Architecture Centric Development in Software Product Lines

Aurangzeb Khan
DCE, College of E & ME
National University of Science and Technology (NUST), Pakistan

Farooque Azam
DCE, College of E & ME
National University of Science and Technology (NUST), Pakistan

Jahanzaib Khan
DCE, College of E & ME
National University of Science and Technology (NUST), Pakistan

ABSTRACT
Architecture centric development approach is reliable and cost effective in software development process in the software product lines. Traditionally used approaches in software development are very costly and unreliable in term of quality attribute and time to market products. If we are working in the same domain then architecture centric software development is very beneficial. In this technique we will reuse already developed applications components for developing new software instead of developing these software from scratch that are very time consuming and unreliable. To minimize the development time we will reuse components from each phase of development to minimize of development time and provide better quality. Already developed, verified and compatible components will be reused for development of new software. In this paper we will study the architecture centric software development and evaluation which focus on quality attributes of software and provide much more quality than traditional approaches used.

Keywords: SDLC (Software Development Life Cycle), ATAM (Architecture Tradeoff Analysis Method), COTS (Commercial off-the-shelf Software), SRS (Software Requirement Specification)

1. INTRODUCTION
Market demands are changing day by day very fast that are changed ever before in terms of customers' needs and demands and faster delivery of software. Software complexity is also growing very fast in term of inter-networked and enterprise systems. Software industry is also evolving with the day to day business demands. The most effective technique to face these emerging problems in software industry is Architectural centric approach in software development. Traditionally software are developed starting from requirement gathering phase, analysis, design and development. This approach is very time consuming if we are working in same domain and doing same work time and again. In the architecture centric approach already developed architectures are used for the new software belonging to same domain. Set of proven software components are more reliable than newly developed components which improve software quality. Evaluation of business case tells the story of success and failure of the applications that are developed and deployed [1].The reason for reusing software architecture is variation in requirements. Reusing one architecture for every application is as unfavorable as building every application from scratch. So there must be a way that there should be a specific way for developing a specific application from variety of appreciate architectures. As a prerequisite, set of techniques are required for classifying software architecture. For this purpose set of design spaces presented by Lanes is a very good approach [2].There are many quality attributes in the software that we are not sure while developing software from scratch are met but in case of developing software using architecture centric approach we can use set of already proven components in software development that covers all the nonfunctional requirements according to user expectations. In this approach, we will use the set of proven components in every phase regardless of starting from scratch; we will use proven set of requirements, design, and piece of code for software constructions. Reuse activities increase system effectiveness, traceability and repeatability of reuse. The basic idea of our architecture reuse is to refine and implement traditional approaches using already existing skeleton. Some components matched partially [3] Application development is performed using already existing architecture framework that focus on domain specific abstractions identified by particular architecture. In this technique, bottom up approach is used on each stage of development model. Certain set of component are reused to boost up development process. The prerequisite for this approach is that all the reusable components must be compatible and there should be minimum mismatch. Component reuse involves both "development with reuse" and "development for reuse [4]

2. PROBLEM DESCRIPTION
Software industry is growing very fast and researchers are thinking about new techniques and approaches to boost up software development process to develop quality software with low cost. Traditional approaches to develop software are very lengthy and rigid process if software development is carried out in the same domain repeating same activities for every application is the wastage of time. So there should be set of components and artifacts that can fit to the same domain software to minimize development efforts. To minimize this we are going to present a new technique that will be used to minimize development efforts and cost in developing new software, in this technique we will use the knowledge of already developed application in developing new application belonging to same domain. This will minimize the development time and provide high quality application in low cost.

3. PROPOSED MODEL SURVEY
Software industry is growing and adopting new techniques in software development to increase software quality, speed and effectiveness. Traditional software processes are used in SDLC is requirement gathering, analysis, design, implementation, verification and maintenance. The impact of requirement change minimizing the impact of volatility [5].

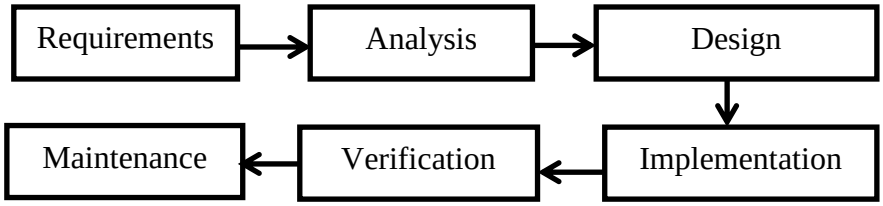

Fig 1: Traditional software development approach

Requirement gathering phase is very time consuming phase and a lot of efforts are required while gathering requirements because requirements are the backbone of any software. If the requirements are not gathered properly this will lead the system towards failure and there will be no use of software if the requirements are gathered poorly. Following phases are involved in requirement gathering

- Requirement elicitation
- Requirement analysis
- Requirement verification

Requirement gathering phase involve the following strategies.

3.1 INTERVIEWS

In this technique interviews are conducted by key stakeholders who have domain knowledge of the application that has to develop. Interviews are conducted by requirement engineers. Interviews conducting activity depend upon the availability of stakeholder in case of global software development process. These interviews are conducted by video conferencing, call conferencing or by using requirements workshop.

3.2 BRAINSTORMING

In this technique different ideas are identified which are priorities by high and low value. Ideas are combined to make a high valued idea. In this type of technique each team member is involved in getting ideas as possible.

3.3 REQUIREMENTS WORKSHOPS

This technique is very useful in requirement gathering phase. Development team's representatives and clients are gathered and encourage sharing requirements and expectations with each other. This technique is time and cost consuming technique. These workshops involves user and cut across organizational boundaries.

3.4 USE CASE

This is the picture of actions that system performs when user (actor) interact with it. It involves the sequence of user actions in descriptive way. It converts client requirements into technical way so that it will help developers to understand the requirements and these facilitate developers while developing software application and it is very effective in distributed environment.

3.5 PROTOTYPE

Prototype technique is used for gathering requirements. Prototype is constructed on well understood requirements and shows to client to elicit further requirements. Then improvements are made in the prototype that is suggested by client. Prototyping technique can be used with other available techniques [6].

Besides above mentioned techniques there are many other techniques that are used for gathering requirements in global software development.

- User interface
- Transactions process diagrams
- Data flow diagrams

Once requirement are gathered and analysis then system design is constructed from analysis requirements, this involve the following activities

- Conceptual design
- Logical design
- Physical design
- Detail design

After designing the system, implementation of the system is started on the basis of SRS document. Testing phase follows system implementation and finally the system is deployed in the user atmosphere.

Above mention activities are very effective if organization is developing the software system of different domain. But working in the same domain and construction of software in the same domain it is time consuming to perform these software activities time and again for the same set of software. To avoid this we are going to adopt architecture centric software development which minimize the software development efforts and use the concepts and knowledge of already develop software in the same domain and reuse components of already verified software. In the traditional model if we change requirements at later stages of the project then the cost of change is increased that is another disadvantage of the traditional approach graph below shows the impact of cost of change at different level of SDLC.

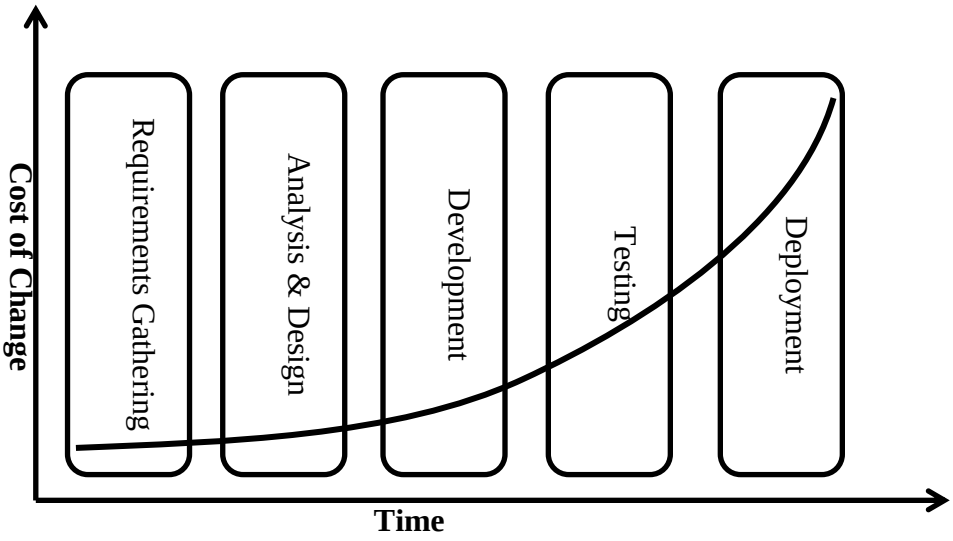

Fig 2: Cost change over SDLC [16]

4. EVALUATIONS AND RESULTS

Table 1.Comparisons of Traditional and Architecture centric approach

Approach	Phase	Time(days)	Cost	Human Resources	Time to Market	Quality
Traditional Approach	Requirement Gathering	30	500000	3	Late	High
	Design	10	100000	2		Normal
	Coding	40	1000000	5		Normal
	Testing	20	200000	2		High
Architecture Centric Approach	Requirement Gathering	10	100000	1	Early	V. High
	Design	2	20000	1		V. High
	Coding	20	500000	5		V. High
	Testing	8	100000	2		V. High

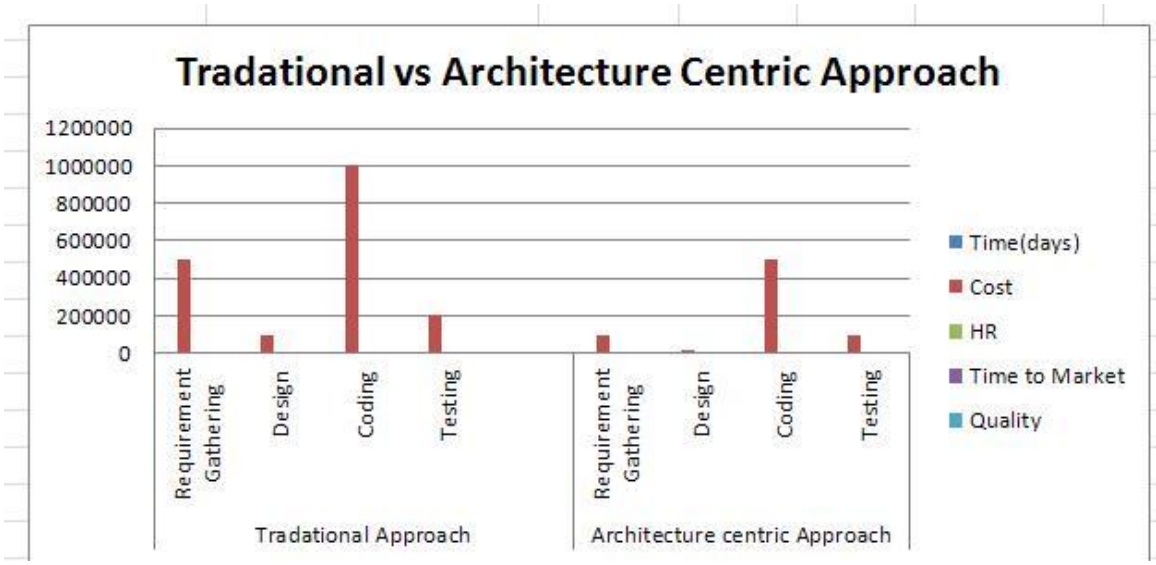

Fig 3: Graphical comparison of both approaches

5. APPLICATION AREA

Architecture centric development in software product line focuses on the software product line. Product lines are those systems which covers a common, managed set of particular features satisfying the specific customer needs of a particular market and that are developed from a common set of architectures in a prescribed way. Software product lines are developed using emerging technology. This type of development approach allow developing companies to realize improvements of magnitude in time to market, productivity, cost, quality, and many other business drivers. Software product line development can also enable applications in rapid market entry and very flexible response, and these applications provide a capability for customization. This will provide the organization capability to overcome the resource shortage problems. Product line mainly focus on the COTS that are available at very low price and can be tailored in the application. The source of the magnitude of improvements by using product line techniques is software architecture reuse commonality throughout the software product line, strategically manage all software product line variation, and it is aggressively eliminate all duplication of developing effort. The model being used its expressiveness' is sufficient the fact that level of improvements are surprising insights in the level of inefficiency that we have to accept as part of software development.

6. EXISTING SYSTEMS

There are numerous approaches in architectural based software development, what components are reused in architectural centric software development and what process are used, these approaches are used as common ground and source of information in developing new techniques. IEEE has introduce approach Recommended Practice for the architectural description of software-intensive systems this standard is used as framework that facilitate adoption of architectural practices and principals to industry and research community. The development of new system in the same domain is take place using plethora of available architectural reuse choices. Architectural reuse development is just like the design decision reuse and structural building blocks like design pattern [7] or whole architectural system. The concept of extended design specification is use that provide the semiformal requirement specification and detail of some components that are reused in new application that are suitable starting for pointing the problems in artifacts that are reused. The problem that most of the researchers faced in the component reuse technique is that it is very difficult to find out the components in advance before software development to guess that which component will work. There should be specific domain knowledge of the software architecture that is going to be developed and often a real problem that is going to be catered and practical experience in using architecture reuse.

Architectural design starts when important system requirements are known. These requirements are called set of architectural drivers and these include the functional requirements and quality attributes. The functionality of the system is the ability of the system to perform the certain task. Estimation efforts made prior to development are very inaccurate [8]. Quality is the nonfunctional requirement in the context of desired functionality. Designing software architecture is moving from software requirements to architectural design decisions; this requires the knowledge and experience from architects. Once decisions about architecture have been placed then constraints on quality attributes are placed [9]. Achieving system quality attributes depends upon the architectural design decisions and these are called tactics and it is the proven quality approach that is very useful for achieving the quality. Quality comes in the requirement due to adoptability of the architecture. To realize one and more quality tactics an architect choose the appropriated architectural style. Architectural style is the repetitive quality approach shows particular quality. Following architectural styles are used in the market

- Pipe and Filter
- Layer Style

- Rule base
- Black board Style
- Object oriented style

Reference architecture is one step ahead from these approaches that reuse the best practices in the architectural reuse. It has the proven best practices in certain product domain [10]. Reference architecture is the blue print for software architecture in developing family software. The reference architecture is very similar to the application framework used in object oriented style and product line. Development of the components from the driven vision of product lines [11]. Project development based on the vision management software [12].
Architectural style is the specialization of view type and shows the repetitive architectural approach that covers the quality attribute independent to any system. This is based on the experience of the architect how well components are reused [13]

Software architecture is the foundation of any software system it represents the set of design decisions that are most difficult to set correct and hard to change at later stages.
Different architectural evaluating techniques are available like ATAM which is the most value able technique that is used to evaluate architecture of software. In this technique stakeholders are people who are most interested in evaluating software architecture to meet quality attributes. Following methods are used to evaluate the architecture

- Prioritize the quality attribute requirement in the form of tree
- Mapping quality attribute with the architectural approaches
- Differentiate between risk and non-risk
- Tradeoff and sensitivity points

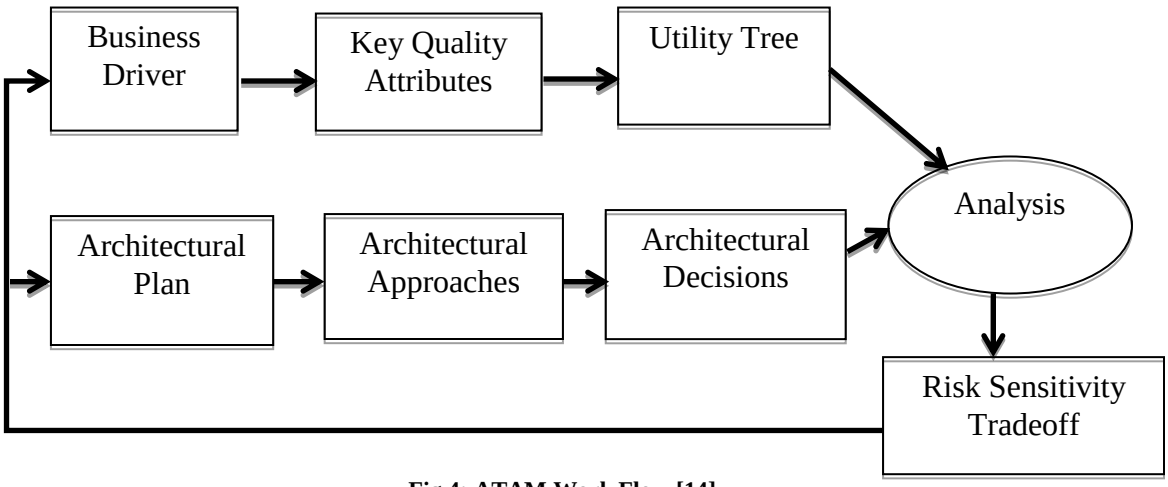

Fig 4: ATAM Work Flow [14]

In the above mentioned techniques, no one is specific to component reuse, some focus on design reuse and some are focusing on architecture reuse and software architecture evaluation in the center point of concern. We are proposing the technique that will focus on the component reuse in each and every phase of SDLC and focus on the quality attributes of the software.

7. POTENTIAL RESEARCH AREAS

After brief analysis of traditional software development method and architecture reuse strategies we are proposing an architecture centric reuse strategy in development of software product line in specific domain. The problems that most of the researchers faced in the component reuse technique are that it is very difficult to find out the components before software development in advance to guess that which component will work. There should be specific domain knowledge of the software architecture that is going to be developed and often a real problem that is going to be catered and practical experience in using architecture reuse. We have considered two projects as a case study in the software product lines. Both have the total duration of 90 working days, the project in which we use the traditional approach for the requirement gathering and documenting each and every requirement and requirement change impact on the software and priorities requirements and analysis of requirement that what requirements are feasible to implement and what requirements are not feasible to implement requirement gathering and requirement analysis procedure takes 30 working days and 3 full time resources work on this activity after completing SRS document then this document is forwarded to designing department and designing the software from scratch took 10 full working days and 2 full time resource finish this activity in the specified time when the designing and architecture activity has done then development team start developing project from the scratch and 5 developer complete this job in the 40 working days and convert all the functional requirements in the working application as application was develop first time and appears as a working software in the world this software was not mature enough and it takes a long to test the application with in specified time actual time schedule for testing was 10 days but this activity took 20 days for testing and fixing bugs in software. In this way project slipped its schedule and marketed late due to unexpected bugs that appears in the software that was not supposed to appear in the software. Bug fixing activity that took some extra time than planned also

effect the budget of the software and due to hurry in marketing marketed project could not meet the quality attribute that was planned in the starting of the project. In case of second project for same domain that was developed using architecture centric approach and components were reused in it and tailored from the already develop and tested application. Some extra features requirements are gathered but most of the requirements of the existing systems are modified and tailored in the SRS document this modification activity of the requirement benefits in two areas one is less number of human resources worked on the requirement gathering and analysis and also it took less time than traditional approach. Requirement gathering and analysis took 10 working days and one expert is worked on this activity. In the design and architecting process phase took 2 days of one architect because we modify the design of the existing project. Most of the code was reuse of the existing products in the market of the same family so 5 developers work on coding activity and they finished this job in the 20 days because they use existing chunks of code and some extra coding required for connecting components. After the coding activity testing and bug fixing activity remain for 8 days and 2 quality experts work on that because they were experienced in testing application in the same domain and they were very much familiar with the software functionality and most of the modules were tested by them for similar projects so they took less time. If we compare both approaches we comes to the decisions that architecture centric approach was very effective approach in software product lines and this saves lot of money, time and deliver quality product. Quality attributes are addressed very well in the architecture centric approach because software developing team had already idea of quality attributes and these attributes are met very well.

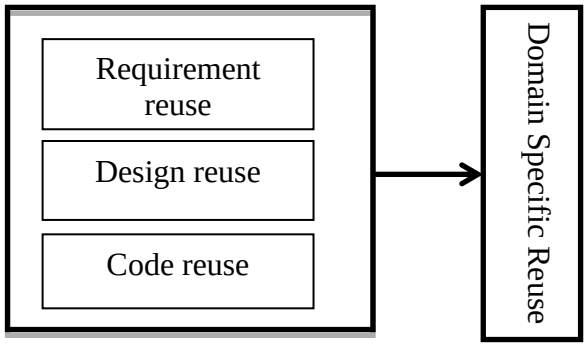

Fig 5: Component Reuse in Product lines

8. CONCLUSION

Architecture centric development is used for the software product lines. It is very smart approach for the component reuse in the software products where working in the same domain, component reuse in the every phase of software development life cycle accelerate the development of the software, it reduces the cost and time for developing the software in the same domain. It provides the acceptable level of quality because software in the same domain are very mature products and these software are working in live environment that's why these are more reliable than the software that are develop from scratch. Products using the software architecture centric approach can deploy early in the market than the products that are using traditional developing approach. Future work can be done in this domain for using architecture centric development approach for the projects other than the software project lines, process centric development and frame work centric development.

9. REFERENCES

[1] Luiz Fernando Capretz "COTS Based Software Product line development" International Journal of Web Information Systems, Volume 4, Number 2, pp. 165-180, Emerald Group Publishing, 2008

[2] Capretz L. F. (2005), "Y: a new component-based software life cycle model", Journal of Computer Science, Vol 1, No1, pp. 76-82.

[3] Lane, T.G.: Studying Software Architecture Through Design Spaces and Rules, Technical Report CMU/SEI-90-TR-18, Carnegie Mellon Univ., 1990

[4] Hofmeister, C., Nord, R. and Soni, D., Applied Software Architecture, Addison-Wesley,1999.

[5] Mary Shaw and Paul Clements. The golden age of software architecture. IEEE Softw., 23(2):31-39, (2006).

[6] Nary Subramanian and Lawrence Chung. "Relationship between the whole of software architecture and its parts: An NFR perspective. In SNPD-SAWN '05: Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Net-working and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05), pages 164{169, Washington, DC, USA, 2005. IEEE Computer Society

[7] Ramya Ravichandar, James D. Arthur, and Shawn A. Bohner. Capabilities engineering: Constructing change-tolerant systems. hicss, 0:278b, 2007.

[8] Lothar Baum "Architecture-Centric Software Development Based On Extended Design Spaces" University of Kaiserslautern

[9] Len Bass. Principles for designing software architecture to achieve quality attribute requirements. In SERA '06: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, page 2, Washington, DC, USA, 2006. IEEE Computer Society

[10] Troy S. Henry "Architecture-Centric Project Estimation" May 14, 2007 Blacksburg, Virginia

[11] D. Weyns, K. Schelfthout, and T. Holvoet. "Architectural design of a distributed application with autonomic quality requirements. In ICSE Workshop on design and evolution of autonomic application software", St. Louis,Missouri, New York, NY, USA, 2005. ACM Press.

[12] Cristena Gacek "Successful Product Lines Development in Small Organizations"

[13] Luiz Fernando Capretz "COTS Based Software Product line development" International Journal of Web Information Systems, Volume 4, Number 2, pp. 165-180, Emerald Group Publishing, 2008

[14] "Architecture Tradeoff of Analysis Method "Software Engineering Institute