Journal of Computer Science 6 (12): 1424-1429, 2010
ISSN 1549-3636

Achieving Goals through Architectural Design Decisions

Lena Khaled
Department of Software Engineering,
Faculty of Sciences and Information Technology,
Zarqa Private University, Amman, Jordon

Abstract: problem statement: the main problem in building any system is that many decisions appear through its design. These decisions are affected mainly by the goals that the architect wants to achieve. These goals shape the architectural design of a system; the architect needs to know the best decisions to use them through building the design of a system. **Approach:** Design fragments used to solve the problem, design decisions controlled by fragments. Fragments themselves need to be controlled to mange the quality that results from them so quality management activities deal in controlling the fragments. **Results:** Using design fragments helped the architect to choose the most important design decisions to achieve high quality. **Conclusion:** Goals are affected mainly with quality attributes. Choosing the right decisions made building the good quality software.

Key words: Quality attributes, architectural design decisions, software architecture, architectural significant requirement, architectural knowledge

INTRODUCTION

Architectural design (also called software architecture) plays a very important role in a software life cycle. It represents a bridge between requirement and implementation. By defining the abstraction of the system, architectural design describes certain properties of the system while hiding other properties; this representation provides the guidelines for building the overall system, permits the designers to satisfy the requirement of the customer and suggests a plan for the software construction. We conclude that architectural design is very important to the life of software for several reasons. First, it communicates between all stakeholders which are interested in the development of software. Second, it highlights the early design decisions that are found on software engineering work (Garland, 2000). So Architectural Design Decisions (ADDs) are the results of a design process during the initial construction of software (Jansen and Bosch, 2005). This leads to make it the main part that directly influences the construction of the final architectural design of the software.

Every software built upon decisions needs to measure the specific qualities to achieve a specific goal, so we need metrics to determine whether the software has achieved the goal or not.

This study presents how architectural design decisions affect on achieving goal that software is built upon. This is done through defining a design fragment concept and the role of quality control on these fragments.

Related works: Many researchers work on architectural design and its relation with achieving goal. (Liu and Yu, 2001) work on the early stage of architectural design. They explored that goal oriented and scenario based models are combined together during architectural design. They proposed that designers should have notations to help visualize the incremental refinement of an architecture, such notations are used to represent scenario oriented architecture UCM which is an abbreviation to Use Case Map.

Perry and Wolfs (1992) built the foundation for the software architecture. They first developed a perception for software architecture and on the basis of this development they presented a model of software architecture which consists of three components: elements, forms and rationale then they discussed these components on architectural styles.

Another study is (Bachman *et al*., 2003), in that study, the researchers worked on quality requirements and architectural design decisions. They proposed that quality attribute models are linked between a specification of a quality attribute requirement and a design fragments which is focused on achieving their requirement. Each quality attribute has a collection of parameters to determine whether the requirements are met or not. These parameters can bind values of quality requirement, through design decisions, here the

researchers presented a series of steps that enable moving from a single quality attribute requirement to design fragment focused on achieving that requirement. These steps are demonstrated through an application of embedded system.

Ahmad *et al*. (2009) focused on finding intrusion characteristic for IDS using decision tree machine learning of data mining, their conclusion is by combination of IDS and firewall they can detect intrusion and prevent it. Ab-Rahman *et al*. (2009) solving a problem through using network, failure can cause a specific problem. To ensure reliability of network a specific architectural design is built. Khaled (2010) describes the role of agent through building the architecture of any design to achieve the high quality. Omar and Ajitha (2008) made a comparison study between two approaches to make a right decision at a specific time.

This study works on defining design fragments and its relation to architectural design decisions and how this fragment has worked to achieve the goal. It shows also how quality management controls the building of design fragments.

Goals and qualities: Understanding goals and their relations to qualities is an important part of building the architectural design of any system; we cannot easily build an architectural design for any system or even specify the architectural design decisions to it without understanding the concepts of both goals and qualities. Therefore, quality attributes and goals drive the architectural design of the system (Bass *et al*., 2009).

Achieving high quality attributes through architectural design needs an early method used to generate and refine qualities, which is called Quality Attribute Workshop (QAW). QAW is a method that connects system stakeholders early in life cycle of the software to discover the driving quality attributes of the software and clarify system requirements before the software architecture has been created (Barbacci *et al*., 2002). This gets qualities that are mapped to business goals scenarios for the qualities which are built by stakeholders according to the main goals. All these scenarios specify whether a system satisfies the user's requirements or not (Bass *et al*., 2009; Barbacci *et al*., 2002). Quality attributes must be well understood and expressed early in the development of a system's life cycle, so the architect can design an architecture that will satisfy these qualities.

Quality attributes: In manufacturing, the concept of the quality is that the developed product should meet its specification, but the popular vision of the quality is that it is an intangible attribute. Terms of bad or good quality represent how people talk about something vague which they don't propose to define. Quality attributes describe the property of the system that refers to its fitness for use. The term, non-functional requirement, is a synonym for quality attributes (Somerville, 2006; Kan, 2002).

A Quality Attribute Requirement (QAR) is specified to show the characteristics of the system that indicate its fitness for use. An Architectural Significant Requirement (ASR) is any requirement that influence the choice of architectural decisions; it is sometimes called the architectural drivers. Most architectural drivers tend to be quality attribute requirements (Berenbach *et al*., 2009).

The international standard on software product qualities classifies software quality as six main attributes: functionality, reliability, usability, efficiency, maintainability and portability. Despite the fact that there are many quality attributes, reliability and maintainability are the main quality criterions and many of these attributes are created at business levels and are better viewed as business goals (Gross and Yu, 2001; Jalote, 2008). Figure 1 illustrates the relation between goal and quality attributes.

Goals: According to (Liu and Yu, 2001), we can define a goal as a state of events in the world that users would like to achieve and it will be either a business goal or a system goal.

Business goals are the parts that drive the methods of the design and are the elements that shape the architecture. They are about a business or state of business and they contact the individuals or organizations wishing to achieve the goal.

The important thing is that all business goals that correspond to quality attributes will view and measure the end of the system (Barbacci *et al*., 2002; Gross and Yu, 2001).

System goals are about what the target system should achieve, which generally, describe the functional requirements of the target system (Liu and Yu, 2001).

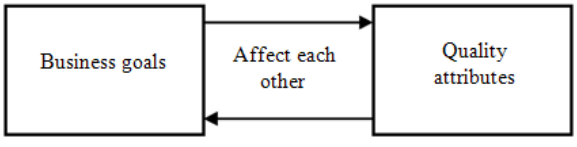

Fig. 1: The relation between goals and attributes

Architectural design decisions: Architectural design decisions can be defined as descriptions of the choices and considered alternatives which are described as an addition, subtraction and modification to the software architecture, the rationale, design rules, design constraints and additional requirement that realize one or more requirements on a given architecture (Johannes, 2008).

With the definition of architectural design decisions we use the following important elements:

- With alternatives: we mean other solutions to the requirement. The choice is the decision part which leads to the architectural design decisions
- With addition, subtraction and modification: they are all changes that are made to the software architecture by architectural design decisions
- Rationale is a brief description of each ADD written behind the decisions
- Rules and constraints are considerations for further decisions

We conclude that architectural design decisions are decisions that directly influence the design of software architecture (Johannes, 2008).

Figure 2 represents the distinctions between architectural decisions and design decisions. The similarities and differences between each concept are:

- Design decisions are decisions that directly influence the design of the system
- Architectural decisions are decisions that directly influence the software architecture like decisions that address Architectural Significant Requirement (ASR). Architectural decision that are not Architectural design decisions are those decisions that affect the software architecture indirectly
- Architectural design decisions are decisions that directly influence the design of the software architecture. For example, choosing the architectural style for a design is an architectural design decisions

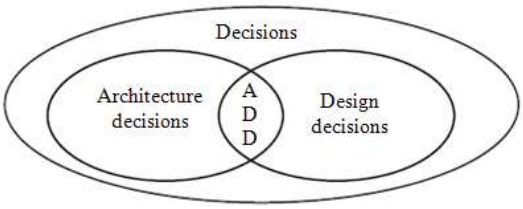

Fig. 2: Similarities and differences between decisions in a Venn diagram

Achieving architectural design decisions: Previously we explained how software architecture is based on the requirements for the system.

Many software architecture design methods present and they all use different methodologies for designing the software architecture.

Figure 3 represents how the architectural decisions are made through the process of the software architectural design. It shows that making decisions is a cyclic process; this means that those decisions are achieved through reputations till achieving right decisions.

Figure 3 shows ADD as a result of the design process during the initial construction of the design. It shows that the main input of the design process is the requirement of the software so the initial design of the software architecture is built in order to satisfy the requirements of the system. If the quality output of the software architecture is not sufficient then the architectural design decision is modified to build the architecture according to the user's requirements. This is done through a number of tactics by adapting one or more architectural styles or patterns to improve the design.

Software architecture and ADD: Software architecture is the structure of the system which includes elements, the visible parts of these elements and the relationship between these elements. It is one of the main disciplines of software engineering which study the high level abstract view of the system.

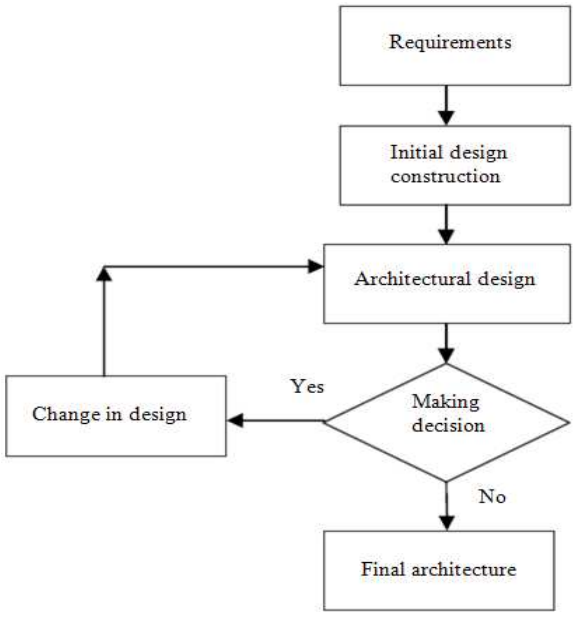

Fig. 3: Software design process

Software architecture is created, maintained and evolved in a very complex environment (Johannes, 2008).

Figure 4 illustrates that software architecture depends on the requirements that explain what the system should do while software architecture describes how this is reached. All information and knowledge about the decisions on architecture are totally set in into software architecture, so all the knowledge about the design disappears into software architecture. This causes some problems like:

- Obsolete design decisions are not removed and this allows unexpected behaviors to happen
- Design decisions affect multiple parts of the design so that associated knowledge is distributed across different parts of the design making it hard to find and change

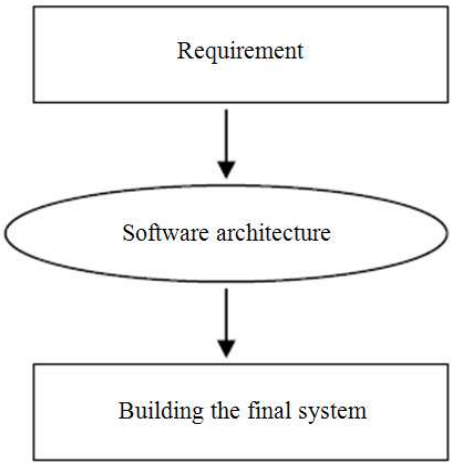

Fig. 4: The road map of software architecture

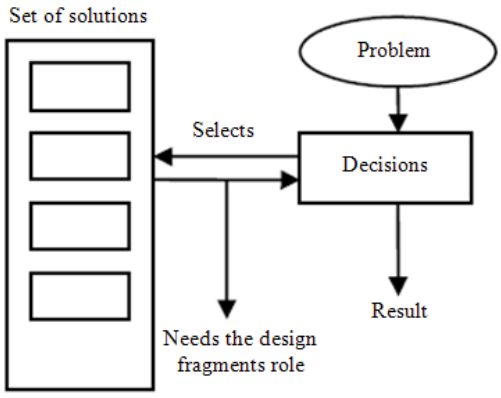

Fig. 5: The role of a design fragment on a design decisions

The notion of Architectural Knowledge (AK) includes the knowledge involved with software architecture; it improves the quality of the architecture and the process that made it. Some researchers define AK as the following formula (Johannes, 2008):

AK= design decisions + design

The lost architecture knowledge leads to evolution problems, blocks the reuse of the system and increases the complexity of the system.

Design fragments: A design fragment is an architectural fragment defining asset of architectural entities. An architectural entity can be part of multiple design fragments. The primary use for design fragment is to define the scale of a solution of design decisions (Fairbanks, 2007). Figure 5 represents the role of a design fragment on a design decisions. Each decision has: main concept, problem and set of solutions. Each solution has its own rationale and the realization part (which is meant a design fragment).

A design fragment on the other hand needs a real control to make the necessary decisions like schedules. At the same time it needs a sufficient information or data to enforce these decisions. Sometimes a design fragment composition concept arises; this is done when a design fragment needs to change another design fragment.

MATERIALS AND METHODS

Quality activities are the main methods that are used to achieve a high quality of a system. Three main activities are defined through management: quality planning, quality control and quality assurance.

To achieve high quality, the project must be planned in advance by deciding which quality factors are important for the project and select standards and procedures from the quality manual that are appropriate to meet the quality goals of the system. The resulted goal must be measured by metrics; GQM can be used as one type of metric which focuses on goals. This measurement process is a part of the quality control process which checks that the quality control factors are being achieved.

RESULTS

To build a complete software system we need decisions. To make these decisions right, a new concept appears, that is what we call a design fragment. In addition to needing quality control to mange it, design fragment also needs data to make control on these fragments.

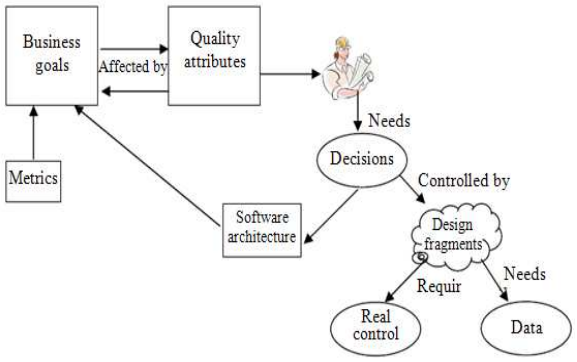

Fig. 6: Achieving goals through architectural design decisions

Quality control in this case checks the process to ensure that the high quality of the software is achieved, so managing the control makes sure that the software developer has followed project quality standers and procedures.

Every software system that is built upon these decisions needs metrics to measure the quality in order to make a decision about the system if it has achieved the specific goal or not. Goal Question Metric (GQM) is one type of the metrics. Figure 6 summarizes the idea of this study and explains the relation between the goal and ADD.

DISCUSSION

From Fig. 6 we can remark the following points on measurement:

- Measurement is a mechanism to evaluate any product; it allows evaluating the quality of the specific process or products
- The effectiveness of the measurement must focus on a specific goal. This means that measurement must be defined in a top-down approach
- Measurement must identify with the environment of the organization and its goals

CONCLUSION

A goal is a condition that deals with the world that the stakeholders would like to achieve. Achieving goals are affected by decisions that are made through building the architectural design. This study describes in detail the process that is used to achieve goals through architectural design decisions; it illustrates the concept of a design fragment and its role on building the design decisions. Completing the final goal needs to be measured. This study shows the position of measurement in the process.

REFRENCES

Ab-Rahman, M.S., S.A.C. Aziz and K. Jumari, 2009. Protection for an immediate split structure of tree-based EPON Architecture-ideal condition analysis. Am. J. Eng. Applied Sci., 2: 372-380. http://www.scipub.org/fulltext/ajeas/ajeas22372-380.pdf

Ahmad, S.N., M.F. Zolkipli and A.N. Abdalla, 2009. Intrusion preventing system using intrusion detection system decision tree data mining. Am. J. Eng. Applied Sci., 2: 721-725.

Berenbach, B., D. Paulish, J. Kazmeier and A. Rudorfer, 2009. Software and Systems Requirements Engineering: In Practice. 1st Edn., McGraw Hill, New York, ISBN: 10: 0071605479, pp: 356.

Bachman, F., L. Bass and M. Klein, 2003. Moving from quality attribute requirement to architectural decisions. Proceeding of the 2nd International Software Requirements to Architecture Workshop, (ISRAW'03), Porltand, Oregon, pp: 1-8.

Barbacci, M.R., R. Ellison, A.J. Lattanze, J.A. Stafford and C.B. Weinstocks *et al*., 2002. Quality Attribute Workshops. 20th Edn., CMU/SEI, Pittsburgh, pp: 40.

Bass, L., P.C. Clements, R. Kazman and R. Nord, 2009. Architectural business cycle revisited: A business goals taxonomy to support architecture design and analysis, the NEWS AT SEI. Software Engineering Institute. http://www.sei.cmu.edu/library/abstracts/news-at-sei/architect20052.cfm

Fairbanks, G., 2007. Design fragment. Ph.D. Thesis, Carnegie Mellon University. http://reports-archive.adm.cs.cmu.edu/anon/isri2007/CMU-ISRI-07-108.pdf

Garland, D., 2000. Software Architecture: A Roadmap. In: The Future of Software Engineering, Finkelstein, A. (Ed.). Carnegie Mellon University, ACM Press, ISBN: 1581132530, pp: 1-9.

Gross, D. and E. Yu, 2001. Evolving system architecture to meet changing business goals: an agent and goal-oriented approach. Proceedings of the 5th IEEE International Symposium on Requirements Engineering, Aug. 27-31, IEEE Xplore Press, Toronto, Ont., Canada, pp: 316-317. DOI: 10.1109/ISRE.2001.948602

Jalote, P., 2008. A Concise Introduction to Software Engineering. 1st Edn., Springer, New York, ISBN: 10: 1848003013, pp: 272.

Jansen, A. and J. Bosch, 2005. Software architecture as a set of architectural design decisions. Proceeding of the 5th IEEE/IFIP Working on Software, (IFIPWS'05), IEEE Xplore Press, Pittsburgh, PA., USA., pp: 109-119. DOI: 10.1109/WICSA.2005.61

Johannes, A.G.J., 2008. Architectural design decisions. Ph.D. Thesis, University of Groningen. http://www.narcis.info/publication/RecordID/oai:ub.rug.nl:dbi%2F4899b138bb0a6

Kan, S.H., 2002. Metrics and Models in Software Quality Engineering. 2nd Edn., Addison Wesley, USA., ISBN: 10: 0201729156, pp: 560.

Khaled, L., 2010. Driving architectural design through business goals. Int. J. Comput. Sci. Inform. Syst., 8: 68-71. http://www.doaj.org/doaj?func=abstract&id=563920

Liu, L. and E. Yu, 2001. From requirement to architectural design-using goals and scenarios. Proceeding of the ICSE-2001 Workshop: From Software Requirements to Architectures, (STRAW'01), Toronto, Canada, pp: 22-30. http://www.cs.toronto.edu/pub/eric/STRAW01-R2A.pdf

Omar, M.K. and A. Ajitha, 2008. Two mathematical modeling approaches for integrating production and transportation decisions in the process industry. Am. J. Applied Sci., 5: 1023-1028. http://www.scipub.org/fulltext/ajas/ajas581023-1028.pdf

Perry, D.E. and A.L. Wolfs, 1992. Foundations of the study of software architecture. ACM SIGSOFT Software Eng. Notes, 17: 40-52. DOI: 10.1145/141874.141884

Somerville, I., 2006. Software Engineering. 8th Edn., Addison Wesley, USA., ISBN: 10: 0321313798, pp: 864.