Available online www.jocpr.com

Journal of Chemical and Pharmaceutical Research, 2014, 6(5):2003-2007

Research Article

ISSN : 0975-7384
CODEN(USA) : JCPRC5

The analysis and design based on fusion process model

Ruihui Mu

College of Computer and Information Engineering, Xinxiang University, China

---

ABSTRACT

A software process model fusion process model to improve the software development process. The fusion process model has five basic phases and a fusion process control and coordination of the whole development process. The fusion process model using the 3C model to summarize each stage of the problem solving process. 3C model, which helps to realize the developing method based on component, provides the control software development process of solid. The method of component driven by cost, risk and time related is finite element and ensure the overall quality of software system, reduce the development cost and time to consider changes in customer demand, risk assessment, identification, component development in every stage of the process of evaluation and related issues. We have implemented the design fusion process model of information system of the real world and evaluation of the implementation of the project to estimate the initial.

Keywords: Fusion Process Model, Process Model, Component Driven Approach

---

INTRODUCTION

A wide array of process models for organizing the process of software development has emerged over the last few decades. These process models represent patterns for successful development under different conditions. In current approaches, process control is performed on overall software process by decomposing the overall engineering process into phases. While decomposing overall engineering process into phases for effective and reliable development of software product, it is not sufficient. To handle cost, time, quality of software product and changing requirement of client, we need to control the internal process of each phase. Providing a solution for a given problem is not simple, it involves the accumulation and use of huge amount of knowledge. The solution space analysis approach is still not integrated into software process models. It aims to identify the right solution domains for the given problems and extract the relevant knowledge from these domains to come up with a feasible solution. To provide quality software, it is necessary to identify the important knowledge sources for a given problem[1]. Not all the solutions identified for a given problem are desirable. In the alternative management process, different alternative solutions are searched and evaluated against explicit quality criteria [2,3]. The high risk in software development led to the inclusion of managerial, financial and psychological factors in models [4,5], and [6,7]. Shaw and Garlan [8] identify seven levels of design specification capability which supports the concept of components, composition, validation, alternatives and finally automation. In the component based development, cost, time and reliability risk for an organization developing software system shrink to component level that can be managed effectively at any stage. The goal of the fusion process model is to address all the concerns and consider each phase of software development as the software development process and provide an effective model for software development phases, which will reduce risk associated with cost and time.

2. Case Study of Fusion Process Model

The case study aimed at investigating the practical aspect of fusion process model and implementation in commercial Software Company. The project objective was to build a complex rights management system that can be used to provide services to all industry types. This was highly generic system that should allow an administrator to configure a system that is tailored for any rights marketplace. The rights management market was highly complex,

with convoluted value chains with a wide range of variables affecting each potential assignment of rights or transaction e.g. a movie might have rights for cinema, DVD and online release in one hundred territories in seventeen languages, each at a different date and price, and each with a different liability in terms of the sums payable to distributors, producers, musicians, actors and producers. Often rights are locked out and a key feature of the system will be in finding what rights are available for what media in what territories so as to maximize the revenue potential for the right. The Figure 1 describes the relationship between various entities.

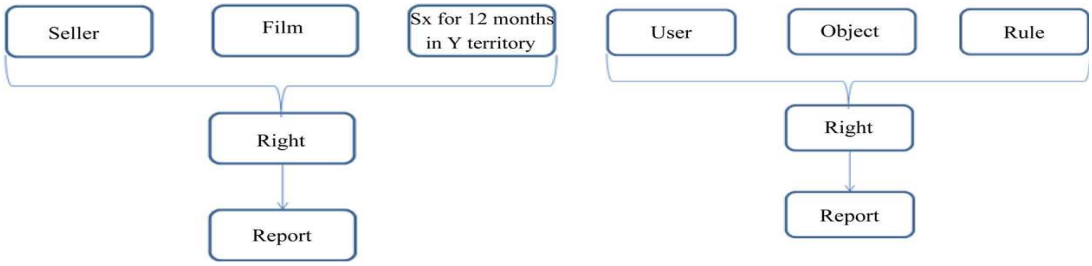

Figure 1. Software entity relationship diagram.

2.1. First Phase (Project Preparation) of Fusion Process Model

It is essential to first understand the requirements of the client. The two senior company employees were involved with client. Meetings were conducted with the client by the project manager and team leader to identify the requirements. Later these requirements were discussed with developers of the team. Based on the client inputs, the team prepared a "understanding document". As a part of fusion process model developers were asked to raise questions based on the control part of the development process and a few important points come into light i.e. single/multiple client handling, online access, internationalization, separate logging mechanism. These queries were further discussed with client for refinement of the requirements and then again discussed with developers of the team. While pursuing phase model, team searched the solution domain knowledge to find out if any similar kind of project handled by the company previously based on the client inputs. Subsequently, found that the concept of this project is similar to the ERP project handled by the company, which has functionality already implemented for rights management.

Then team defined priority for various components like database design, complete UI design as required by customer, logs framework, internationalization support, user functionality, reporting functionality, application sign-in/sign-out, administration functionality design, object functionality, rules functionality, user/object/rules mapping, reporting functionality for other entities like rules/objects. Further decomposition was done on the above mentioned components and assigned to team members, depending upon the level of coupling team can parallel start working on different components.

2.2. Second Phase (Software Blueprint)

Depending upon the inputs from the phase one, various design decision was taken after considering various alternatives based on the evaluation of control part in phase model. Team performed software design documentation, including Architecture design, Database design doc and Sequence diagram to understand initial flow.

Software design divided into various layers:

1) UI Layer;2) Business Logic Layer;3) Database access.

Further decomposing the three layers and prepared the documents for each low-level design requirement. The low-level requirements of each component were documented describing the technical implementation details including time frame and constraints (if any). Architecture document, which was the detail design document for each different component of project, was prepared. Then the priority for each component was defined and five reusable components were identified, as part of domain engineering in phase model.

1. Identify and Prioritize the Solution Domains. For the overall problem and each sub-problem, the search was executed for the solution domains that provided the solution abstractions to solve the technical problem.

2. Identify and Prioritize Knowledge Sources. Each identified solution domain covered a wide range of solution domain knowledge sources. These knowledge sources were not all suitable and vary in quality. For distinguishing and validating the solution domain knowledge sources team basically consider the quality factors of objectivity and relevancy. The objectivity quality factor referred to the solution domain knowledge sources itself and defines the general acceptance of the knowledge source. The relevancy factor referred to the relevancy of the solution domain knowledge for solving the identified technical problem.

3. Extract Solution Domain Concepts from Solution Domain Knowledge. Once the solution domains was identified and prioritized, the knowledge acquisition from the solution domain sources was initiated. Due to the large size of

the solution domain knowledge, the knowledge acquisition process was a labor-intensive activity, so a systematic approach for knowledge acquisition was practiced.
4. Structure the Solution Domain Concept. The identified solution domains concepts were structured using parent-child relationship. Here all the attributes and operations associated with the concept were defined.
5. Refinement of Solution Domain Concepts. After identifying the top-level conceptual architecture, the focus was on each sub-problem and followed the same process. The refinement was necessary as the architectural concepts had a complex structure themselves and this structure was of importance for the eventual system. The ordering of the refinement process was determined by the ordering of the problems with respect to their previously determined priorities. Architectural concepts that represented problems with higher priorities were handled first and in the similar manner the refinement of the architectural concepts was done.

2.3. Third Phase (Software Realization)

Development started once client approved the design documents and development of components started based on the priority of component. Initially teams started working on two independent components UI design and database creation. As the client was extensively involved during the design phase, each small level detail was incorporated in UI design and database design after lot of modification before approval. Now, the team had clear vision for development. These components were immediately approved by client after completion. Now the base was ready to build a complete software system on it. The team started work on User entity and authorization part. The client was involved in the development also and a few minor modifications suggested by the client, which were immediately applied. After the development team evaluated these components and found that if they would have implement log framework and internationalization during development rather than considering it as an extra activity, they could have saved a lot of time in development and testing. All the customer suggestion and evaluation results were noted down for predecessor components. The team moved the component immediately to testing phase after completion of development. The testing results of each component were used as guidelines for the development and evaluation of other components.

Now we will summarize the software realization phase technically, the development team has a development project divided into various components, the development of components start based on the priority of component. Each component follows a different line of development and shared with the client to get the client feedback (requirement change/new requirement), before each cycle completes the development process. Each component is monitored using various control techniques defined by the development team to keep track of quality, cost and time. Once the development of any component complete, it immediately moves to testing phase.

2.4. Fourth Phase (Testing)

The complete software system design was based on component driven development approach. Each component directly moved to testing phase after the completion of development phase. There were various similar kinds of components in application; the test case used for one component was used with little or no modification for other components, which saved a lot of time required to build test cases. The test cases for reusable components were already available with the company, which were used to test various scenarios of application. Finally the integration testing was completed to deliver the complete software solution. Figure 2 shows the high level testing analysis.

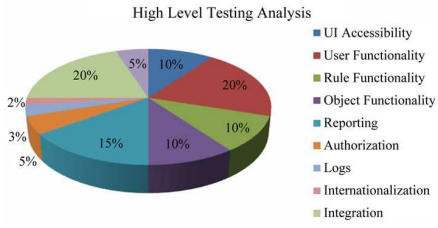

Figure 2. High level testing analysis

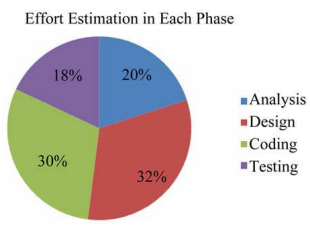

Figure 3. Effort estimation in each phase

2.5. Go Live and Support

The purpose of the Go Live and Support phase was to cut over to live productive operation and to continuously support and improve live operations based on project agreements finalized with client. Finally the software deployed on customer landscape within projected cost and time. Due to component driven approach and customer involvement in each component, the software solution had all the required functionality.

3. Results

Based on the judgments of the project manager and team leader on their individual experience, results where concluded. A lot of time spent on planning and design, as shown in Figure 3. But the time spent during first two phases help the development team to fully understand the requirements—problems/sub-problems till final level of

decomposition. This decomposition helps in the final delivery of the product functionality and monitors the development process of each component separately at unit level to keep track of cost, quality and time schedule.

Most of the requirement changes and new requirements were clear during the first two phases, which were immediately incorporated in software design. As shown in Figure 4 effort estimation chart of design phase, more than 40% of design time spent on design change due new requirements or change in customer requirements.

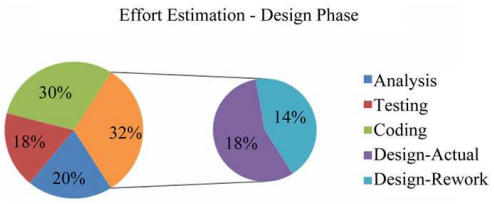

Figure 4. Effort estimation-design phase

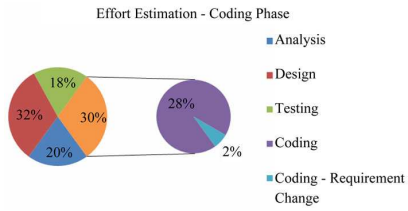

Figure 5. Effort estimation-coding phase

If a flaw is found in the plan, major changes will need to be made during or after coding. This could result in a waste of productive time. As described earlier, majority of the rework happened in design phase only. Due to which the development team got the clear development vision. The rework done in coding phase estimate only 2% of entire development time or 6% - 7% of coding phase time, as shown in Figure 5.

The development process followed component driven approach, all the requirement changes or new requirement easily accommodated in development process. As development team was able to monitor the development at unit level, the problems identified at earlier stages and modified within scheduled time and cost. The Figure 6 describes the project stability based on changing requirement in each phase on monthly basis. The design phase starts in the second month and got 50% stability at end, because no major design changes happened after that. Major work in the coding phase starts in fourth month, got 85% stability in the end and so on.

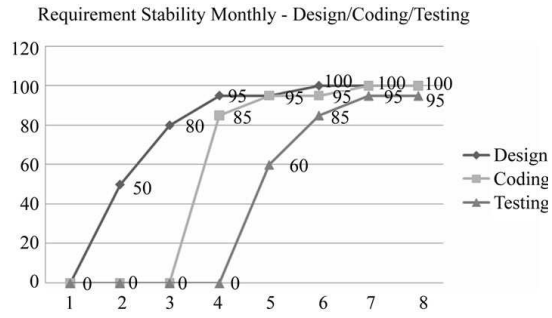

Figure 6. Requirement stability monthly-design/coding/testing

Finally the testing team was able to work parallel to the development team on delivered component, bugs/ issue raised by testing team fixed during development and again delivered for testing phase. This insures the delivery of quality product within given time frame.

CONCLUSİON

We have discussed a fusion process model for software development process and 3C-Model for each phase of development process model. Fusion process model includes an explicit phase for searching design alternatives in the corresponding solution space and selecting these alternatives based on explicit quality criteria. It has been implementation in commercial software company. The key results in this paper include the fusion process model, analysis of fusion process model and experience of project manager and team leader using fusion process model. The experience indicates results which demonstrate how this approach helps in controlling the overall development process by implementing component based approach. Fusion process model ensures the overall quality of software system; reduce the development cost and time by considering the changing requirements of customer, risk assessment, identification, evaluation and composition of relative concerns at each phase of development process.

REFERENCES

[1] A. Jansen and J. Bosch, Software Architecture as a Set of Architecture Design Decision, 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, 6-10 November **2005**:109-120.
[2] J. Lee, Software Engineering with Computational Intelligence, Springer Publication, **2003**:183-191.
[3] X. Ferre and S. Vegas, An Evaluation of Domain Analysis Methods, 4th CASE/IFIP8 International Workshop in Evaluation of Modeling in System Analysis and Design, **1999**:2-6.
[4] B. Boehm, *IEEE Transaction on Software Engineering*, **1984**;10(1):4-21.
[5] B. Boehm, *IEEE Computer*, **1988**;21(5):61-72.
[6] A. Hamid and S. E. Madnick, *Communication ACM*, **1989**;32(12):14-26.
[7] J. Ropponen and K. Lyytinen, *IEEE Transaction on Software Engineering*, **2000**;26(2):98-112.
[8] N. Medvidovic and R. M. Taylor, *IEEE Transactions on Software Engineering*, **2000**;26(1):70-93.