features.

4. Meanings of architecture in practice

During the interviews, it became evident that the meanings related to architecture varied according to the position and background of the interviewee. The aspects of architecture emphasized by various stakeholders differed substantially. For instance, the aspects emphasized by the designers were not the same as those emphasized by the managers [29]. The managers saw architecture as a communicative vehicle for establishing common understanding among the stakeholders, whereas for designers (and also for some architects), architecture served as a specification to be implemented as such, i.e. they saw architecture design as a pre-phase for programming. In addition, the business environment, organization of the software production, and especially the role of the customer in the design process had considerable effects on the conceptions of architecture [30]. The more the customer was involved in the process and the more diverse set of stakeholders with different backgrounds and experiences participated the process, the more architecture and its description was related to the creation of common understanding, instead of being an explicit specification for the implementation.

These varying emphases on architecture and its description lead us to scrutinize the meanings related to architecture more closely. Architecture included different connotations that occurred simultaneously for different stakeholders. Architecture emerged as a plastic concept offering means to integrate the work of various stakeholders, including designers, architects, project managers, general managers, customers, marketing people, data administration, and so on. Each stakeholder group used architecture for satisfying its own informational requirements. These requirements were related to buying or selling the system, running the system, understanding its operation, understanding the project scope and estimating its progress, designing the high-level structure of the system, or programming the components of the system, among many other aspects related to the informational requirements of architecture.

As the analysis continued, patterns of more general meanings began to emerge. The analysis concentrated on resolving the purpose of architecture and architecture description for different stakeholders, and through that, distilling generalized meanings for architecture in a metaphorical form. As Lakoff and Johnson [31] note, most of our conceptual system is metaphorically structured. Many of our concepts are either abstract or not clearly delineated in our experience, and therefore "we need to get a grasp on them by means of other concepts that we understand in clearer terms" [31]. In fact, software architecture itself is a metaphor where we understand the structure of a software system in terms of buildings, which we can grasp more easily. Metaphorical analyses have been made in many areas, including language and philosophy of meaning [31, 32], organizations [33], information systems [34, 35], and workflow management [36]. In our analysis, we extracted four general metaphors for architecture and analyzed the circumstances under which the metaphors occurred and the characteristics for architecture and architecture descriptions associated with the metaphors. The four metaphors were named as follows:

- *Architecture as blueprint*: architecture is the structure of the system to be implemented.
- *Architecture as literature*: architecture resides at the documentation and reference architectures for future readers.
- *Architecture as language*: architecture is the language for achieving common conception about the system.
- *Architecture as decision*: architecture is the decision and basis for decisions about the system to be implemented.

The metaphors for architecture were distinguished and characterized using certain emerging properties [5] related to the architecture descriptions and design activities. The properties appeared in the context where the metaphors were emphasized and the combinations of the properties made the distinction between the metaphors possible. The following properties emerged from the data and proved to be essential in characterizing the metaphors:

- *Time orientation*: are architecture descriptions oriented towards the solution made in the past (documentation), towards understanding the present design situation at hand, or towards prescribing the future system to be implemented?
- *Formality of descriptions*: are the descriptions only for enabling understanding or are they eventually meant for generating executables?

- *Detail level*: is architecture described in full technical detail or is too much detail considered even harmful?
- *Purpose of descriptions:* are the descriptions made only to describe a solution already built or do they prescribe an artifact not existent yet?
- *Typical activity*: what are the typical activities associated with architecture descriptions?
- *Objective*: what is the basic objective of architecture design and description?

These properties can be used to characterize the metaphors, but there are also other related properties characterizing the circumstances under which the metaphors arise and are emphasized [30]. These include:

- *Customer orientation*: how much interaction between the development organization and its customers is regularly needed?
- *Business orientation*: how much reasoning the development group must make about the business area of the system?
- *Diversity of stakeholders*: how many roles with different backgrounds, organizations, and professions occur in software development projects?

In the following, the four metaphors are described in detail.

4.1. Architecture as blueprint

"Our development is organized so that we first describe the architecture and from that comes the DLL[1] descriptions and then possibly different persons make the individual DLL's. In a way it [the architecture description] is the basis for the next phase, which is the DLL design."

Jack, Software Engineer

The first metaphor for architecture is given the name "architecture as blueprint" because it emerges strongest among those involved in implementing or programming the system according to specifications. Within this metaphor, architecture description is considered as a high-level implementation of the system. It is directly guiding more detailed implementation work aiming at the production of individual components. Architecture descriptions are used for transferring explicit information from architects to designers and other software engineers. The full architecture then resides in and can be observed from the working implementation of the system. In the current software architecture research, this metaphor can be associated with the architecture description languages [see 37].

Architecture description in "Architecture as blueprint" is oriented towards future. Its purpose is to prescribe the future system. The typical activity associated with this metaphor is implementing and the objective is to build software artifacts for the system. This necessitates both high formality and high detail level in the descriptions.

Of the stakeholders involved, designers and some of the technically oriented architects emphasize "architecture as blueprint". This metaphor also co-occurred with the low customer and business orientation of the interviewees. In addition, "architecture as blueprint" was more widely used conception when the diversity of stakeholders was low compared to the situations with diverse stakeholders and high level of interaction with customers.

[1] DLL = Dynamic-Link Library

4.2. Architecture as literature

"I think the purpose of architecture descriptions is to keep the knowledge of what kind of a system we have. Many times the environments are heterogeneous and their parts are here and there. It is nice to have such a document or a part of a document that you can take a view and see what is the scope here. In addition, those who will possibly make further development or maintenance can learn the system easily with it."

Michael, Software Engineer

The second metaphor is closely related to the documentation of the technical structures and to transferring information over time. In "architecture as literature", architecture description is seen as the documentation for future readers and architecture as the solution or the collection of solutions made in the past. In the current software architecture research, the terms used partly within this metaphor include "reference architectures" [e.g. 38], "product line architectures" [39], and "architectural frameworks" [40].

Architecture descriptions are oriented towards past and aim at documentation transferring explicit knowledge and understanding about technical artifacts made in the past. The typical activities associated with "architecture as literature" includes reading and analyzing the descriptions. Since the objective is not only to construct artifacts but also to transfer explicit knowledge between humans over time, the formality of the description varies, but the detail level of the descriptions may rise in many cases high.

As a meaning, "architecture as literature" is not so specific to any stakeholder group. Designers tended to emphasize documentation, but so did more or less the other stakeholder groups too. "Architecture as literature" relates to the problems of managing and making explicit knowledge and transferring the explicit knowledge over time. The metaphor is also related to the reuse of components and designs, which was a well-recognized problem among the interviewees.

4.3. Architecture as language

"From my point of view the purpose of architecture description is that you know where you are going. Usually our projects use quite new technologies and you need to know at a coarse level what our gurus have designed. Therefore, it should not just be in the heads of programmers and architects. The project manager and the project steering group must also know and follow where you are going. It is sure that eventually there will be problems and you must be able to act on them to avoid failures."

Harry, Project Manager

"The purpose of architecture description is that you must be able to tell to the customer and to the team what is your idea."

John, Architect

The third metaphor, "architecture as language", sees architecture enabling the common understanding about the structure of the system. The main purpose is not the creation of formal artifacts or transferring explicit knowledge over time but understanding the present and being able to communicate meaningfully across stakeholder groups. Architecture description serves as the communication between different stakeholders about high-level structures and solutions.

The objective in "architecture as language" emphasizes the understanding between people at present about the situation at hand. The time orientation focuses both on describing the present situation and on prescribing the alternative future scenarios. The typical activity involved in this metaphor becomes communicating and this leads to the requirement of minimal formality and detail level. Too much formality or too high detail level were even considered harmful when the aim was at understanding between a diverse set of stakeholders with varying backgrounds and experiences.

"Architecture as language" was especially emphasized by those with high customer or business orientation, such as various managers and those having sales duties. On the other hand, some designers were not aware of this purpose – at least explicitly. They saw architecture mostly as "blueprint" and "literature" and were not aware of the informational and communicational requirements of other stakeholders. The more diverse stakeholders, the more emphasis were put on "architecture as language". Especially important this metaphor became, for instance, when the customer participation was intense, marketing was closely involved in the process, or when external data administration departments had strong interests.

4.4. Architecture as decision

"The customer is now wondering about the ambiguity of the situation. One architectural choice looked technically quite good but its price could rise so high that they must think about business premises. If it costs 10 millions then how much it must have usage so that it pays the price back in a reasonable time."

Arthur, Architect

"Many times the project has limitations with money, schedule, and also with objectives and requirements. If the expected lifetime of the system is three years and we have only 2 euros and 50 cents money, then it is useless to pursue highest quality. We must then approach from the implementer's point of view and ensure that the implementation work will be efficient and fast. The maintenance has then no importance. If we have other objectives, like if the expected lifetime is ten years or more, or if the usability must be top class, then I know where to put the stakes and what problems must be solved".

Richard, Architect

The fourth metaphor, "architecture as decision", perceives architecture as the decisions about the structure of the system. These decisions influence on the needed resources when building the system, including the needed work force and their special skills and the amount of money that must be spent on third party licenses. The decisions also deal with the trade-offs and strategies needed when making a resolution between conflicting architectural requirements, like usability, maintainability/understandability, and performance. Architecture description becomes then rational decision-making concerning resources (like people or money) and strategies.

"Architecture as decision" is oriented towards future and its associated purpose is to prescribe the future solution. The objective is to plan resources and strategies through decisions and the typical activities associated with this metaphor are evaluating alternatives, and selecting and making decisions about the technical solutions. As with "architecture as language", too much formality and detail may be considered harmful for the efficiency of the decision-making. However, on some technical trade-off situations high formality and detailness may be needed.

Naturally, various managers and resource planners (like project managers) emphasize the metaphor of "architecture as decision". Architecture description was also observed to form the basis for the division of work between working units. However, slightly surprisingly, architecture

Table 1. Four metaphors summarized

Architectural metaphor	Meaning of architecture	Meaning of architecture description	Stakeholder environment	Examples of related areas
Architecture as blueprint	The working implementation	The high-level implementation of the system	Strongest among designers and architects	Architecture description languages [37], module interconnection languages [41]
Architecture as literature	The solution or the collection of solutions made in the past	The documentation for future readers	Emphasized by designers, important for all	Application frameworks [40], product line architectures [39], architecture documentation [19]
Architecture as language	Common understanding about the structure of the system	The communication between different stakeholders about high-level structures and solutions	The more diverse set of stakeholders, the more emphasis on "architecture as language"	Computer supported cooperative work [42], knowledge management [43]
Architecture as decision	The basis for rational decision-making concerning resources and strategies	The decision about the structure of the system	Emphasized by managers	Trade-off analysis [18], software engineering economics [44], IT strategy management [45]

was seldom mentioned as a tool dealing with technological risks, including risks associated with immature technologies or the risk that a technology becomes obsolete.

4.5. Summary of the four metaphors

The purpose of the four metaphors for architecture is not to provide a complete and mutually exclusive classification of different conceptions. Instead, the metaphors show how architecture is understood in practice in terms of other concepts. When describing architecture, we are therefore making blueprints for implementation, documenting the system for future developers, communicating with others stakeholders, and recording decisions. Architecture itself is a metaphor, which we can "deconstruct" into the metaphors of blueprint, literature, language, and decision. The metaphors have overlaps and most of the stakeholders use all of them, but the emphasis varies from stakeholder to stakeholder. Especially the role of an architect in most software producing organizations necessitates simultaneous and balanced usage of all these metaphors.

Table 1 summarizes the four metaphors for architecture. The second column, 'Meaning of architecture', condenses the typical conception of what is meant by architecture within the metaphor. The typical conception about what is the role of architecture descriptions is distilled in 'Meaning of architecture description'. The column 'Stakeholder environment' shows how the metaphor is emphasized among stakeholders. As expressed in the table, the emphasis varies according to the metaphor. The emphasis is quite obvious and evidenced by common sense - still it highlights the plasticity of architecture as an organizational concept. The last column shows a sample of areas of research and practice associated closely to the metaphors. As presented in the column, architecture can be approached from a multitude of viewpoints and many research schools could give contribution to the field.

5. DISCUSSION

5.1. Implications

The direct and obvious implication from the four metaphors is that there are many and even mutually conflicting perspectives of architecture to be described. Some designs and descriptions aim directly at implementation and some only at creating understanding and enabling decisions between the stakeholders. Those architecture descriptions aiming at understanding and enabling decisions may have strong constraints related to politics, social settings, or skills and experience, requiring simplifications, shortcuts, image building, and in some cases even selling glorifications. Therefore, a conflict between architecture descriptions of different purposes is evident and built-in to the concept of architecture. This fact should be recognized in any process and method development occurring in real-life organizations. Unambiguous UML models within the 4+1 framework [21] may work well among software architects and designers, but other stakeholders need also softer and more communicative methods and description languages.

Most of the current research seems to be concentrated on "architecture as blueprint" and "architecture as literature" and some on "architecture as decision". Interestingly, it is hard to find architecture research directly related to “architecture as language”. This calls for intensified research on software organizations about their ability to create, communicate, and transfer architectural knowledge between different stakeholder groups. In addition, research on groupware and communicative tools for architecture description could enable software-producing organizations better meet their architectural goals.

One way to approach architecture as a vehicle for achieving common understanding is to interpret it as a *boundary object* [46-48] to which several stakeholder groups associate their particular meanings. Boundary objects are shared among several communities of practice and they satisfy the informational requirements of each of them. Architecture is shared by most of the stakeholder groups of systems development, who satisfy their informational requirements concerning implementation, maintenance, and decision-making, among many other tasks. Architecture is a plastic concept whose meaning changes according to the stakeholder, situation, and the phase of the project. Even though architecture is plastic, it is robust enough to maintain its identity among the stakeholders. The structuredness of architecture becomes stronger from the weakly structured informal descriptions in the beginning of a project, which are aimed at a common use among diverse stakeholders to enforce understanding, objective setting, and decision-making. When the focus shifts to implementation, the descriptions become more formal and detailed when they are accessible only to software engineers.

5.2. Grounded theory as a research method in software engineering

The field of software engineering contains many organization- and process-related assumptions and argumentation that are hard to study and prove empirically. Grounded theory could give enlightenment to many questions related to activities in software production, software-producing organizations, and software engineer’s workplace. As an example from the field of systems and software architecture, these questions could include for instance, how architecture is created in large information systems projects. Is it purposely designed by architects in a pre-planned manner or does it emerge, for example, from the constraints stated by other legacy systems, conflicting requirements, relationships of political power between the stakeholders, and used technologies. The usage of grounded theory could also shed light on the question of the relationship between requirements and architecture. How much an implicit architecture is involved in determining the requirements and what happens in the interplay between analysts and architects during requirements analysis? We could list many other research problems in the field of software engineering where methods based on testing and experimentation would not work, but where grounded theory or any flavor of practice-oriented research such as action research [49, 50] would be more helpful.

6. Conclusions

This paper has described the process of grounded theory applied to a software engineering subject. The objective was to achieve deep understanding of the meaning of architecture in the context of systems development. The research process produced four metaphors for architecture. The observed four metaphors enable researchers to locate and explicate their interests to architecture with more awareness of intersecting research viewpoints. A conflict between architecture descriptions of different purposes is evident and built-in to the concept of architecture. Unambiguous formal models may work well among software architects and designers but other stakeholders need softer and more communicative methods and description languages to enable understanding and decision-making.

Intensified research on software organizations about their ability to create, communicate, and transfer architectural knowledge is needed. Grounded theory has proved its usefulness when dealing with new and unexplored areas related to processes and change in organizations. The usage of grounded theory could give enlightenment to many questions related to organizational activities in software production, including the communication, constraints, and knowledge transfer involved in the creation of large-scale information systems architecture.

ACKNOWLEDGEMENTS

The author wishes to thank Sandeep Purao (Georgia State University) for helping to rename the four metaphors after the first round of this paper. Many thanks also to Tero Päivärinta (Agder University College, Kristiansand), Matti Rossi (Helsinki School of Economics), and Anne Eerola (University of Kuopio) for their facilitative comments.

REFERENCES

[1] J. Bosch, Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach: Addison-Wesley, 2000.

[2] C. Sauer, G. Southon, and C. N. G. Dampney, "Fit, Failure, and The House of Horrors: toward A Configurational Theory of IS Project Failure," Proceedings of the eighteenth international conference on Information systems, Atlanta, Georgia, USA, 1997, pp. 349-366.

[3] H. Drummond, "Riding A Tiger: Some Lessons of Taurus," Management Decision, vol. 36, no. 3, 1998, pp. 141-146.

[4] B. Glaser and A. L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research. Chigago: Aldine, 1967.

[5] A. L. Strauss and J. Corbin, Basics of Qualitative Research: Grounded Theory Procedures and Applications. Newbury Park, CA: Sage Publications, 1990.

[6] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice: Addison-Wesley, 1998.

[7] J. Baragry and K. Reed, "Why We Need A Different View of Software Architecture," Proceedings of the Working IFIP/IEEE Conference on Software Architecture (WICSA 2001), Amsterdam, The Netherlands, 2001, pp. 125-134.

[8] Software Engineering Institute (2001). "How Do You Define Software Architecture". http://www.sei.cmu.edu/architecture/definitions.html, accessed 28 Mar 2002.

[9] M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Discipline: Prentice Hall, 1996.

[10] P. Kruchten, "The Software Architect - and the Software Architecture Team," Proceedings of WICSA1: 1st Working IFIP Conference on Software Architecture, San Antonio, TX, USA, 22-24 Feb 1999, 1999, pp. 565-583.

[11] D. Garlan, "Software Architecture: a Roadmap," in The Future of Software Engineering, A. Finkelstein, Ed.: ACM Press, 2000.

[12] C. Hofmeister, R. Nord, and D. Soni, Applied Software Architecture. Reading, MA: Addison-Wesley, 1999.

[13] R. N. Taylor, N. Medvidovic, K. M. Anderson, E. J. Whitehead, J. E. Robbins, K. A. Nies, P. Oreizy, and D. L. Dubrow, "A Component- and Message-Based Architectural Style for GUI Software," IEEE Transactions on Software Engineering, vol. 22, no. 6, 1996, pp. 390-406.

[14] G. Fregonese, A. Zorer, and G. Cortese, "Architectural Framework Modeling in Telecommunication Domain," Proceedings of The International Conference on Software Engineering (ICSE'99), Los Angeles, 1999, pp. 526-534.

[15] D. Garlan and A. J. Kompanek, "Reconciling the needs of architectural description with object-modeling notations," Proceedings of the Third International Conference on the Unified Modeling Language - UML 2000, York, UK, 2000.

[16] C. Hofmeister, R. Nord, and D. Soni, "Describing Software Architecture with UML," Proceedings of the First Working IFIP Conference on Software Architecture, 1999.

[17] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-Based Analysis of Software Architecture," IEEE Software, vol. 13, no. 6, 1996, pp. 47-55.

[18] R. Kazman, M. Barbacci, M. Klein, S. J. Carrière, and S. G. Woods, "Experience with Performing Architecture Tradeoff Analysis," Proceedings of the 1999 International Conference on Software Engineering, 1999, pp. 54-63.

[19] IEEE, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE, IEEE Std 1471-2000, 2000.

[20] R. Sanlaville, J.-M. Favre, and Y. Ledry, "Helping Various Stakeholders to Understand a Very Large Component-Based Software," Proceedings of the 27th Euromicro Conference, Warsaw, Poland, 4-6 Sept. 2001, 2001, pp. 104-111.

[21] P. B. Kruchten, "The 4+1 View Model of Architecture," IEEE Software, vol. 12, no. 6, 1995, pp. 42-50.

[22] K. Smolander, K. Hoikka, J. Isokallio, M. Kataikko, and T. Mäkelä, "What is Included in Software Architecture? A Case Study in Three Software Organizations," Proceedings of 9th annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS), 8-11 April 2002, Lund, Sweden, 2002, pp. 131-138.

[23] F. DeRemer and H. H. Kron, "Programming-in-the-Large Versus Programming-in-the-Small," IEEE Transactions on Software Engineering, vol. SE-2, no. 2, 1976, pp. 80-86.

[24] W. J. Orlikowski, "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," MIS Quarterly, vol. 17, no. 3, 1993.

[25] L. J. Calloway and G. Ariav, "Developing and Using a Qualitative Method to Study Relationships among Designers and Tools," in Information Systems Research: Contemporary Approaches and Emergent Traditions, H. E. Nissen, H. K. Klein, and R. Hirschheim, Eds.: North-Holland, 1991, pp. 175-193.

[26] C. B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," IEEE Transactions on Software Engineering, vol. 25, no. 4, 1999, pp. 557-572.

[27] Scientific Software (2001). "ATLAS.ti - The Knowledge Workbench". http://www.atlasti.de/, accessed 28 Mar 2002.

[28] M. B. Miles and A. M. Huberman, Qualitative Data Analysis: A Sourcebook of New Methods. Beverly Hills: Sage, 1984.

[29] K. Smolander and T. Päivärinta, "Describing and Communicating Software Architecture in Practice: Observations on Stakeholders and Rationale," Proceedings of CAiSE'02 - The Fourteenth International Conference on Advanced Information Systems Engineering, Toronto, Canada, May 27 - 31, 2002,, 2002, pp. 117-133.

[30] K. Smolander and T. Päivärinta, "Practical Rationale for Describing Software Architecture: Beyond Programming-in-The-Large," WICSA3 - The Working IEEE/IFIP Conference on Software Architecture 2002, Montreal, August 25-31, 2002.

[31] G. Lakoff and M. Johnson, Metaphors We Live by. Chigago: The University of Chigago Press, 1980.

[32] G. Lakoff, Women, Fire, and Dangerous Things. Chigago: The University of Chigago Press, 1987.

[33] G. Morgan, Images of Organization. Beverly Hills: Sage, 1986.

[34] J. E. Kendall and K. E. Kendall, "Metaphors and Methodologies: Living beyond the Systems Machine," MIS Quarterly, vol. 17, no. 2, 1993, pp. 149-171.

[35] R. B. Gallupe, "Images of Information Systems in The Early 21st Century," Communications of the AIS, vol. 3, no. 1es, 2000, pp. 1-16.

[36] S. Carlsen and R. Gjersvik, "Organizational Metaphors as Lenses for Analyzing Workflow Technology," Proceedings of The International ACM SIGGROUP Conference on Supporting Group Work: The Integration Challenge, Phoenix, Arizona, USA, 1997, pp. 261-270.

[37] N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," IEEE Transactions on Software Engineering, vol. 26, no. 1, 2000, pp. 70-93.

[38] D. Batory, L. Coglianese, M. Goodwin, and S. Shafer, "Creating Reference Architectures: An Example from Avionics," Proceedings of the Symposium on Software Reusability, Seattle, Washington, April 1995, 1995, pp. 27- 37.

[39] P. Clements and L. M. Northrop, A Framework for Software Product Line Practice - Version 2.0. Pittsburgh: Software Engineering Institute, 1999.

[40] E. M. Fayad and E. R. Johnson, Domain-Specific Application Frameworks: John Wiley & Sons, 2000.

[41] R. Prieto-Diaz and J. Neighbors, "Module Interconnection Languages," Journal of Systems and Software, vol. 6, no. 4, 1986, pp. 307-334.

[42] J. Grudin, "Groupware and Social Dynamics: Eight Challenges for Developers," Communications of the ACM, vol. 37, no. 1, 1994, pp. 92-105.

[43] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation," Organization Science, vol. 5, no. 1, 1994.

[44] B. W. Boehm, Software Engineering Economics. Englewood Cliffs, NJ: Prentice Hall, 1981.

[45] J. C. Henderson and N. Venkatraman, "Strategic Alignment: Leveraging Information Technology for Transforming Organizations," IBM Systems Journal, vol. 32, no. 1, 1993, pp. 4-16.

[46] S. L. Star and J. R. Griesemer, "Institutional Ecology, "Translations" and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39," Social Studies of Science, vol. 19, no., 1989, pp. 387-420.

[47] S. L. Star, "The Structure of Ill-Structured Solutions: Heterogeneous Problem-Solving, Boundary Objects and Distributed Artificial Intelligence," in Distributed Artificial Intelligence 2, M. Huhns and L. Gasser, Eds. Menlo Park, CA: Morgan Kauffmann, 1989, pp. 37-54.

[48] G. C. Bowker and S. L. Star, Sorting Things Out: Classification and Its Consequences. Cambridge, MA: MIT Press, 1999.

[49] D. Avison, F. Lau, M. D. Myers, and P. A. Nielsen, "Action Research," Communications of the ACM, vol. 42, no. 1, 1999, pp. 94-97.

[50] R. Baskerville and J. Pries-Heje, "Grounded Action Research: A Method for Understanding IT in Practice," Accounting, Management and Information Technologies, no. 9, 1999, pp. 1-23.