

Opening Windows of Software Construction with more Productive Techniques

L.İlteris ÖNEY

E1305937 – Computer Engineering Department – METU

Date : 09 June 2004

Dr.Semih ÇETİN

ABSTRACT

Motivation to write this article is based on explanation of streamlining the Software Construction. Problem that is explained in this article is defining step by step relations between common concepts which are Object Oriented Approach, Component Based Software Engineering, Agile Approach, Plan Driven Methodologies, Extreme Programming and Software Construction. Component Based Software Engineering is pioneer of productive Software Construction. To achieve productivity of software construction without losing quality and scope is explained as balancing Plan Driven Methodologies and Agile Approach with perspective from benefiting Component Based Software Engineering concepts. Finally one type of Agile Approach Methodology which is Extreme Programming is defined and role of it in streamlining software construction is expressed.

Keywords: *XP, CBSE, PDM, OOA, Software Construction, Component, Object, Methodology, Agile Approach*

1 INTRODUCTION

Software construction is one of the biggest emerging issue in Software world. Software Projects are day by day increasing in count also in complexity and size. This rapid improvement requires efficiencies in software projects. Our main aim is to provide methodologies about software construction and how to increase productivity. This concept is explained below starting with Compo-

nent Based Software development what it is about and how it streamlines our construction, after then explaining how to benefit both new Agile Approaches and Plan Driven Methods. Finally one of the Agile Approach Methodology which is Extreme Programming (XP) is explained further.

As in the fact there exist no silver bullet in Software Engineering world some Methodologies which is based on approaches helps our systems increase their production streamline if they are correctly preferred. I hope this article will help you to understand the core facts.

2 ROLE OF OBJECT ORIENTED APPROACH IN COMPONENT BASED SOFTWARE ENGINEERING

Component-based Software Engineering (CBSE) is concerned with the development of software systems from reusable parts (components), the development of such reusable parts, and with the maintenance and improvement of systems by means of component replacement and customization[1]. Object Oriented Approach involves encapsulation, data hiding, membership functions or methods, attributes or instance variables. To re-define components will help us to solve the relation. Component is nontrivial, nearly independent, and replaceable part of the system. In short Component is a piece of software that has a

collection of objects which has an extensive set of mechanism and larger units of granularity. Components interfaces are also too complex in spite of an object. In real world we can look at things as an object and systems as a component. But to define these systems they must have some existing interfaces. In object oriented approach we look at the whole world as an object, analyze, design and implement it. In component based software engineering we start development with components. Motivation is to select reusable components. Passing the analysis and definition part we reach component selection and evaluation part which we start to construct our system. For every component to evaluate we use Object Oriented approach to find out their reusability, maintenance and interoperability. We can buy it from off-the self, use previously created component or create from scratch. Every component creation or selection is based on Object Oriented Approach methodology and we all provide OOA steps to figure out their functionality and usability. Components may change but the methodology never changes.

To explain the role of Object Oriented Approach in Component Based Development is every selection, creation or maintaining a component in our component based system approach is done via Object Oriented Approach. For example we encapsulate our implementation details by setting interfaces in CBSE.

3 COMPONENT¹ BASED SOFTWARE ENGINEERING INDISPENSABLE FOR SOFTWARE CONSTRUCTION²

Software constructed systems can be built from assembled components. It is essential for such systems to have predictable behavior, if the risks of failure are too high. To enable practical and modular³ verification of real systems, software architects need to learn to build both behavioral specifications of components and component implementations that are suitable for the internal systems. Neither of these tasks can be automated, in general. Every components reusability is a one question and these components maintenance is another problem. Component based software engineering is indispensable for software construction because software components can be understood as code blocks, previously tested, with well defined interfaces which offer advantages to make easier for software construction. But we must never mix component based software engineering with system development with software components. Here our main aim is constructing the software with reusable component even they are off-the self, internally ready or can be produced. For off-the self product one of the major concern is maintenance also. Our construction must involves a B plan for every component. Also Component Based Software Engineering is very important especially in the areas of Distributed System Development⁴.

¹ Component : a nontrivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture

² Software Construction : The general name for detailed design, coding, unit testing, and related activities - the collection of activities focused on creating source code.

³ Module : In software, a module is a part of a program. Programs are composed of one or more independently developed modules that are not combined until the program is linked. A single module can contain one or several routines.

⁴ Distributed System : A system in which computations are performed by separate programs that co-operate to perform the task of the system as a whole

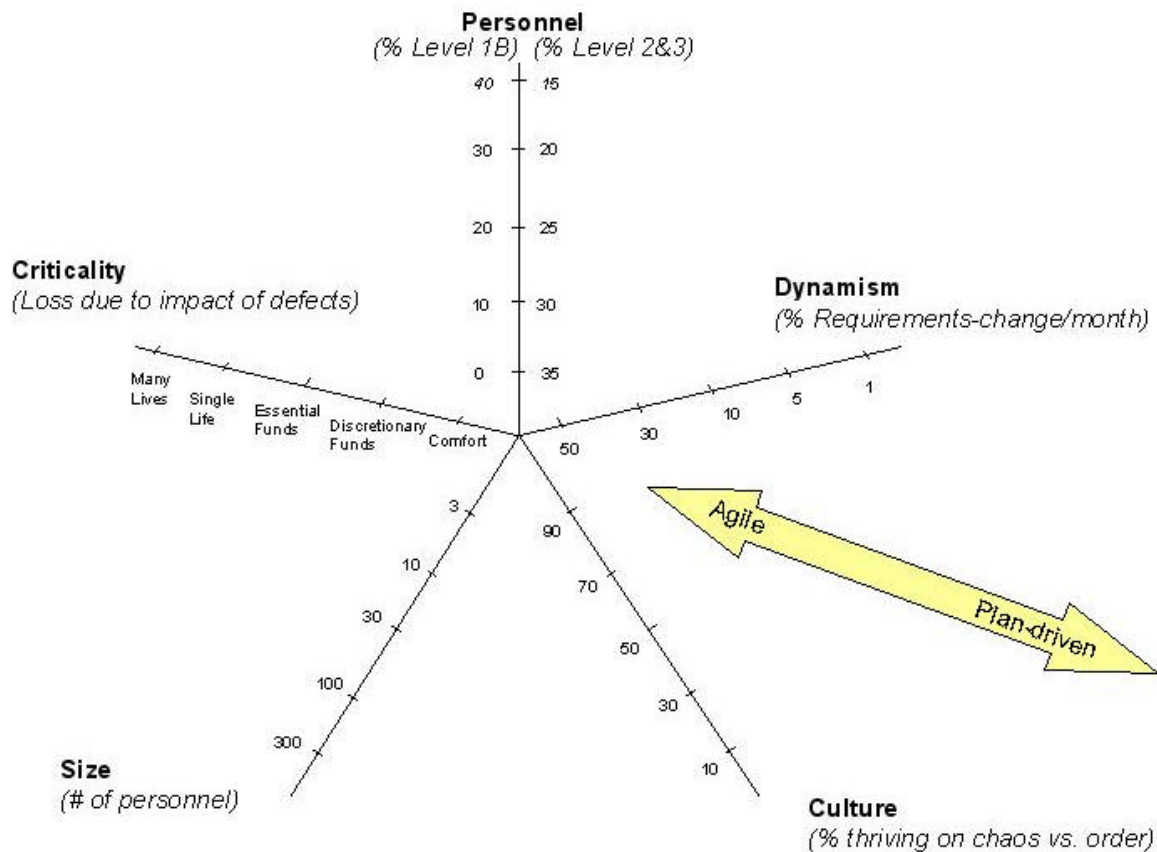
Component Based Software Engineering has a major goal which is mission for Software Construction which are Reusability, Separation of Development Processes component based development (consumer perspective) and Component Development (producer perspective). In Software Construction even from the word construction we mention about reusability. All experience can be reused and integrated in Software Engineering. Analysis is necessary to determine when and if appropriate[2]. We can think software construction as a blue print of a system. If this blue print is build on some modules that can be reused and gather to provide some functionality than construction will be more stable that it have ever be. Component Based software Engineering has also main considerations that effects software construction which are component maintenance, integration and cost. For selected component which can be off-the self we have a question about if component producer can deliver a new version if required. In construction as we mentioned before there must be a tip what to do in construction design if producer can not deliver it. Also Integration is another question how it will effect to whole system and the new version if appears, will it be compatible with the whole system.

Finally due to construction is pattern based OO, these patterns main goal is reusing the experience. What we call this experience is components. We can increase usability only with Component Based Software Engineering and streamline our processes.

4 WHERE TO USE PLAN DRIVEN METHODOLOGIES AND AGILE APPROACHES FOR COMPONENT BASED SOFTWARE ENGINEERING

Neither plan driven methodology nor agile provides a silver bullet. What we mean here about silver bullet is there is no single solution or exact solution for all types of problems. Software architectures must approach the problem with using a certain methodology⁵. This methodology can be a balanced methodology which is taking different properties of some methodology that is required for our case. If we have a look at from the Component Based Software development side, our approach must not only includes solving the problem also our problem domain and environment where the solve will occur. Component Based Software development requires software development process to fit the systems approach. This is not always the case if we accept Plan Driven Methodologies as a single base. Some problems have properties like significant change in time, standard processes exist and all documented clearly, requirements are not clear and subject to change also . These reasons are all about problem domain. There is some considerations about the company involved in solution or size of the problem also. Plan Driven Methodology workers capabilities are also different from agile methodology works. We can define it like expertise here also important. In the following figure it is defined where to choose agile, where not to choose or both.

⁵ Methodology : codified set of recommended practices, sometimes accompanied by training materials, formal educational programs, worksheets, and diagramming tools. An approach to solve a problem



Comparison has five critical dimensions. Size, Culture, Dynamism, Criticality and Personnel[3]. As the number of personnel increases it will be easier to accomplish a plan driven methodology. But with only three personnel involved in the process it is obvious that no project can be accomplished with plan driven methodology. If we would like to mention about culture which is the personnel's success in the environment where chaos⁶ or order⁷ exist. Whether personnel is more reluctant to work in an area having strictly defined processes then PDM is more efficient or not take agile. Dynamism is about the system alteration. If system require more changes in a short period it is obvious not to use plan driven methods due to when it is complete then the system will not be on obey time-to-market. Criticality is when a change or defect happens it shows the risk

about the project. As the risk increases agile is no longer seem to be fair. Finally about personnel .Each software company have some sort of personnel's. These personnel's expertise differs because of the work that is accomplished by that company. If the personnel has some lower expertise but higher in count plan driven methodology fits best here .

If personnel are few and expertise is high agile concept required. An essential point about agile methods, and a clear distinction between them and the plan-driven methods, is that agilist firmly believe that "individuals and interactions" matter more than "processes and tools." This people oriented assumption means that agilest will always expect a high-ability cohesive team do better than a team of lower ability, whichever one fallows more agile app-

⁶ Chaos : Program development environment where there is no set of rules to process

⁷ Order: Program development environment where strictly rules are set.

roaches. The assumption does mean that organizations should put their primary emphasis on getting and growing high-ability people for their teams[4]. Which is not low-ability team will do worse with agile than plan-driven methods.

But now here comes an important question. It is obvious that in some dimensions our environment will fit in agile in some how it will fit plan driven. So how can a architect can choose one?. The answer is choose both. But the measure will involve importance of the factors. Dynamism is the most important dimension that we defined above .After selecting priorities we have stable solution appears on our table. The solution involves both agile and plan driven concepts. How can will it be?

We will use plans to scale up our agile methods and we will use agility to streamline plan driven methods[5].In agile methods due to complexity of the problem we must have some plans on. These plans can be about function integrations, some planned test patterns, make it more modular or upfront the architectural planning. With planning the lack of communications between functional parts will be overcome. To streamline our plan driven method in our project we can use spiral method as a base and every part of project plan we can use benefits of agile methods just like communications, expertise, preliminary design, testing as-you-go, customer collaboration etc...This type of work will reduce our construction and implementation time and increases motivation to project progress also without losing scope and quality.

5 ROLE OF XP IN SOFTWARE CONSTRUCTION

We have many Agile Methods in software world. Examples are Adaptive Software Development, Crystal, Scrum, Feature Driven Development and probably the most famous method Extreme Programming(XP)⁸. Here I will explain Extreme Programming and its role in Software Construction. Extreme Programming as the name implies taking into account the common practices and principles of software and pushes them to the extreme to achieve most effective and productive processes. Extreme Programming is a method of Agile Approach and Agile Approach is an approach for Software Construction. So what we are trying to do with XP is to Construct our Software with an agile approach which is Extreme Programming. XP is centered around keeping things simple and reaching to a goal which is accomplishing the most important functions of the project that will be used in time. This is because customers are not faithful enough often don't know what they want until you put a product in front of them, and their own desires, schedules, and funding are likely to change frequently over the lifecycle of a software product. Extreme Programming prevents golden plate syndrome⁹ of the project.

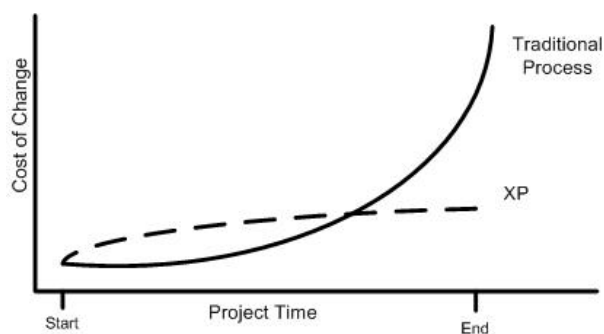
Let's have a look at four variables of XP. These are Cost, Time, Quality and Scope. The main aim is delivering the software in less time and cost without losing much from the quality and scope. Doing the things in short period of time can have negative effect on quality but extreme program-

⁸ Extreme Programming : is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

⁹ Golden Plate Syndrome : All requirements accepted even they are really required or not.

mers strongly believe that not time-to-market software that has perfect quality is useless . To achieve time-to-market without losing quality and scope is explained below.

XP has four values which are communication, simplicity, feedback and courage . To decrease cost and increase quality there must be a mechanism along the team members so that improved communication appears. Communication is important because in agile approach our team is either lot of experienced or a few non experienced members. These members must communicate accordingly to reflect their capabilities. Extreme Programming is based on people so communication required to achieve quality and prevent unnecessary possible changes. As expressed in previous part to get benefit of extreme programming requirement definition must be frequently changes over time. The cost of changes in traditional methods are too high to cover. (See figure)



XP encourages “Make it simple” and expand it as required . Feedback is between programmers and customer to check stories that the programmers have created. This is done with short releases which allows customer keep their requirements and designs simple, avoiding speculative design features that might never be needed . Finally XP programmers have the courage to start it all over.

By the way in analysis part of traditional approaches we are consuming too much effort to document and analyze the whole system. Infact that is not required for most of the project. There can be common requirements that are need to be done quickly and efficiently and some requirements are just nice to have. Document and construct a blueprint takes time and urgent requirements will not be appears in short period of time. XP flattens the cost of change over time so any change which is required can be added to project easily on later stages.

XP is more about shortens traditional approach and give flexibility concepts to support the methodology. Activity of XP is planning in 2-3 weeks, Designing, Coding and Testing. Programmers refactor the process as required.

Consequently XP is one of the main methodology in Agile Approach. To accomplish Software Construction we must be either have traditional approach or agile approach. XP is the famous agile Approach but not every organization can or is willing to use XP. It requires an on-site customer, or at least a programmer that knows their needs. Programmers must have significant expertise and XP requires a lot of trust and sacrificing of control between management and the developers. XP is mainly people oriented, and gives new ideas about communication like pair programming , unit testing etc. If these things sound ok or at least possible for your organization, you can explore more about XP and decide whether you want to give it a try and how to go about it. XP will help you to streamline your project but project manager must consider the plus or cons deeply to decide to use or only just get some benefits from .

6 CONCLUSION

Software System Developers or Architects face to a problem initially to prefer an approach to solve problems. These preferences based on the situations of environment, problem domain and cost issues. Our aim was to support information to this preference from the point of view streamlining the Software construction. Software Construction has some standing points which are Component Based Software Engineering to improve, find or create our components we have Plan Driven methodologies and Agile Concepts. We explained what is Component Based Development and relations with the Object Oriented Approach and why it is crucial in Software Construction from the view of reusability. Even do we provided an approach which can be plan driven but benefiting Agile concepts with differences between these two approaches and explained one of the Agile Method Extreme Programming which have fundamental benefits of cost, speed without losing quality and scope. As in the fact there exist no silver bullet in Software Engineering world architects must construct their software development systems based on considering the situations. For every problem domain there can be an optimum solution to be tracked. Our aim was providing concepts to increase productivity and tracking this on the optimum.

CONTACT

Lütfi İlteris ÖNEY.

Middle East Technical University

ioney@tobb.org.tr

REFERENCES

- [1] Clemens Szyperski “*Component Based Software Engineering*” Edinburg Scotland <http://www.sei.cmu.edu/pacc/CBSE7> 24-25 May 2004
- [2] Dr. Semih Çetin “*Object Oriented Construction – Component Based Development*”- April 2004
- [3] Richard Turner “*Using CMMI to Balance Agile and Plan Driven Methods*” The George Washington University OUSD (AT&L) CMMI Technology Conference Denver CO November 19, 2003
- [4] Martin Fowler,” *Can average developers use agile methods*” Thought Works <http://www.martinfowler.com/bliki/IsAgileForAll.html> 4 April 2004
- [5] Boehm/Turner, “*Balancing Agility and Discipline A Guide for the Perplexed*”, Addison Wesley, Boston, 2003.
- [6] Pat Hall “*Educational Case Study – What is the model of an ideal component? Must it be an object?*” The Open University England,
- [7] Barry Boehm “*Agile and Plan Driven Methods Oil and Water*” Agile Universe 2002 <http://www.agilealliance.org/articles/agileAndPlanDrivenMethods.pdf> 5 August 2002
- [8] Don Wells , “*The Rules and Practices of Extreme Programming*” 26 January 2003 <http://www.extremeprogramming.org/rules.html>
- [9] Brain Noyes “*Develop On the Edge With Extreme Programming*” <http://www.fawcette.com/resources/managingdev/methodologies/xp/default> 27 June 2002
- [10] Robert C. Martin and Robert S. Koss “*Engineer Notebook: An Extreme Programming Episode*” 1991
- [11] Peter Grassman “*Unofficial XP FAQ*” http://www.xprogramming.com/xpmag/unofficial_faq.htm