Correspondence between Software Requirements and Architecture

Wajid Rafiq† Nadeem Sarwar†† Muhammad Bilal†††, Iqra Ishaq††††, Abdul Rauf †††††, Muhammad Abbas††††††, Muhammad Sohail†††††††

NUST, Pakistan, University of Gujrat Sialkot Sub Campus, Pakistan, The Islamia University of Bahawalpur, Pakistan

Abstract
Most of the software community would accept that software architecture and software requirements have a strong association. Whatever, requirement analyst capture during requirement gathering phase, it is exactly depicted in software design. However some experts has different perspective about this correlation, they argue that requirements are just description of the problem itself and architecture is the abstract structure of software system, in which different main elements are connected with each other. New trends in the field of software architecture have changed the perspective of considering software architecture as abstract structure only to a wider scope of architectural knowledge. Additionally, these new trends assigns a first class status to architectural design decisions of the software. In this paper we argue that, fundamentally there is no difference between software requirements and architectural design decisions, by adapting this comparatively the latest point of view, we can identify those areas where both requirements and architecture communities can help each other.

Keywords:
Software Design and Architecture; software Requirement Engineering; software architectural knowledge.

1. Introduction

For the last ten years the correlation between software architecture and software requirements is a reasonably important topic. In a symposium on the topic of requirement engineering in the year 1994, some authors presented the idea of software requirements to architecture. Most of the other authors were trying to figure out the difference between requirements and architecture [1]. Garlan talked about the problem space and solution space and difference between them. Jackson talked about the difference between application domain and machine domain [2]. Mead discussed that architect thinks in terms of developer's point of view and on the other side requirement analyst thinks from the customer's point of view. Potts explained the difference between "What" and "How", here "What" means requirements and "How" means design. Reubenstein told requirements work as an index toward a solution and resist the as-built architecture. Similarly, Shekaran also define the difference between problem and solution like Garlan, in addition it also talk(s) about that there is somewhere relation exists between two of them. The main point is that these two disciplines evolved independently from each other, there are the fields to which they should be correlated have still vast gape to be explored [3]. There are different starting points of any software system that is to be built; there is a choice for the startup, either to choose requirements or software architecture. Both choices will result in different software development life cycle [4].

Requirement engineering is a field which concerns with the elicitation of the goals that a user wants to accomplish with the software system. Now the goals are specified in the form specifications and constraints after that the responsibilities of the intended requirements are distributed among agents like humans of, available software or software to be developed and devices [9]. The requirements specify problem domain [10], so they should answer the questions such as:

1. What are the problem phenomena?
2. What is the cause relation of those [11] phenomena?
3. On which basis these phenomena are problematical
4. Which stakeholder is involved?

The analysis of the user requirements are identified from top level to bottom and it is further refined until properties of the desired solutions are established [12]. The problem frames discussed by Michael Jackson, in his problem frames, the problem structures which were frequently occurring were recognized and subsequently were particular frame [13].

Jackson used the own terminology as indicative was used for the stated choices. The optative word was used for the selected choices machine specification. Problem domain and requirements are all come under problem analysis field. Usually problem domain is static portion, it relates to the place of the problem. And this portion has the indicative domain to which the problem part relies. The requirements are the optative explanations of what the client would identify to be true in the problem side [13]. The machine specifications and the actual behavior of the machine at its interface are different, there are limitations on the behavior and specifications, and this

Manuscript received September 5, 2025
Manuscript revised September 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.9.16

describes the difference between architecture, requirements, problem domain and solution.

Software architecture is a collection of design decisions; the software architecture is a high level structure of a software system, which comprises usually of components and connectors. In early 2000s, the software architecture field was considered structure oriented, but research in this field, made a shift from structure oriented view to knowledge concentrated view. The architecture is not a solution structure(,) but is a collection of design decisions which resulted that structure [14].

In this research paper we will discuss different approaches used in collaboration of requirements and architecture. Different author's perspective will be summarized. Different models of collaboration of requirements and architecture will be discussed. We will discuss the architectural design decisions in detail. And at the end, we will conclude our discussion of what architectural choices are most appropriate for software development in the context of software requirements.

2. Different Approaches on Software Requirements and Architecture

After the discussion of this panel, many research communities started to work on this widely discussed topic Software requirements and Software architecture and also tried to reduce the gap between them. Many people do numerous [2] attempts to find different approaches to integrate the requirements and architecture. These attempts can be listed down in terms of different approaches like

1. Problem domains model
2. Twin Peak model
3. CBSP approach

These types of studies give awareness and clear idea about the exact relationship among software architecture and software requirements. Therefore, according to the Garlan research this intrinsic relation is habitually based on the difference between requirements and architecture and requirements represent the problem domain and architecture represents the solution domain.

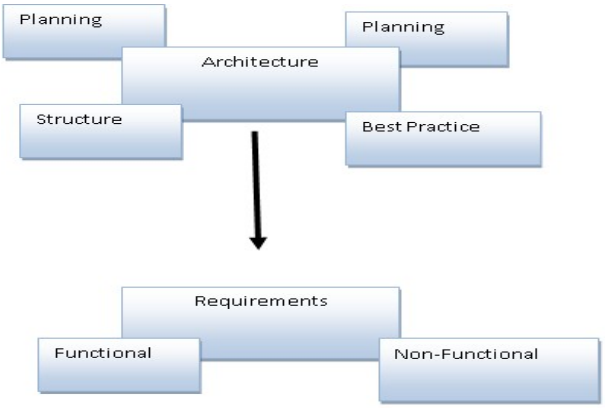

Fig 1: Requirements to Architecture

3. Requirement Engineering and Architectural Design

In Figure 1 it is clearly shown that architecture is an exact mapping of requirements. What is written in the requirements must be depicted in the architectural design. We can draw a line between the requirements and architecture. Requirements are considered with the analysis of the problem domain and architecture with the solution domain. According to the recent development in Software design and architecture domain we imagine that there is no basic difference between so called requirements and architecture. If we talk about more precisely than in fact architecturally important requirements are those which represent the essential design decision. According to some people this hypothesis is very proactive it needs more corroboration. Requirement engineering and software design architecture has been passing through many advents which are listed below here.

Table 1: RE vs. Architecture design

Requirement Engineering	**Architecture Design**
• Goal [3] oriented Requirement Engineering	• Pattern based research Architecture Design
• Use case oriented Requirements Engineering	• Architectural style based research Design
• Sociology and linguistics bases Requirement Engineering	• Attribute based Architecture Design
○	• Component based Architecture Design
○	• Product line based Architecture Design

However, both requirements and architecture are emerging separately from each other and mutual interest areas of both fields yet to be dig down. Different type of architectural design consists of many related problem classes which is related to that particular design research solution classes. There is a very interesting connection between Software domain and Software architecture design domain in software engineering. Recently the research in problem frames related extended towards the architectural patterns and try to investigate the relationship between both of them.

Both requirement engineering and architecture design communities held many conferences and workshops to explore the mutual benefits of both disciplines. In these workshops they discuss what's currently going on and how to improve it further. Mostly the papers presented in these workshops are normally positioned papers, which indicate the author's interest in (the) workshop. These papers are not considered as a formal publication, but they can be used in future as more formal way of publication. Normally the papers which are submitted in these workshops are categorized into 3 groups, which are given in table 2.

Table 2: RE vs. Architecture design research papers

Groups	Papers
Group 1	Paper about moving from requirements to architecture design
Group 2	Paper about moving from architectures to requirements
Group 3	Paper about integration of requirements and architecture design

3. Twin Peak Model of Requirements and Architecture

In Software development field, it's a normal observation that as early as you understand your customer's requirements, it is easy to move towards such a solution which is your customers are expecting. Similarly, before time understanding of architecture provides a basic idea to discover the further constraints related to requirement and architecture, it also helps to evaluate the system's feasibility [4]. Various software development communities often opt different substitute for initiation of software requirements or architectures. Waterfall process model also creates the system architecture that confines the users and developers by doing unavoidable changes in requirements. This was the main drawback of the waterfall development process. Then Spiral process model comes which resolve many deficiencies which were included in waterfall model and offer incremental software development approach, which help developers to easily evaluate and change the requirements according to the project risks. The Spiral process model reflects both necessities and realities of software systems. Spiral process model also acknowledges the need of development of software architecture that are yet stable and changeable in frequently changing environment. The purpose of this model was to enable developers so that they can work on requirements and architecture at the same time [5].

From Figure 2 it clearly shows that twin peak model develop comprehensive requirements and architectural specifications in an incremental manner. This model we can say is the next version of Stephen Mellor's and Paul ward Development [6] model which they proposed for Real Time Systems. In software development, the change control is a fundamental problem. The Twin peak model is prone to change in a controlled way as they occur. Analysis and identification of core software requirements are necessary for the stable software architecture while changing requirements. Different processes are used to develop software systems in this context. Using COTS components means, re-using some built-in products at an earlier stage of requirements. If we want to be able to compete with the changing requirements environment, than we have to accomplish all the development activities quickly.

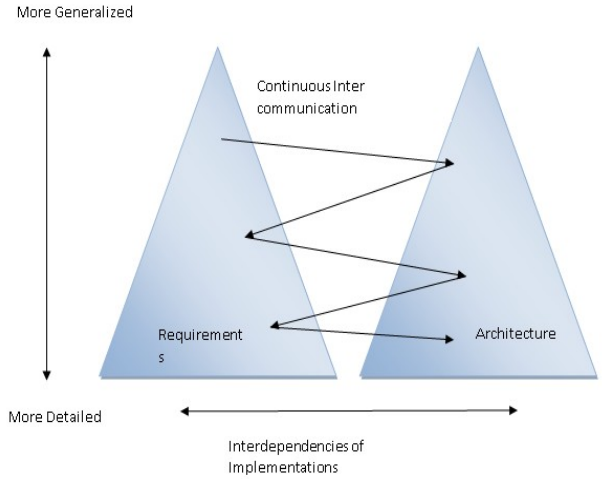

Fig 2: Twin Peak Model

3.1 Twin Peak Model Management Concerns

Nowadays, many software development organizations use twin peak model approach and deal with requirements specification and design issues at the same time, except for some fine grain problem domains and firm contractual procedures. However, doing requirements and architecture design phase separately is very tricky because there are some restriction applies on developers, so that developers are not able to pay attention on their own part in software development. In actual, architectures can restrain designers to meet the specific requirements, and

collection of requirements can also help designer to build architecture. If we see industrial software development, then most of the people moving toward spiral process model. It is normally known as Twin peak model to emphasize that the requirements and architecture have an equally contribution in software development. Barry Boehm identified that Twin peak model address following 3 management concerns:

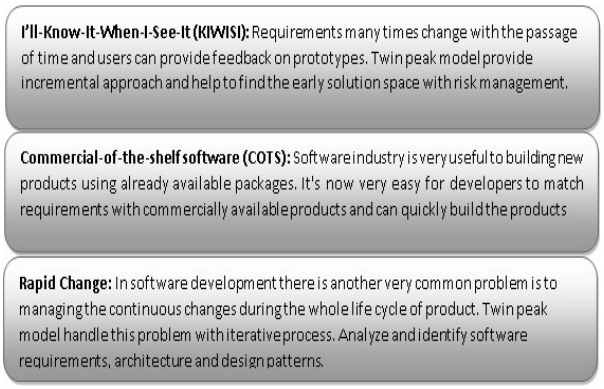

Fig 3: Twin Peak Model management concerns

. Recently most of the Software design communities have already identified many design patterns to express the range of many implementations. These software design communities also recognized many suitable architectural design patterns to meet the various types of global requirements. These Software Design and Requirements communities also encourage the usage of problem frames and analysis, pattern frames of Michael Jackson's and Martin Fowler respectively, to discover the problems for which solution exists.

3.2 Building Modular Software Incrementally

If we build software systems with well-defined interfaces, which offers capabilities such as maintenance and re-use of components. Software design community has developed design patterns for a specific implementation. To meet various global requirements, software architecture community has developed various architectural styles. Using Michael's problem frame and Martin's analysis patterns we can identify problems for which solution domain exists.

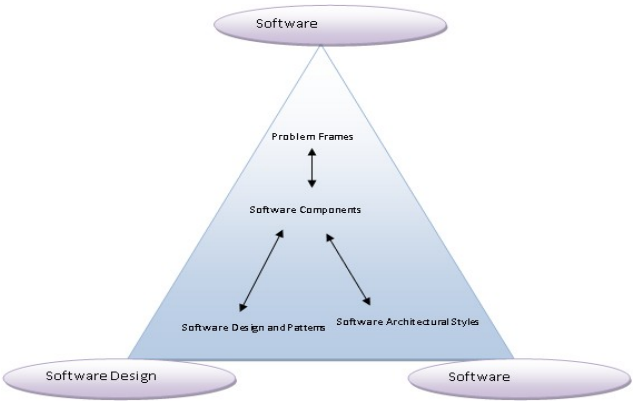

Fig 4: Software Architectural Patterns Design, and Requirements

In above Figure 4 it clearly shows that in software development environment requirements, architecture and design get equal attention. Styles and patterns which are adopted by the developers according to the system components which need to build and the relationship between these components. These patterns are connected by a relationship, figure shows that how different requirements, design and architecture patterns are connected and which one is the initial point for component based development. The predetermined architecture can easily limit the types of problems. It can be used to develop different types of design; on the other hand rigid requirements pose limitations on the architecture and design choices. From the perspective of requirement engineering, problem structuring can be achieved by using problem frames. In the given context, the existing architectures can influence the perception of (developers of structuring the problem, some problem frames are reverse engineered from existing designs.

3.3 Weaving the Development Process

The twin peak model is similar to Kent Back's extreme programming approach. So that the goals of exploring possible implementations of the given context is early and iterative. The twin peak model is harmonized in XP that's why it put more focus on front end software development activities, architectures and requirements. Large scale projects can be efficiently managed if the requirements are understood early and the choice of architecture is made in accordance with the requirements. The XP focus on production of code whether it is at the expense of requirements or architecture. On the other hand, focusing only on requirements and architecture, resist scalability issues. Iteration and modularity are also important. The twin pea model is in itself is iterative, integrating it with components which are tested and

derived from well understood prototype can help in development of large scale applications in increments.

3.4 Questions in Software Development

Practitioners and researchers are stressed to improve processes that permit fast software development in a modest market, joint with the better analysis and arrangement that is essential to yield high quality software within low budget and time constraints. A faster and ideal development technique allows requirements and architecture of the software system work collaboratively and iteratively to describe the features required. This can improve understanding problems by the developer by considering architectural conditions and the architectures can be built which are based on requirements. There are many questions which are yet to be answered.

Table 3: Basic set of question's in Software development.

Sr. #	Question's
1	What requirements are stable in changing requirements and how are they selected.
2	What type of requirements are stable then other requirements how are they identified.
3	What type of changes we are prone to expect in the software architecture?
4	How can we manage architecture and requirements to minimize changes impact?

The Twin Peaks model signifies already available, but inherent, state of the exercise in software system development. Because it is based on well-known research in its evolutionary development, the software development communities don't accept that such a model signifies acceptable practice. The answers to the above questions will open new fields such as:

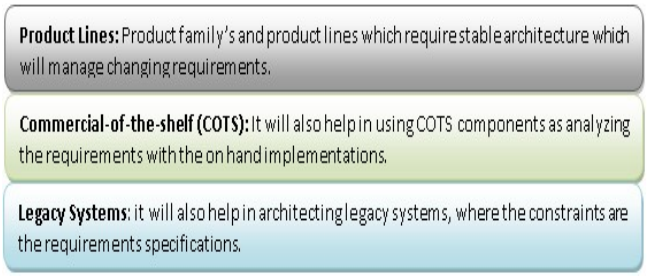

Fig 5: Twin Peak Model as identification of new fields

The development process which makes it possible fast and incremental delivery are essential for the software's that need to be developed quickly and shorter time to market, as a key requirement. So software development communities still aren't identifying such type of model represents the acceptable practice.

4. Boundary between Problems and Solution

Software development life cycle involves two important activities, requirement engineering and software architecting. The objective and purpose of the software system is contained in the requirement engineering process. This process involves requirements to be unambiguous, correct and consistent so that they provide a baseline for further software development, system validation and its evolution. The software architecture is concerned with the solution space. The architecture of the software is explicitly defined and a baseline is prepared on which subsequent development activities will be planned [7].

In software engineering there is always a challenge that how to design software that will fulfill the requirements of the customer and architecture that will maintain the intended behavior in a systematic way [8].

4.1 Problem Exploration by Requirement Engineering

Requirement engineering is a field which concerns with the elicitation on the goals that a user wants to accomplish with the software system. The goals of a software system are given in terms of requirements. The requirement engineering is concerned with the customer choices, which a customer wants particularly from a software system. The solution cannot be separated from the problem analysis, the reason behind that is they will have an effect on the problem area. Architectural frames were introduced by Rapnotti gave a new idea of architectural frames [32]. Architectural elements are depicted by the use of these frames. They applied their approach on the pipe and filter style by creating architectural frames for this style. Pipe and filter style uses components and connectors, components are pipes and connectors are filters, components have the capability to take input and give output. The use of pipe and filter style introduced new problems, concerning scheduling and input to output transformation. The new formalization of the problem, created new requirements and new filters were needed to be planned, to accomplish the original problem [2].

There is always an interplay between the problem analysis and solution considerations, it can happen conceptually but in reality this separation cannot be true. There is always interdependency between them. There is a trade-off between the implementation of certain requirements over others. The architecture depends not only on the requirements that can be satisfied but also on the others which cannot be satisfied. New sub problems (requirements) can also be faced while implementing

certain requirements [2]. So requirement engineering explores the problem domain by working on those requirements.

4.2 Comparison of Software Architecture and Solution Structure

The design decisions usually related to the other design decisions [15]. Kruchten categorizes the design decisions in three categories:

1. Existence Decisions
2. Property Decisions
3. Executive Decisions

Table 4: influence of design decisions

Sr. #	Name of Design Decision	Impact
1.	Existence Decisions	The existence design decisions are not much important to capture, but they are shown in the implementation of the software system.
2.	Property Decisions	Property decisions affect too many elements, and they are implicit and may not be documented, there may be other design decisions made which overlap the previous property decisions.
3.	Executive Decisions	These decisions are all political, personal, financial, cultural and technological constraints. Executive decisions constraints or frame the property and existence decisions.

Architectural design decisions are supreme stakeholders for building challenging software. The relations between the decisions play a good role in the evolution and maintenance of the development of such systems [15].

For example,we are building a Java application and we make a decision that to use JSF, then it limits the use of JMS and a conflict will rise to use PHP. And similarly we use publish/subscribe style, then it will conflict the decision to use peer to peer style and it will constraint with the decision of choosing the publisher technology. The architectural design is broken up into individual design decisions, a new perspective of the architectural design. This process also helps in diverging the focus onto the result of using design decisions [15].

4.3 Discussion on Problems and their Solution

After getting requirements from the stakeholders, a software architect will come to know that there are some requirements which don't play any role in software architecture, for example, if we want to use a matrix based display to show the speed of the vehicle, then we will not consider this requirement in the architecture level discussion. Here we are discussing only those requirements that plays role in the software architecture. We are limiting our discussion in context of requirements to architecture as discussing the architecturally significant requirements.

Poort stresses on the point that the three are conflicts that arise by having conflicting requirements in the domain of solution [16]. Kozaczynski supports this argument that the main requirements cannot be understood until the architecture is not baseline [17]. Hofmeister worked on five different architecture designs, he summarized that all the architectures started in a non-sequential manner and goals and the limitations are demarcated when the architecture was finalized [18].

Savolainen and Kuusela presented a mixed design of requirements specifications, they support that the design details which don't include any option are highly forced requirements [19].Seemingly the choice of mandatory performance in a particular situation has an immense effect on architecture of a software system associated with the under discussed problem. When we make a choice of using a particular architectural domain and style, it will have an effect on the problem side because there are new requirements according to the style used for which new design decisions has to be taken. In this discussion(,) we can conclude that architecturally significant requirements (ASRs) and architectural design decisions (ADDs) have not well defined relationship, the sources and the outcomes cannot be implemented in the problem or solution domain. So contrasting problem and solutions is merely a false contradiction.

5. Architectural Design Decisions and Requirements

The problem frames classify the problem in (a) structure, analysis and the area in which the problem is situated, which is called problem space [13].This approach uses problem rather than (the) solution, this approach uses the problem domain to allow the owner who have(has) the knowledge of the base of the problem to initiate and run the requirement phase. Problem frames has(are) following different variations:

Table 5: different situations, solutions and the resource to attain a solution

#	Situation	Solution	Solution Attained by
1.	Trade-offs between requirements	Owner and solution engineers negotiation [20]	Software engineer's field knowledge
2.	New software development	Existing components, [21] frameworks [22]and architectures are used	Selection of architectural styles, pattern development, selection of field specific architectures,
3.	Experts workers	Express their expertise through development, even for modified software.	reuse of past Development experience.

5.1 Indicative and Optative Statements

ASRs and ADDS) are all same in a way(,) if way if we consider them as optative statements. ASRs and ADDs constraint other decisions and sometime themselves are constrained by some decisions. This has meanings that the telling problem domain accompanies indicative specifications. In this sense the decisions that are attained are categorized into two categories, one of them is named ADDs and one of them named as requirements.

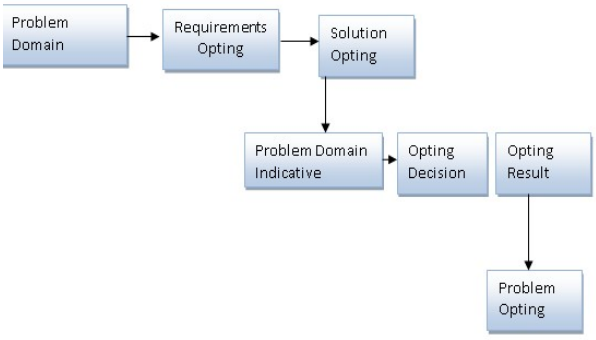

Fig 6: Indicative properties of optative

An example is discussed here about an application, a Package Router, for example, when the hardware is selected and its properties are defined then this will be considered as the indicative properties and if we build its architecture then it will define limitations about the requirements and design. Then we have to define its software properties, so we have to make the decisions about the architecture to finalize the architecture. This is quite natural because if we define the architectural style then it will pose limits on the requirements conditions. This is slightly different with a hardware already determined variation as in this situation we are determining the architecture of our hardware also because it was not provided. This concludes that in the first situation the properties of the hardware was part of domain problem itself so it used requirements, while in our second situation the properties of the intended hardware was part of the problem and it had effect on the problem side [2].

Both ASRS and ADDs have affects that are similar to the development of the software system, the preferences for the desired implementation and ruling out the features that are not desirable. This can only be possible and implementable for the opinion that ADDs are an important part of the knowledge of architecture and the intended architecture is not just a mere structure. .

5.2 Architectural Decision Loop

The decision loop that is drawn in the diagram below shows the relation between architectural design decisions. It shows that design decisions are taken, and they introduce new design decisions that are to be made with respect to the previous design decisions [23].Relations in between ADDs is modeled in a loop called decision loop. Here by using the phenomena the ADDs put forward latest matters, for those matters new ADDs are to be engaged. Decision loop can be viewed in the Figure 7. Implementation needs to take some design decisions, requirements reflect some design decisions, and some of them are limiting decisions. For example, if there is a situation that we want to have check on the data storage, different scenarios can solve this situation, when one method is selected this, becomes a design decision, and this decision topic creates various latest assessments for example the circulation of the events generated by the system to the intended components. A New instance of example is, about the space concerned, according to this condition the software system should handle data that in increasing in amount continuously. We will consider this condition as an ADD because new requirements will be accomplished by this single requirement. The Rationale has an important impact of the RE process for instance, taking an example of goals concentrated RE procedure [9]. Taking a goal and the need of goal is categorized, now high level goals can be signified. How the system will help satisfy a goal, this goal will further refine new goals. . New instance of example is, about the space concerned, according to this condition the software system should handle data that in increasing in amount continuously. We will consider this condition as an ADD because new requirements will be accomplished by this single requirement. Rationale has an important impact of the RE process for instance taking an example of goals concentrated RE procedure [9]. Taking a goal and the need of goal is categorized, now high level goals can be

signified. How the system will help satisfy a goal, this goal will further refine new goals. Bi directional process of Decision Making

Holyok and Simon discuss the decision making as a bidirectional process. In this process and point of view, there is no proper distinction between the problem and decision as in the problem of requirements and design decisions. This type of decision making process plays (a)role in conflicts, ambiguity characteristics that accompany in maximum SE environments [25].

5.3 Repository of ADDs and Requirements

Here the authors defined a combined repository of ADDs and ASRs and name it as a magic well. The user intention dominates the choice, ASRs and ADDs are extracted from the magic well,according to the intention or the user of the magic well. The perception of the statements as we do is merely different to the perception of the magic well.

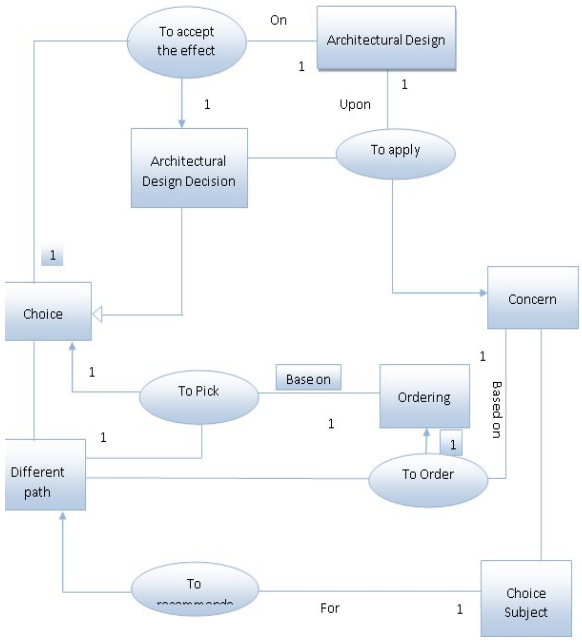

Fig 7: Architectural decision loop

The relation between the RE to architecture with reference to the magic well is elaborated in table 6.

Table 6: Methods and techniques: Requirements engineering versus architecture

Sr. #	Requirements Domain	Repository as Magic well	Architecture Domain
1	Requirements elicitation	Formation of statements	Choice making of architecture
2	Requirements negotiation		Architectural Exchange analysis
3	Requirements Descriptions	Storage of statements in the repository	Architectural Design
4	Requirements Validation	Relate repository data with certainty	Architecture Valuation
5	Requirements Documents	Writing down the repository	Architectural Explanation
6	Requirements Administration	Organizing the repository	Knowledge management of the architecture

Architecture and requirements uses own criteria to see the well. The new requirements are extracted and dropped in the well is the process of requirement specifications. Design and what architects call the design. The requirements that are elicited or architectural design decisions should explicitly be defined. Because if this process is not performed, then it will result in the forgotten of the requirements, architectural requirements and architectural design decisions both have their own way of expression, i.e. formal language specification, UML diagrams, ER diagrams, sequence diagrams etc. [2].

5.4 Architectural Design Decisions an Example

Roller gives an example of some material floating in compound with architectural design decisions. If the material is not touched for a moment, it will sink and disappear [28]. If we drop the statements in the well and don't check them, then this phenomena will be observed. Architectural statements, management has received a great interest in requirement engineering as well as architectural community.

Both architectural decisions and requirements management consider the well not as statements repository, but the well but as an information database. Both requirements and architecture should impose a structure of their own on the well contents which will capture the connections between architectural design decisions and requirements [2].

6. Requirements Discussion

Requirement engineering is a field which concerns with the elicitation of the goals that a user wants to accomplish with the software system.

6.1 Requirements Elicitation

The requirement engineering pays a great attention towards elicitation techniques the elicitation can be focused groups, interviews, and prototyping and use cases. After elicitation every requirement is given a relative weight by requirement negotiation process. The architecture pays attention of use of a particular alternative as what architecture to use and which is not used; selection of a specific architecture uses an approach of tradeoff analysis. In the software architecture the requirements are not processed as they are elicited, but they are less formally implemented as they are elicited. Bass explained the business life cycle in the paper they focus on addressing stake stakeholders and their requirements but they didn't discuss the methods of eliciting those requirements [20]. While applying requirements and architectural design decisions a certain amount of creativity is required for example, for requirements, architectural solutions are needed.

6.2 Requirements and Architectural Validation

Validation is an important part of requirement engineering and software architecture. The name evaluation or assessment is given to architectural validation. Various approaches have been devised by the architecture community for architectural assessment and their impact on software quality for example, modifiability [26] or they consider different quality issues and the relationship between those quality issues [27]. Different approaches are used for requirements. Validation such as reviews and inspections. But those techniques are less formal. In both of these methods, the technique of validation is different but they share much in common. Organizations usually use an approach which is similar to cafeteria like, this method is to hire the snippets of the methods to be used which are according to the particular and specific to the under discussion scenario. Usually scenario based methods are used both in architectural assessment and requirements validation.

7. Cross Fertilization

In this topic we discuss that how software engineering community can benefit from architecture and vice versa. Architecture business cycle is a model for architecture community other than that there is no other model for requirements elicitation. It is not the point of any concern because, according to the magic well elicitation is not the architect's job. Architectural business cycle concentrates on the interaction between stakeholders and understanding of their requirements. Business goals and stake holder's requirements play a major role for software architects. The main approach in this area is the management. The ADDs and related knowledge of the architecture are of great interest in relation to the management. This management is the most recent field in the architecture community. There are many issues that requirement engineering society is facing; some of them are evolving, modifiability, traceability, and rationale and evolution management. The management of the knowledge of architecture accompanies the building of frameworks which will arrest the knowledge [29]. There are different approaches of requirement engineering management which are similar to the knowledge of architecture management. Goals plays very important role in requirement engineering, while the management of the architecture includes areas such as traceability, conflicts discovery and exploring new design variations [30].

8. Future Work

Currently requirement management and architectural knowledge management are considered different information wells. Both fields compare challenges to the each other, but they don't pay attention to the other perspective of similarity between those fields. With a broader perspective, we can realize that both areas are concerned with a same problem but looking at them a problem has different angel for both or the areas. If we see deeply we will find that the requirement management has some areas that are still to be explored and there are many issues in this area in contrast with the management of architectural knowledge. Further exploration of this are will open new horizons for the requirement management. Both communities can learn from each other's experiences. The discussion on the magic well elaborates that architecture is not just the responsibility of the software architect, but requirements that are architecturally significant can shape the architecture. This is because that both fields are complementary to each other. We cannot do anything without the consideration of either of them. Because they are complementary, we require both of them. Both fields have overlaps) but they use different perspectives,so we should use our techniques and methods to surf common goals rather than differences. Further exploring cohesions between architecturally significant requirement and architecture design decision will serve that goal.

New research areas in the field of goals modeling can be explored in this field the research questions are:

1. What arerequirement interdependencies?
2. How interdependencies are identified?
3. How requirement interdependencies are described?
4. How requirements can address interdependencies?

In architectural knowledge management, interdependencies have important characteristics. The reasoning proposed by Kruchten tries to answer those

questions. The management areas of both requirements and architecture because of the emergence of these fields a one to one collaboration and exchange of results of research is very useful to all the stakeholders, concerned to(in) this particular field.

9. Conclusion

The conclusion that can be extracted from this research paper is that ADDs and ASRs are on the equal level of significance. The difference may be the point of view through which they are observed. Some people may disagree, but this paper plays a constructive role in this area, it will also open new doors towards tighter collaboration between the two fields. The requirements at the level of software architecture are very important to build the software architecture. We have worked to change the viewpoint of a different perspective of both fields to a closer and collaborative view. The analysis of the requirements should be done by considering the architecture of the software to be built. So we can get architecturally significant requirements and hence we can build an architecture which is more collaborating with requirements. Architectural design decisions plays a key role in software evolution and maintenance, so we should use a proper mix of architecturally significant requirements and architectural design to build a product that will be maintainable and evolving with time as well as it will accomplish the customer requirements.

References

[1] Shekaran, C., Garlan, D., Jackson, M., Mead, N.R., Potts, C., Reubenstein, H.B., 1994. "The role of software architecture in requirements engineering". In: First International Conference on Requirements Engineering (ICRE), pp. 239–245.

[1] [2] Remco C. de Boer , Hans van Vliet , VU University Amsterdam, Dept. of Computer Science, De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands, "On the similarity between requirements and architecture"

[2] [3] STRAW '03 Second International ,Portland, Oregon , 2003"Software Requirements to Architectures Workshop"

[3] [4] Bashar Nuseibeh, the Open University, "Weaving Together Requirements and Architecture""

[4] [5] W. Swartout and R. Balzer, "On the Inevitable Intertwining of Specification and Implementation," Comm. ACM, vol. 25, no. 7, 1982, pp. 438-440)

[5] [6] Paul Ward, Stephen Mellor's, vol. 1, Prentice Hall, " Structured Development for Real-Time Systems: Introduction and Tools"

[6] [7] Dongyun Liu, Hong Mei. , Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P.R.China liudy@cs.pku.edu.cn, meih@pku.edu.cn , "Mapping requirements to software architecture"

[7] [8] R.G. Dromey, Software Quality Institute, Griffith University, Nathan, Brisbane, Qld., 4111, AUSTRALIA rgd@cit.gu.edu.au , "Architecture as an Emergent Property of Requirements Integration"

[8] [9] A. van Lamsweerde, Requirements Engineering in the Year 00: A Research Perspective, Keynote paper, Proc. ICSEí2000 - 22nd Intl. Conference on Software Engineering, IEEE Press, June 2000.

[9] [10] Ant´on, A., Potts, C.: "The use of goals to surface requirements for evolving systems". In: International Conference on Software Engineering (ICSE'98), IEEE Computer Society (1998) 157–166

[10] [11] Dardenne, A., Lamsweerde, A.v., Fickas, S.: "Goal-directed requirements acquisi-tion". Science of Computer Programming 20 (1993) 3–50

[11] [12] Yu, E.: "An organization modeling framework for information systems requirements Engineering", In: Proceedings of the Third Workshop on Information Technologies and Systems (WITS'93). (1993)

[12] [13] Jackson, M., 2001. Problem Frames: "Analyzing and Structuring Software Development Problems". ACM Press Books, Addison-Wesley

[13] [14] de Boer, R. C., Farenhorst, R., 2008. In search of "architectural knowledge". In: Third Workshop on SHAring and Reusing architectural Knowledge (SHARK), Leipzig, Germany.

[14] [15] Kruchten, P., 2004. "An ontology of architectural design decisions in software-intensive systems". In: Second Groningen Workshop on Software Variability Management, Groningen, NL

[15] [16] Poort, E.R., de With, P.H., 2004. "Resolving requirement conflicts through non-functional decomposition". In: Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA), IEEE Computer Society, p. 145

[16] [17] Kozaczynski, W., 2002. "Requirements, architectures and risks". In: Tenth Anniversary Joint IEEE International Requirements Engineering Conference (RE), p. 6

[17] [18] Hofmeister, C., Kruchten, P., Nord, R.L., Obbink, H., Ran, A., America, P., 2007. "A general model of software architecture design derived from five industrial approaches". Journal of Systems and Software 80 (1), 106–126

[18] [19] Savolainen, J., Kuusela, J., 2002. "Framework for goal driven system design". In: Twenty-sixth Annual International Computer Software and Applications Conference, p. 749

[19] [20] Bass, L., Clements, P., Kazman, R., 2003. "Software architecture in practice", second SEI Series in Software Engineering Addison-Wesley Pearson Education, Boston

[20] [21] [J. Cheesman, J. Daniels, UML Components: "A Simple Process for Specifying Component-Based Software", Addison-Wesley, 2000.]

[21] [22] D.F. D'Souza, A.C. Wills, Objects, Components, and Frameworks with UML : "The Catalysis Approach", Addison-Wesley, 1998

[22] [23] De Boer, R.C., Farenhorst, R., Lago, P., van Vliet, H., Clerc, V., Jansen, A., 2007. "Architectural knowledge: getting to the core". In: Third International Conference on Quality of Software-Architectures (QoSA), vol. 4880 of LNCS. Springer

[23] [24] van Lamsweerde, A., 2003. "From system goals to software architecture". In: Bernardo, M., Inverardi, P. (Eds.),

Formal Methods for Software Architectures, vol. 2804 of LNCS. Springer-Verlag, pp. 25–43

[24] [25] Holyoak, K.J., Simon, D., 1999. "Bidirectional reasoning in decision making by constraint satisfaction". Journal of Experimental Psychology: General 128 (1), 3–31

[25] [26] Bengtsson, P., Lassing, N., Bosch, J., van Vliet, H., 2004. "Architecture-level modifiability analysis (ALMA"). Journal of Systems and Software 69 (1–2), 129–147

[26] [27] Kazman, R., Bass, L., Webb, M., Abowd, G., 1994. SAAM: "a method for analyzing the properties of software architectures". In: Sixteenth International Conference on Software Engineering (ICSE), Sorrento, Italy, pp. 81–90.

[27] [28] Roeller, R., Lago, P., van Vliet, H., 2006. "Recovering architectural assumptions". Journal of Systems and Software 79 (4), 552–573

[28] [29] Ali Babar, M., de Boer, R.C., Dingsøyr, T., Farenhorst, R., 2007. "Architectural knowledge management strategies: approaches in research and industry". In: Second Workshop on SHAring and Reusing architectural Knowledge Architecture rationale and Design Intent (SHARK-ADI), Minneapolis, MN, USA

[29] [30] Rolland, C., Salinesi, C., 2005. "Modeling goals and reasoning with them. In: Aurum", A., Wohlin, C. (Eds.), Engineering and Managing Software Requirements. Springer-Verlag, Berlin Heidelberg, pp. 189–217

[30] [31] van Lamsweerde, A., 2001. "Goal-oriented requirements engineering: a guided tour". In: Fifth IEEE International Symposium on Requirements Engineering (RE). pp.249–263

[31] [32] Rapanotti, L., Hall, J.G., Jackson, M., Nuseibeh, B., 2004. "Architecture driven problem decomposition". In: Twelfth IEEE International Requirements Engineering Conference (RE), pp. 80–89.

[32] Sajjad, R., & Sarwar, N. NLP based verification of a UML class model. In *Innovative Computing Technology (INTECH), 2016 Sixth International Conference on* (pp. 30-35). IEEE.

[33] Kelley, K. Automated test case generation from correct and complete system requirements models. In Aerospace conference, 2009 IEEE (pp. 1-10). IEEE.

[34] BAJWA, I., & SARWAR, N. AUTOMATED GENERATION OF EXPRESS-G MODELS USING NLP. *Sindh University Research Journal-SURJ (Science Series)*, *48*(1). (2016)

[35] Cheema, S. M., Sarwar, N., & Yousaf, F. Contrastive analysis of bubble & merge sort proposing hybrid approach. In *Innovative Computing Technology (INTECH), 2016 Sixth International Conference on*(pp. 371-375). IEEE. (2016, August)

[36] Ibrahim, M., & Sarwar, N.. NoSQL database generation using SAT solver. In *Innovative Computing Technology (INTECH), 2016 Sixth International Conference on* (pp. 627-631). IEEE. (2016, August)

[37] Sarwar, N., Latif, M. S., Aslam, N., & Batool, A. Automated Object Role Model Generation. *International Journal of Computer Science and Information Security*, *14*(9), 301. (2016)

[38] Aslam, N., Sarwar, N., & Batool, A. Designing a Model for improving CPU Scheduling by using Machine Learning. *International Journal of Computer Science and Information Security*, *14*(10), 201. (2016)

[39] Ahmed, F., Khan, A. H., Mehmood, J., Sarwar, N., Ali, A., Mehboob, M., & Waqas, A. Wireless Mesh Network: IEEE802. 11s. *International Journal of Computer Science and Information Security*, *14*(12), 803. (2016)

[40] Bajwa, I. S., Sarwar, N., & Naeem, M. A. Generating EXPRESS Data Models from SBVR. *A. Physical and Computational Sciences*, 381.(2016).

[41] Bilal, M., Sarwar, N., & Saeed, M. S. A hybrid test case model for medium scale web based applications. In Innovative Computing Technology (INTECH), 2016 Sixth International Conference on (pp. 632-637). IEEE. (2009, March)

[42] Saeed, M. S., Sarwar, N., & Bilal, M. Efficient requirement engineering for small scale project by using UML. In Innovative Computing Technology (INTECH), 2016 Sixth International Conference on (pp. 662-666). IEEE. (2009, March)

Nadeem Sarwar is a Lecturer in the Department of Computer Science at University of Gujrat Sialkot Sub Campus, Sialkot, Punjab, Pakistan. His primary research interest involve developing Software Models and Natural Language Processing tools for Software Developers, IT Manager and Testers, in particular for Industry related problems. He have 6 years teaching experience of Software Engineering and have 11 research publication in National and International Journal/IEEE Conferences. He doing his PhD in Computer Science from the International Islamic University, Islamabad, Pakistan. Contact him at Nadeem_srwr@yahoo.com and Nadeem.sarwar@uogsiakot.edu.pk