I.J. Information Technology and Computer Science, **2015, 07, 57-65**
Published Online June 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.07.07

Management of Possible Roles for Distributed Software Projects Using Layer Architecture

Yumnam Subadani Devi, Laishram Prabhakar
Manipur Institute of Management Studies, Manipur University, Canchipur, Imphal 795003, Manipur, India
Email: y_subadani@yahoo.com, prabhakarmims@gmail.com

***Abstract*—** Several members are involved in development and management of the Distributed Software Projects. Each member needs to know the responsibilities of each other for proper management of the activities of such distributed projects to produce coherent outcomes. Distribution middleware software has higher-level distributed programming models whose reusable APIs (*application programming interface*) and components automate and extend native operating system capabilities. Software management tools like Work break-down structure (WBS), Gantt chart, Critical Path Method, and Critical Chain Method etc. does not fully help the managers to manage the member's responsibilities during the development of distributed applications. The layered architecture can help to do so. This style not only gives the layer level description of the activity involved, it also defines and directs the group of workforce. By listing the groups of workforce, the development team as well as the customer can know the activity and the member involved to work on those specific activities. This layered architecture is much benefited to development team and also to numbers of stakeholder of the large distributed project. The extended new approach of layer pattern with 'Responsibility Index' adds extra value to manage all the members' responsibilities. Managers, stakeholders and others can have an easy management system. The request or complaint from the customer can be passed to appropriate team without much delay. Most importantly this will give facility to collect timely feedback from all levels of customers.

***Index Terms*—** Layer Architecture, Responsibility Index, Team Management, Customer Service, Distributed Projects

I. Introduction

Most of today's software projects are geographically distributed with limited face-to-face interaction between participants. A typical software development life cycle has several phases. Some of the phases are requirement analysis, Designing, Implementation or coding, Testing, Deployment and Maintenance. Several people (development team) and stakeholders and customers work on different cycle or phase of a project. A good control is required to maintain over the life of the project by a team lead or a project manager. It could be easy to manage such a project if all the evolving people are resides in one place or within some define location. However it will be difficult to manage if many people work from different locations with different time zone and moving from place to place during a phase of project. Manager much uses appropriate tools to control, support and manage the entire team. Manager must have following knowledge

- to manage a distributed projects and its team
- to build teams across sites
- to break down the tasks and distribute
- to share knowledge as per time, space, and other differences, and
- to coordinate the work to produce best outcomes [1].

The distributed projects have difficulty of coordination. Management becomes more challenging and troublesome to make the 'teams' to work effectively and delivering better outcomes on time and within budget. To manage such distributed projects, the software design patterns are used. A design pattern is a template of formalized best practices to solve a problem. Different patterns are used for different activities of managing a software development. Layered architecture is one of common design pattern uses for allocating and managing the different responsibilities of a team during a life cycle of software development [2]. However the traditional Layered architecture does not have facility to update the status of project on sudden changes. Any new changes of the roles of any member or update of the status of the project should be available to entire team for proper management and common understanding.

This study proposed a new concept 'Responsibility Index' to incorporate with traditional layer architecture to manage all the responsibilities of a large project. This concept will help to manage all the roles or responsibilities of members involved in a large distributed project. The Index will be automatically updated when a role changes and team member will know role of each other. It will help the development team, stakeholder and customer to come closer. Customer will know their designated developer. This will minimize delays in handling ticket (complaints) by maintenance teams. It will improve relation with customer and stakeholder. It will also help in timely collection of feedbacks from the customers. This will help to bring all involving members closer reducing the communication gap.

A. The concept of software development tools

There are different phases in life cycle of a software project starting from requirement gathering to deployment and maintenance of the project. The role and responsibilities are different for executing different phases of the life cycle of software. It may not easy to manage all the roles and responsibilities for a distributed project. Software management tools like Work break-

down structure(WBS), Gantt Charts, Critical Path Method, Critical Chain Method, Program Evaluation and Review Technique (PERT), Network diagrams like Activity on node (AON), Activity on Arrow (AOA) do not exactly fully help the manager to manage the roles and responsibilities. Mostly these tools are developed to manage the centralized projects [3]. The amount of work and members involved in developing open system software applications like Linux, Open CV, etc are very large and managing the responsibilities of each of the member are not easy. Again during maintenance, retrospective phase, sorting reusable components, feedback, emergency of such projects, finding the target group is also a time consuming activity. The new proposed layer software architecture will be benefited to all the personnel involved in the large distributed software projects even to the target user or the customers. Customer will come to know whom to contact or finds the target group in difficulties.

High-level patterns or style are referred as the architectural styles. It includes patterns such as client and server, layered architecture, component based architecture, message bus architecture, and service-oriented architecture (SOA). Each style describes different aspects of applications. For example, some deployments patterns are represented by architectural styles describe, some describe structure and design issues etc. Several applications usually use a combination of more than one of the styles. The architectural style is a set of principles and instruction pattern that provides an abstract framework for developing systems. It improves partitioning and helps design of reuse by providing solutions to recurring problems. They provide a common language. This facilitates a higher level of understanding that is inclusive of patterns, rules and regulations, without getting into details. Architecture styles can relate to client and server versus n-tier. Architectural styles can be organized by the different key focus area [4].

B. The Software usable concept

The purpose of reuse of the Software is to improve software quality and productivity. The concept of 'software reuse' is to build systems that are bigger and even more complex, more reliable, less expensive and derivable on time. Most software systems are not totally newly develops; instead they are variants of systems that have already been developed. Institutions build software projects within a few business requirements, repeatedly building system variants within those domains. This is to improve the quality and productivity of the software production process. Each team of a distributed project has a responsibility to contribute reusable assets to team and, therefore, asset development, and support responsibilities are distributed among team of projects.

C. The concept of layered architectural style

Layered architecture aims for grouping of related functionality within an application into distinct layers-related by a possible common role or responsibility. A software component can be a software module, a service, a resource or a package that encapsulates a set of related functions. Communication between layers may be explicit and loosely coupled. This style has been described as an inverted pyramid of reuse. In this each layer aggregates the responsibilities and abstractions of the layer directly beneath it. In the situation of strict layering, components in a layer can interact with components in the same layer from the layer directly below it. The relaxed layering situation allows tasks or components in a layer to interact with components in the same layer or with components in any lower layer. Some layers of an application may reside on the same physical computer of the same tier or may be distributed over separate computers of n-tier, and the tasks or components in each layer communicate with components in other layers through some defined interfaces. For example, an application design consists of a presentation layer to represent functionality, business layer to represent business rules, and data layer to represent the data access. Each layer in this example has unique contributions.

For a successful management of project a number of key behavioral factors have emerged. There are several factors that are crucial in impacting on the behavioral issues relating to project management [5]. The two main factors are Project Teams and Virtual Teams. The better future of the organizations will rely on project teams and project management that utilized the distributed teams comprising individuals who may directly or indirectly interact with each other as part of virtual team. Another factor is the conflict and negotiation management where all the conflicts are going to minimize. The key actions and behaviors across the interaction process lead to the learning that incorporates the development and improvement of new skills, to new knowledge to satisfy team task performance, to being flexible and dynamic, and to improving the quality of working life. In addition to this, coordination assists in meeting goal settings, managing and integrating people and information, setting and managing time schedules and planning and managing divisions. Communication as an additional behavior enables people to understand the nature of a problem coherently and to share synchronously or asynchronously. Furthermore, the decision-making process as an action involves both intellectual and judgmental tasks influencing team outcomes [6].

The detail of the study is described in several sections. The related works in given in section 2, the concept of a layered architecture in section3. Section 4 has the new proposed layer architecture with ‘Responsibility Index.' The demonstration of the proposed new layer framework is given in section 5 and section 6 has the conclusion of the study.

II. RELATED WORKS

The development of distributing software development can continue to increase time-to-market by around-the-clock. It is to increase flexibility on merger and acquisition of different opportunities. Such geographical distribution of task becomes increase with high transfer of development and maintenance activities from the

developed countries to developing one or vice versa [7]. There is lack of high skilled and engineering expertise and trained personnel force for off-shoring innovation decisions [8].

Such distributed environment, the coordination and communication works are difficult. Management of projects becomes more challenging and troublesome. Distributed software development has several complexities and challenges. Making distributed project teams' work effectively and delivering better outcomes on time and within budget are serious challenge of industries. In order answer to these challenges, experts, and researchers or practitioners are continuously finding and developing vast amounts of frameworks, guidelines, tools, methodologies and tips [9, 10]. Performance analysis of any organization can be done with respect to the factors such as productivity, innovativeness etc[11].

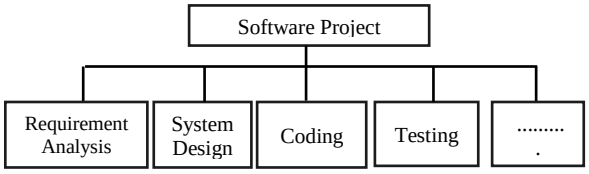
A. Impossible management over software crisis

Software development would be slow, expensive and error prone, often resulting in products with large number of defects which cause serious problems in reliability, usability, and performance. As per the Chaos report of Standish group [12], the companies in the United States spent more than $250 billion each year on IT application development of approximately 175, 000 projects. Around 16% of these projects finished on schedule and also within target budget. Around 31%were cancelled before completion giving losses of about $81 billion. Approximately 53% exceeded their original budgets by an average of 189% for losses of about $59billion. Those projects that managed to final completion delivered an average of 42% of the planned features. Another report [13] from the Software Engineering Institute (SEI) indicated that out of around 542 software organizations participating in the CMM maturity assessment, 67% of them were at CMM Level 1, and 20% were at CMM (Capability Maturity Model) Level 2. Inputs to the process are ill-defined, and the transition from inputs to final software products is uncontrolled.

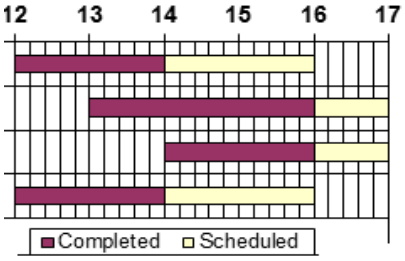
However, there is still no visibility as to how software products are produced, and any disturbance to the development team or resources can easily cause project failure. It is said that 87% of the software organizations in the survey were unable to control their development processes. The development process is so reactive that management control is impossible. Rick management studies are also performed based on deliverable of projects. Such study can report the risk associated to a deliverable to alerts the developer team to minimize the risk using preventive measures [14].

B. Some related tools

Work break-down structure (WBS) provides the basis for planning and managing the project schedule and breaks the work required into smaller and more manageable pieces. Itisan outcome oriented analysis of the work involved in the project. It also assigns duration to each problem or task or outcome identified. Gantt Charts gives a graphical representation that shows the project calendar giving the start scheduled and end dates, and the durations. It also records the people who are responsible for each activity as optional purpose. It is difficult to update manually. Critical Path Methods schedule all project activities so that it can be completed quickly in identifying those activities that, if delayed, are likely to affect the overall project completion time. Critical Chain Method mainly used in predictive planning, but can also be used for iterative development. The program Evaluation Review Technique (PERT) takes into account of handling the uncertainty in project planning and scheduling [15]. Sample diagrams of a work break down structure and Gantt chart are shown in fig1(a) and (b) respectively.



(a) A Work Break down Structure



(b) A Gantt Chart (schedule in weeks)

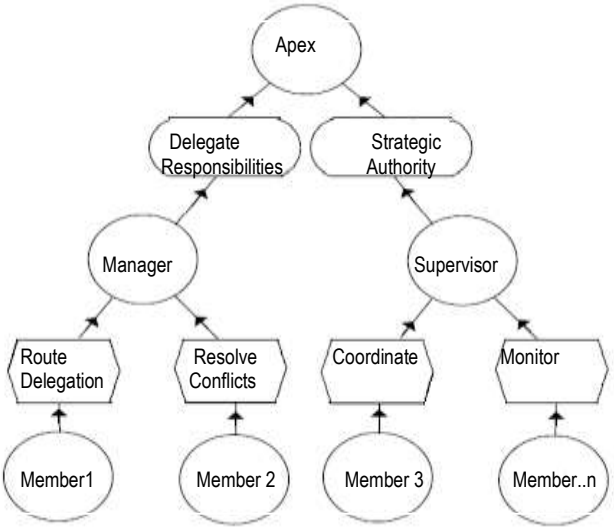
Fig. 1. Project Management charts

C. Some architectures for proper management of responsibilities

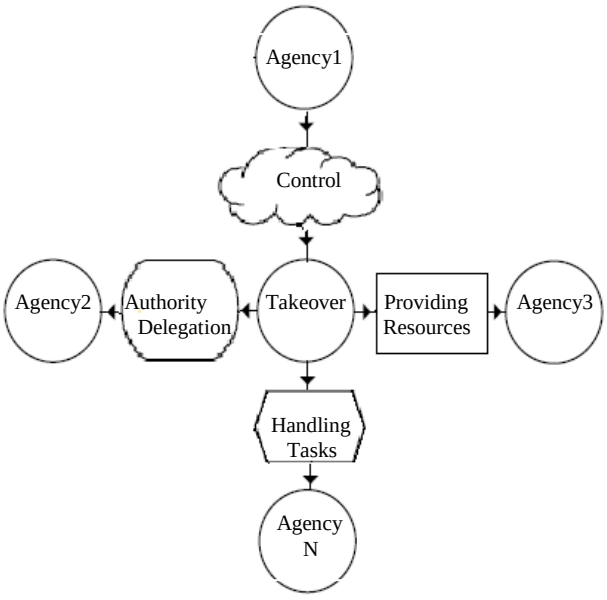
Software architectural design has treated as a crucial phase of the design process. System architecture constitutes manageable model of system structure that describes how system components work together. Several works have identified architectural styles to guide high-level system design. However management of the responsibilities of the members of the project is not discussed. For example the flat style, a Software architectural design has no fixed structure and no control of one actor over another is assumed. Another architecture design, the structure-in-5 style consists of the typical strategic planning and logistic components generally found in many organizations. At the top of the most organization lies the apex composed of strategic executive actors. Below it resides the logistics, control or other standardization and management components respectively support coordination and middle agency.

The pyramid architecture style is the hierarchical authority structure exercised within organization. Members at the lower levels depend on members of the higher levels. The mechanism is direct supervision from the apex body. Most of the managers and supervisors are only intermediate actors routing strategic decisions and authority from the apex routing to the operating level. They can coordinate by their own at a local level. Such

style can be applied when deploying simple systems. Moreover, it encourages dynamicity since coordination and decision mechanisms are happened directly, not complex and immediately identifiable. It is not suitable for huge system requiring many kinds of agents. However, it can be used to manage and resolve crisis situations. A pictorial flow of pyramid style is shown in fig2.a.



(a) A pyramid (hierarchical authority style)



(b) A takeover style

Fig. 2. Architectural styles

The takeover architecture style involves the total delegation of authority and management from multi partners (say agency 1,2etc) to a single collective takeover member or actor. This style identities and autonomies of the separate units that no direct relationships, dependencies or communications are tolerated except those involving the takeover. Different task has been assign to several agencies. It involves agreement between more agencies to obtain the large benefits, little investment and lower maintenance costs. Each partner or agency can manage and control itself on a local dimension and interact directly with other agency or partners to exchange, provide and receive services, data and knowledge. A simple pictorial flow of takeover style is shown in fig2.b [16].

Hybrid forms of integration of agile practices are discussed to develop quality software as per customer requirements [17]. Neural Networks were used to analyses the performance of Software Effort Estimation Models [18]. However in most of such study, the proper framework for managing the responsibilities involved in distributed software projects are not carried out.

III. The Concept of Layer Architecture

Architectural Styles are the high-level set of rules that constrains the architecture for certain context and describe to describe solutions to the problems. Architecture is considered to consist of several components and the connectors (interactions) between them. A software component can be a software module, a service, or a resource that encapsulates a set of related functions. Design explicitly addresses functional requirements while architecture explicitly addresses functional and non-functional requirements. Some of the requirements are reusability, maintainability, testability, efficiency, portability, interoperability and fault-tolerance and the other Quality Attributes. The goal of software architecture is to enable the construction of very large system architectures [19]. Some of the benefits of Architectural Style are:

1. Promotes communications among the designers and developers by using similar pattern names for representing the lengthy description of projects
2. Designers and developers use similar terminologies as they provide a set of predefined subsystems
3. Supports reusability and responsiveness
4. Improves development efficiency and productivity among the team
5. Standardized layer interfaces for common libraries

The Layers style helps developers to structure the applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction. Networking protocols are the common known example of layered architectures. This protocol consists of a set of principles and conventions that describe working process. Large system requires decomposition. A large system may be decomposed to small portions for proper management. The following is an example of designing a system that has mix of low and high-level issues. Such systems require some horizontal structuring. This is the case where several operations are on the same level of abstraction but could be independent of each other. Example of above system is OSI 7-layer model. Parts of the system should be exchangeable. Components should able to replace without affecting the rest of the system. Its platform may be subject to change in the future. Option of code changes, recompilation, and reconfiguration of the system can also be perform.

Adjusting cache or buffer sizes are examples of a change of such system.

The layers pattern is better suited to systems that require reliable operation, because it is easier to implement error handling. Structurally, each layer provides a related set of services. Dynamically, each layer may only use the layers below it. Layer A may use layer B because it depends on something B does for example data written to the database by B to be used by A. Layer A evokes layer B says Layer A passes control or data or both directly to B.

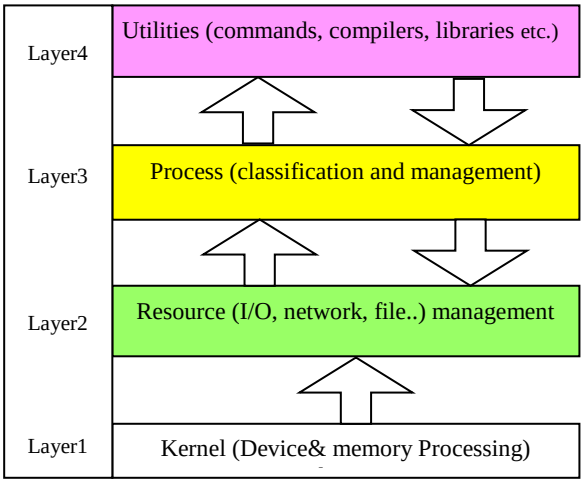


Fig. 3. Sample Layer styles (Operating system)

IV. The New Proposed Architecture

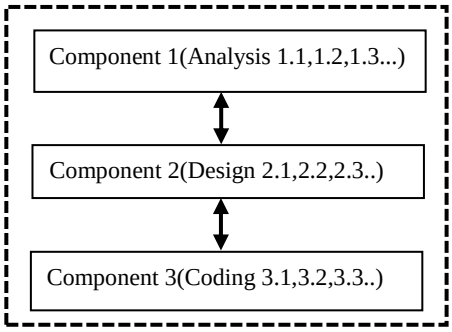
A large project has several components to perform. A component can be a module or activity of a project, a service, or a resource that encapsulates a set of related functions. Such project has numbers of team members who worked on different components (modules) with different responsibilities such as analysis, designing, coding, testing etc. In this example the component1 has all the modules and related activities of the analysis phase of development of a large project. Component2 has all the modules and related activities of the design phase and so on.

This study proposed a new concept of layered architecture which can handle the index of responsibilities of a large project. The index will be automatically updated on changed of the roles of a member. The new status will be update among all evolving members. This new framework has been evaluated using small case study.

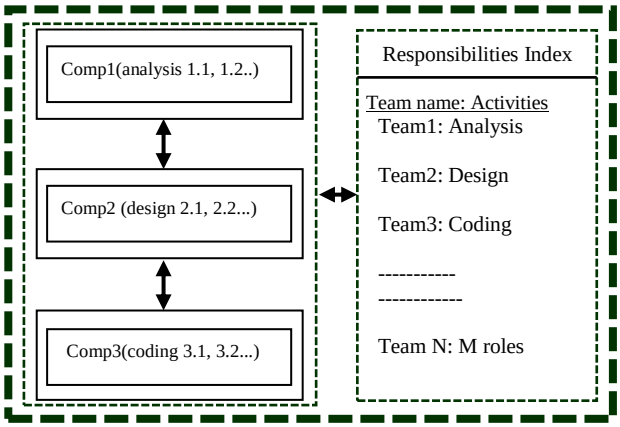
An example of project with several components is shown in fig.4(a). Each layer consists of three components (comp) in different levels and it also has several sub layers. The components are the modules and sub modules of a large project. The components are interrelated with the help of request. In the middle layer two components are interacted. Components in different layers call each other directly and shield each layer by incorporating a unified interface. In the design, component 3.1 no longer calls component1.2 directly, but calls a Layer 1 interface object that forwards the request instead. It is probably that a client issues a request to Layer N. Since Layer N cannot carry out the request on its own. It calls the next Layer N-1 for supporting subtasks. Layer N-1 provides these services. In the process sending further requests to Layer N-2 and soon until Layer1 is reached. Here, the lowest-level services are finally performed. If necessary, it replies to the different requests passed back up from Layer 1 to 2, from Layer2 to Layer3, and so on until the final reply arrives at Layer N.

The concept of 'Responsibilities Index' is introduced along with the layered architecture. The sketch of the enhanced version of the Layer Architecture is given in fig. 4(b). This Index is used to track the responsibilities of each member who are evolved in each of activity of the entire project phases. A development team may have several team groups such as team1, 2, 3 etc. Assume that project has several phases such as Analysis, Designing, and Coding etc. Different teams were assigned for different works.

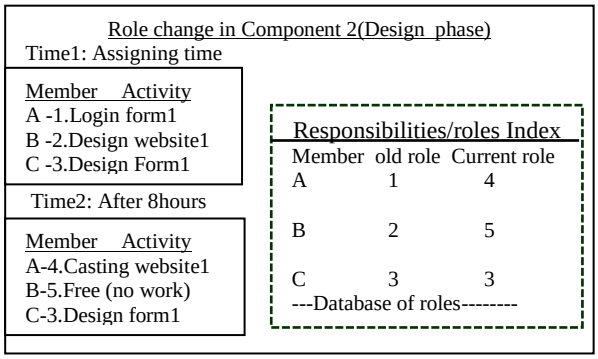
Let's assume that team2 is responsible for designing a component of the project. Team2 has three members who work on different roles. A sample of the scenario is shown in fig.4.(c). Initially in the time of assignment of the role(time1), the member 'A' has works of designing 'Login form1' which index1 as per this scenario. Likewise member 'B' has work of 'designing website1', 'C' has 'designs of form1' respectively.



(a) General Layer Architecture



(b) Enhanced version of the General Layer Architecture



(c) Roles changes in Layer Architecture and R Index values

Fig. 4. New layer Architecture

After 8 hours(Time2) (say a day), the role of 'A' is change to 'Casting the website1' which was finish product from other member. The Index value of 'A' is changed to 4 from previous value1. Likewise the role of 'B' change to 'free', as no work was assigned, 'C' continues with the 'design of form1'. The 'Responsibility index' was automatically updated wherever there comes any changes in roles or component of work. From this current index value, the evolving team members will come to know about the component handled or handing by the team members. If the type of the project is customer service (frequent interaction with customers on a business), the index value can be share with the customer and customer can interact with the corresponding responsible person in times of any issues. In the current customer service system the ticket raised by the customers are received by a central team and distributed to responsible member. Sometimes the ticket goes around and services are delayed, annoying the customers. Management of activity will be much earlier with help of database which contains the roles.

By adapting this Index, any changes in the responsibilities will be automatically updated. This option is not be available in traditional layered architecture. Such activity of managing the activities of projects with respect to the members' responsibilities can be implemented as per the requirement of the different projects. Management of the distributed projects has several issues namely communication, coordination, and control as consistently used in previous research and in practice. The details of these are given below. The possible problem of communication, coordination and control can be minimized by implementing such of new form of layered architecture.

A. Communication

In this process people convey meaning to one another via some medium through which they exchange messages and information in order to carry out project activities. Distributed team members can find it difficult to deal with different interaction styles and preferences. In some situation, they sometimes make regular and negative attributions based on infrequent communication and perceptions of unresponsiveness.

B. Coordination

It can be taken as a mechanism through which people and technical resources are combined to carry out specified activities in order to accomplish stated goals. It requires action related to list of task, team member roles, member relations, time etc.

C. Control

During control it does monitoring and measuring project activities. It is done so to anticipate and manage variances from project plans and organizational goals. The details of these factors are well discussed in the previous of study [20].

The execution of the proposed framework with respect to development of large project is demonstrated in next section 5. This new framework makes the easy management of team members, developer and users. In case of any inquiry, the users can directly transfer their query directly to concern groups of developer, without confusion and reducing waiting time.

V. Demonstration of the Proposed New Layer Architecture

Let's consider a scenario to develop a project name as 'Computerized Medical Clinic'. The project is going to develop in distributed environment; meaning the developer team and customer is scatter in different parts of globe. The sketches in fig5. represents the requirements gathered along with some of the main required components of the system. This shows the shows the arrangement of the component before applying the new scheme of Layer architecture. From this figure we see only the components of the possible software. But it does not shows that which components are going to developed together or what are sequences of the components.

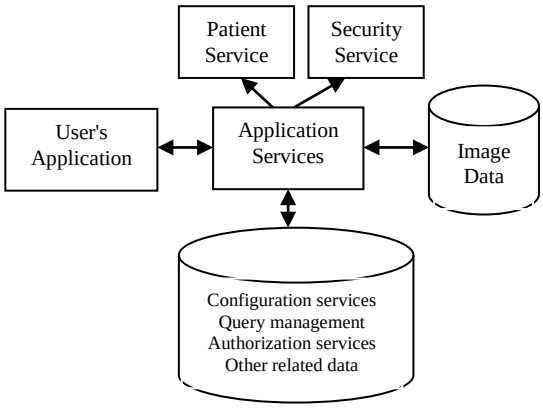


Fig 5. A medical Imaging Systems (before applying Layer architecture)

It also does not show which group of the developer will work on which component. Some of the sequences of the workflow are unknown to participating groups. It may require several tools such as WBS, Gantt chart etc to show different aspect of the project. This tools needs to be update now and then to show the progress of the work and changes in roles of evolving members. The layer architecture can be applied to resolve key distributed

system design challenges as discussed in earlier sections. It will help to resolve the following common design challenges like separating concerns between tiers, improving performance, enabling client extensibility, ensuring platform-neutral, network-transparent communication, decoupling suppliers and consumers, locating and creating components scalable, minimizing resource utilization etc.

As the members of the different teams are staying far away from each other across the globe, managing the responsibilities of the team members becomes complex. Defining the levels of control is necessary at appropriate level along with responsibilities of the team members involved in developing this project. Adapting layer architecture can minimized such problem in distributed environment. The new concept of 'Responsibility Index' will help each involving members to keep updating new changes in any role. In the scenario of 'Medical Imaging Systems', after applying the architecture layer pattern, roughly system can have three layers –Database Layer, Middle Layer and Presentation layers. The work flows are process among these three layers to Analysis, Design and Implementation etc. With the concept of Responsibility Index, the managing the responsibility in a layered architecture becomes easy. The theory of this Index is also shown in fig4(c).

The Index stores two current main data: (1) team name and (2) activities (component) associated with a team. Team 'A' will work on activities associated with 'Layer1'. Team B works on 'layer2 and so on. Table1 shows the sample data that a Responsibility Index can hold. Here team 'A' has three members Mr.X,Mr.Y and Mr.Z and all are assigned the work of designing the 'Database Tier'. Index value already define, example 0-no assignment, 1-design, 2-populate table, 3-refresh etc. Initially Mr.X of team A has no work and it index value is 0. At 09hours Mr.X was assigned to the designing 'table3', and at 09.30hours Mr.X to populate table1 and then refresh table1,2 and 3. Index values show the changing roles of Mr.X(A) with respect to time. Anyone access this Index will come to know about the roles and works executed by Mr.X(A). The remaining other layer2 and 3 also can be executed in similar fashion adapting the Index values.

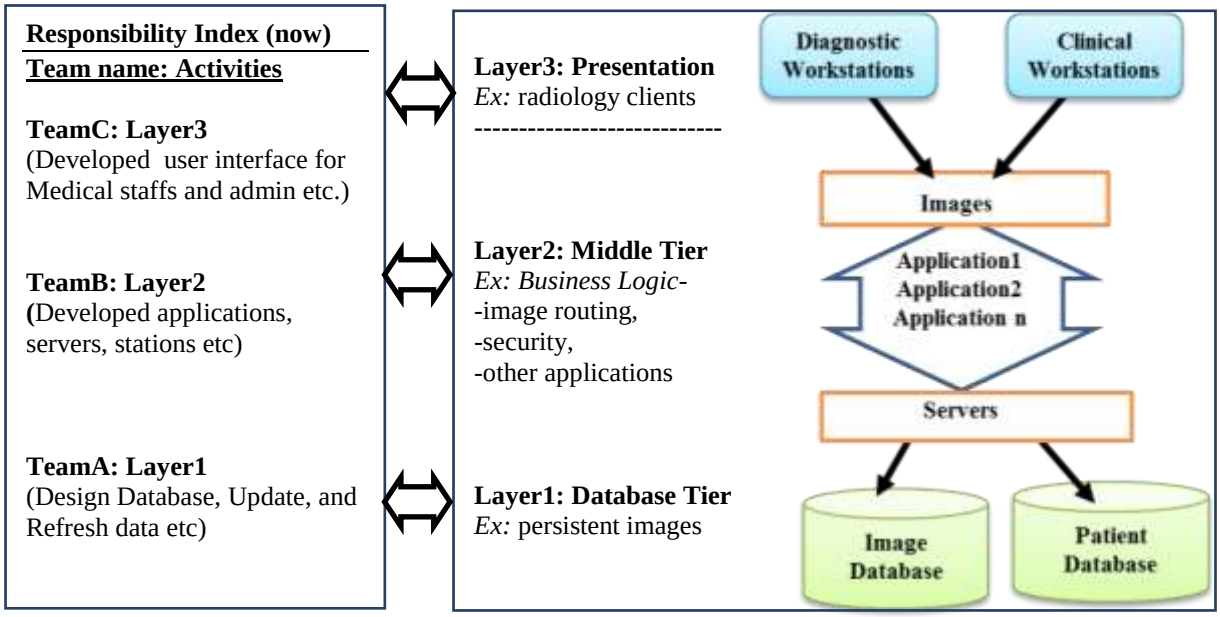


Fig. 6. Electronic Medical Imaging Systems (After applying Layer architecture and Responsibility Index)

Table 1. Sample contains of Responsibility Index

Team Name: TeamA, Component: Layer1(Design database Tier)

Team Name	Roles	Subcomponent	Current Index value	
			Time	Value
Mr. X(A)	--	--	08.00	0
Mr. Y(A)	Design	Database table1	09.00	1
Mr. Z(A)	Design	Database table2	08.00	1
Mr. X(A)	Design	Database table3	09.00	1
Mr. X(A)	Populate	Database table1	09.30	2
Mr. X(A)	Refresh	Table1,2,3	10.00	3
Mr. Y(A)	--	--	10.00	0
Mr. Z(A)	--	--	10.00	0

This Index records the current roles and assigned task of each members. The index automatically updates any changes in responsibility of any team member. In such way the team manager, a team member knows the latest responsibilities each of them. During the time of new assignment, transfer of responsibility this Index helps to bring transparency among the team. At the same time, it also helps the stakeholder and customer of the project to interact, inquire or discuss with desired and correct member of project development. Otherwise, sometimes the tickets (complains) rises from the customers are unattended and delayed in services. Fig5 shows the structure of the simple project. It does not answers question such as which team works on which component of project, which team handle what type of responsibilities etc. Fig6 shows that layered architecture along with the level of hierarchies of the project. These layers will help the developers to show the flow of data-inputs to outputs and detail phases of the project. This new layered architecture with the concept of Index value will help the evolving team to execute the project smoothly from time of inception to deployment.

VI. Conclusion

The new proposed 'Responsibility Index' incorporating with traditional layer architecture added an extra value to manage all the responsibilities of a large project. This new Index has the capacity to manage all the roles and responsibilities of members involved in developing a large distributed large project. Even though the projects are conducted jointly from various parts of globe, several roles changes happen, the Index will be automatically updated and member will know each role of each member. Hence it helps the development team, stockholder and customer to come closer. Customer will know their designated developer, in case of rising ticket (complains) and again delay between rise of ticket and received by maintenance teams will be reduced. There is a need for such type of combined development framework that can manage development activities and member's responsibilities improving the relation with customer and stakeholder. Most important is timely collection of feedbacks from the customers. This is more important if the type of project is of distributed in nature which has very frequent update and changes of application types, roles, customers and stakeholder etc. This helps to bring all involving members closer reducing the communication gap.

References

[1] JD Herbsleb, D Moitra, D, "Global Software Development," IEEE Software, 2001, 18(2): 16–20.
[2] RC Martin."Design Principles and Design Patterns." Retrieved 2000.
[3] RS Pressman, WS Jawadeka. "Software engineering." New York 1992.
[4] M Shaw, D Garlan, "Software architecture: perspectives on an emerging discipline" PH, 1996.
[5] P Morris, JK Pinto (Eds.) "The Wiley Guide to Project Program and Portfolio Management", 336 pp. The Wiley Guides to the Management of Projects.Wiley, Hoboken NJ , 2008.
[6] G.Kapogiannis, "a conceptual framework for project managers to improve projects performance", Ph.D. Thesis, University of Salford, Salford, UK,2013
[7] B. Meyer, "The unspoken revolution in software engineering," IEEE Computer,2006, 39(1): 121-123.
[8] AY Lewin, S Massini, C Peeters "Why are companies offshoring innovation?The emerging global race for talent," J. of Int. Business Studies, 2008.
[9] J.Espinosa,DeLone, W., and Lee, G. "Global boundaries, task processes and IS project success: a field study," Information Technology and People, 2006: 345-370
[10] Y.Subadani Devi, Y. Jayanta Singh, Laishram Prabhakar. "Grooming a New Team with Potential Roles using the Scrum Practices."Proc. Int. Multi Conf. of Engineers & Computer Scientists. Hong Kong, 2012.
[11] ANH Zaied. "An integrated knowledge management capabilities framework for assessing organizational performance."Int. Journal of Information Technology and Computer Science(IJITCS)4.2(2012): 1.
[12] Standish. The Chaos Report. www.standishgroup.com /sample research/PDFpages/-chaos1994.pdf,1994.
[13] W.C.Peterson "SEI's software process program-presentation to the board of visitors", Software Engineering Institute, Carnegie Mellon University.1997
[14] Y. Subadani Devi and L. Prabhakar. "A Preventive Risk Analysis for Managing Distributed Software Projects based on Deliverable." International Journal of Computer Applications 68.1 (2013): 27-31.
[15] DG Malcolm, JH Roseboom, CE Clark, "Application of a Technique for Research and Development Program Evaluation OPERATIONS RESEARCH",7(5): 646–669
[16] M. Shaw and D. Garlan. "Software Architecture: Perspectives on an Emerging Discipline", Upper Saddle River, N.J., Prentice Hall, 1996.
[17] Z Mushtaq, MRJ Qureshi. "Novel Hybrid Model: Integrating Scrum and XP." International Journal of Information Technology and Computer Science (IJITCS) 4.6 (2012): 39.
[18] E Praynlin, P Latha. "Performance analysis of software effort estimation models using neural networks." Int. Journal of Information Technology and Computer Science (IJITCS) 5.9 (2013): 101.
[19] M. Shaw, P. Clements. "A Field Boxology: Preliminary Classification of Architectural Styles for Software Systems", 1996.
[20] J.Goodbody, "Critical success factors for global virtual teams" Strategic Communication Management, 2005,9(2):18-21.

Authors' Profiles

Yumnam Subadani Devi is Ph.D research Scholar in Manipur Institute of Management Studies (MIMS), Manipur University,India. She holds MBA specialization in Information Technology from Sikkim Manipal University. Her interest areas are Software Engineering, Management Information Systems and Database Management System.

Laishram Prabhakar, is an Assistant Professor in Manipur Institute of Management Studies (MIMS), Manipur University,

India. He holds a Ph.D degree(Information System Management. Presently working as Coordinator of Community College, Manipur University. He has over two decades of experience in the field of Information System Management. His area of interest is in the area of developing strategies for innovation and knowledge management. He provides concepts of knowledge management in Business and Technology.

How to cite this paper: Yumnam Subadani Devi, Laishram Prabhakar,"Management of Possible Roles for Distributed Software Projects Using Layer Architecture", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.7, pp.57-65, 2015. DOI: 10.5815/ijitcs.2015.07.07