JKAU: Eng. Sci., **Vol. 21** No.1 pp: 19-35 (1431A.H./2010 A.D.)
DOI: 10.4197 / Eng. 21-1.2

Planned Methodologies vs. Agile Methodologies under the Pressure of Dynamic Market

M. Kamel, I. Bediwi and **M. Al-Rashoud**
Faculty of Computer Science and Information Systems, King Abdulaziz University, Jeddah, Saudi Arabia

Abstract. In the software development, the most challenging task is to develop projects under the pressure of dynamic market, where Time To Market (TTM) and requirements instability could fail the development process. Therefore project management should choose the development methodology that can control the problems associated with the dynamic market. Agile team argue that planned methodologies are heavy to cope with the rapid changes of the dynamic market, because the planned methodologies strongly emphasize on the planning process , by incorporating a lot of detailed design techniques like UML. On the other hand agile team claim that agile is the marvelous approach that has solutions for all problems related to the dynamic market, because agile achieves higher flexibility, and better to satisfy actual customer requirements. Agile achieves this, by developing and delivering the software product in an incremental fashion. Agile methodologies try to avoid any development overheads, and minimize unnecessary effort. This paper presents a comparative study that compares between planned methodologies -which have coupling relationship with UML analysis and design techniques – and the agile methodologies. The comparison compares between the two approaches in many respects, such as analysis, design, human resources, cost of the changes of the requirements and communication. The comparison shows how the lightness of the agile methodologies gives better responses to the different problems related to the dynamic market. Also the study shows that agile minimizes the cost of the changes of requirements during the development process.

1. Introduction

Planned methodologies are invented to control and solve the problems of "code and fix" development style, where the software is written without much of an underlying plan, and the design of the system is cobbled together

from many short term decisions. As the system grows it becomes increasingly difficult to add new features or to fix any bug [1]. Planned methodologies try to solve these problems by imposing a disciplined process upon software development, with the aim of making software development more predictable and more efficient. To be more predictable, planned methodologies focus on creating a comprehensive up-front design, from which detailed construction plans are formulated. "Waterfall and incremental models are the most two well known models of planned approach" [2]. As Fig. 1 shows "waterfall suggests a systematic sequential approach to software development that begins with customer specification of requirements and progress through planning, modeling, construction and deployment, culminating in on-going support of the complete software" [2] .

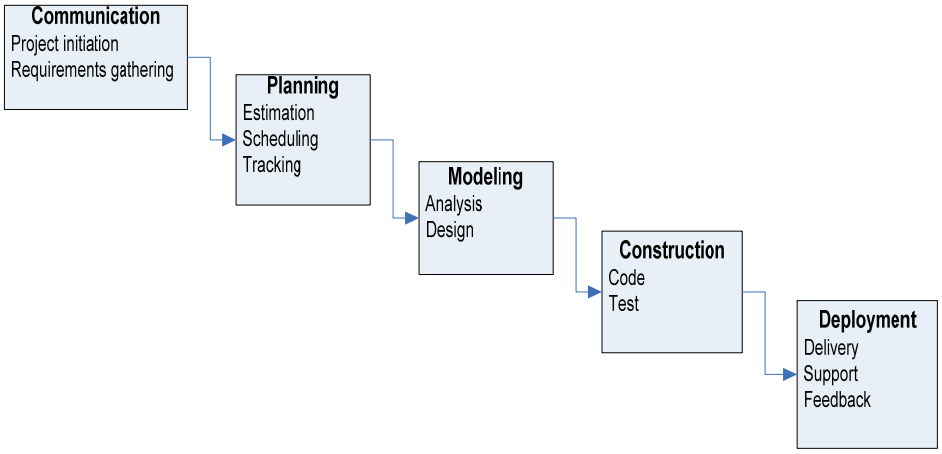

Fig. 1. Waterfall model.

"The incremental model combines elements of the waterfall model applied in an iterative fashion" [2]. By referring to Fig..2, the incremental model applies linear sequences in a staggered fashion as calendar time progress. Each linear sequence produces deliverable increment of the software.

Using incremental model, development team "can start working on the known increments, and clarify the rest later. Other problems may arise later if project is not well defined or if the definition changes much later. Rule of thumb is: 80% of the requirements should be known in the beginning. Development team should make a project priority chart, and

plan the increments accordingly" [3]. The trend of planned methodologies does not serve projects that have a compelling need to get software to market quickly such as web applications . Such applications exhibit a time to market that can be a matter of a few days or weeks with giving high consideration to maximizing product values and customer satisfaction. For that reason more flexibility and more customer involvement in the development process are needed. Agile methodologies are invented to be the super methodologies that achieve the needed flexibility. These methodologies aim to "satisfy the customer through early and continuous delivery of valuable software with high welcoming changing of requirements, even late in the development. Agile processes harness change for the customer's competitive advantage. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. Business people and developers must work together daily throughout the project" [4].

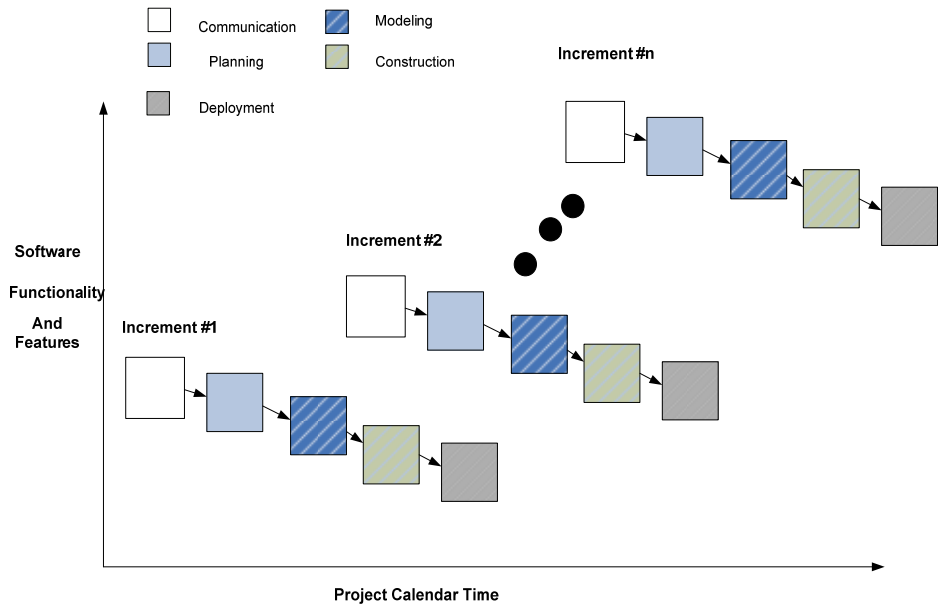

Fig. 2. Incremental model.

2. Agile and Planned Methodologies Requirements Analysis

When it is said requirements are predictable that means they are clear, well defined and well understood. The rate of changes in the predictable requirements is very small. "However most of the business software requirements are not predictable. Figure 3 shows that for many projects, it would be extremely rare for requirements not to be altered before the system is completed. In some applications the rate of changes is a matter of weeks or even days " [4] .

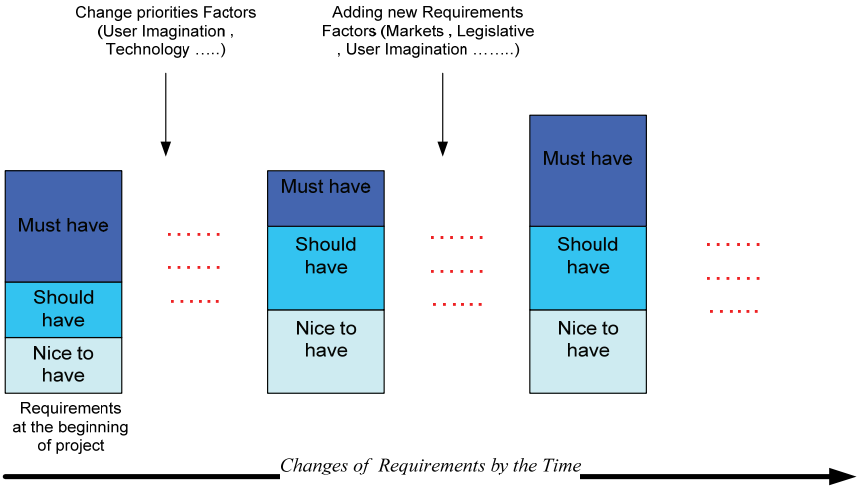

Fig. 3. Nature of requirements.

It is very important for project managers to know the proper reasons that could alert their project requirements. Then they should categorize the project under development, and specify if it is a predictable or a changeable project .Depending on this categorization the development methodology should be chosen. This obviously will minimize the risk of project failure.

2.1 *Planned Methodologies and Requirements Analysis*

The requirements in the planned methodologies are analyzed and defined upfront, and then the specification document and the software project management plan are produced". Once these documents are approved by the customer, they become the basis for the design phase which produces architectural and detailed design specifications. The

Object Oriented (OO) analysis employs a lot of UML analysis tools. The UML structure diagrams are used to identify the main structure of the system. The main functionality and the behavior of the system are identified using UML behavior diagrams. UML interaction diagrams are used to specify the flow of control and data among the components of the system" [5]. Consequently OO analysis is composed of three important modeling elements as Fig. 4 shows.

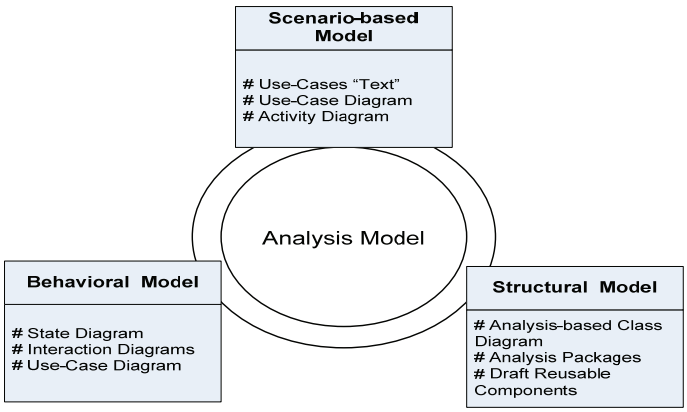

Fig. 4. OO analysis modeling elements

After the requirements are analyzed and modeled, they are well documented according to a selected documentation standard (such as Defense System Software Development Dod-Std-2167). All requirements are assumed to be static. The probability of changes is very little. All requirements are given the same priority. They are treated equally in the analysis effort. As Fig. 5 shows, once the requirements are gathered and analyzed, the customer pause his relation with the project until the product is finished. validations is accomplished by the customer only when the project reached to acceptance phases "includes acceptance tests, deployment and delivering " [6-7].

2.2 *Agile Methodologies and Requirements Analysis*

Unlike most of planned approaches, which deliver a monolithic system after a long development time , agile methodologies focus on generating early and small releases of working products, using mostly collaborative techniques, (XP uses pair programming and refactoring). The requirements in agile methodologies are implemented in an iterative manner. The most important requirements are implemented first. During

development process, customers work on site as team members. They carry the responsibility of controlling the requirements. Therefore they have the right to define new requirements, change their minds about existing requirements, and reprioritize requirements as they see fit (Fig. 6). They must also be responsible for making decisions and providing information in a timely manner [8]. They supply a continuance feedback about the developers understanding of the requirements and the progress and the quality of product.

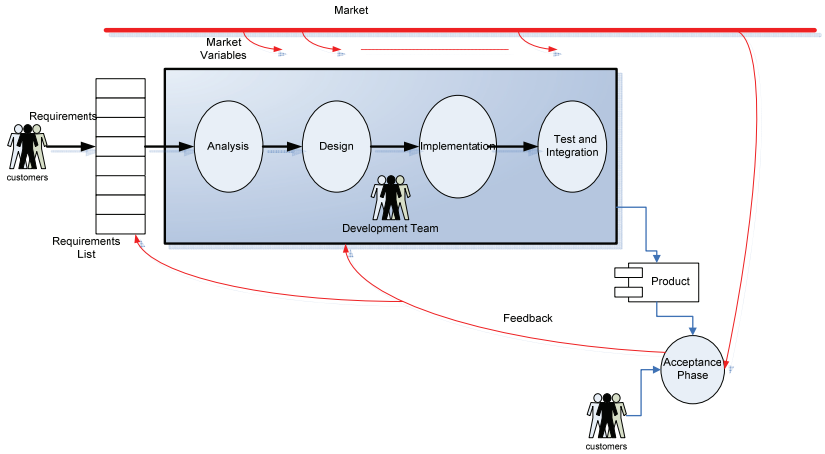

Fig. 5. Customer validations in the planned methodologies.

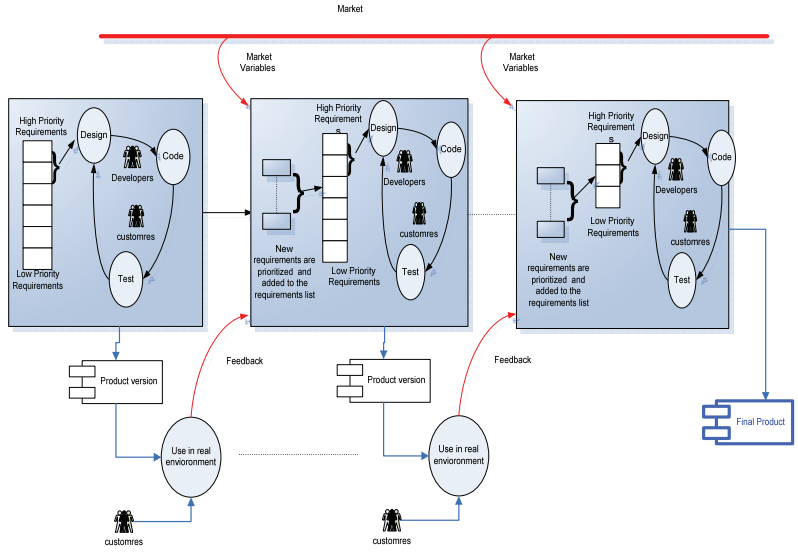

Fig. 6.Incremental fashion and customer validations in agile methodologies.

3. Design in Agile and Planned Methodologies

In the software development there are three styles of design. "First is evolutionary design (or no design). Essentially evolutionary design means that the design of the system grows as the system is implemented. Design is part of the programming processes and as the program evolves the design changes. This style of design ends up with a bunch of ad-hoc tactical decisions, which makes the product harder to alter, especially in the late phases"[9]. The Second approach is the long-term design, where a lot of complex and heavy design activities are done. This style of design is totally separated from coding activities; usually this kind of design is used by planned methodologies. Simple design (or design for today) is the third approach of design. Simplicity is the major feature of this style. The design is only applied for currently needed requirements. Agile methodologies use this style of design [1-2,9].

3.1 *Planned Methodologies and Design*

"Design in planned methodologies begins once the requirements have been analyzed, modeled and documented"[12]. in the planned methodologies, design team (designer) is usually separated from construction team (programmers). "Designers think out the big issues in advance. They don't need to write code, because they do not build the software, they design it. So they use a lot of UML design techniques that get away from some of the details of programming, and allow the designers to work at a more abstract level. Once the design is done, designers can hand it off to the programmers to write the code. Since the designers are thinking on a larger scale, they can avoid the series of tactical decisions" [9]. By referring to the documents of analysis model elements (Fig. 4); designers create four design models that are required for a complete specification of design (Fig. 7). The architectural models use information derived from the application domain, and analysis model (structural model) to derive a complete structural representation of the software, its subsystems and, its components. Interface design models represent the external and the internal interfaces to other systems, devices, networks or other procedures or consumers of information. Interface design models also represent the internal interfaces between various design components. Another aspect of interface design models is to model the interface with system users. Component-level models define each of external and internal components that populate the architecture of

the system. Deployment-level design models specify the elements allocating the architecture, its components and the interfaces to the physical configuration that will house the software [2] .

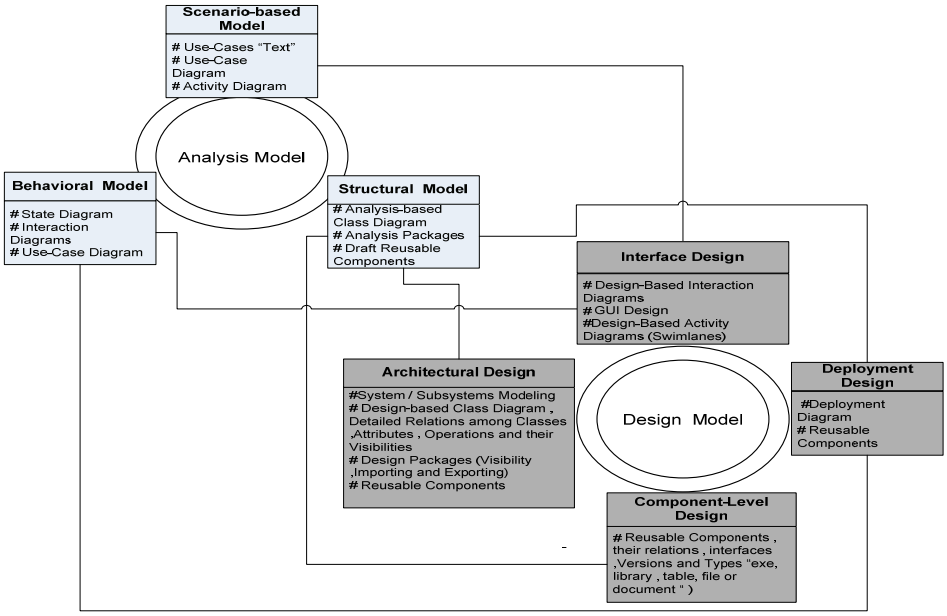

Fig. 7. The design models.

All design activities are well documented using a documentation standard that has been selected in the analysis phases. These documents will be the main source for the programmers to implement the system.

3.2 *Agile Methodologies and Design*

Agile design rigorously follows the (keep it simple/ and design for today) principle. Agile methodologies assume that more design for future, results in more complex design, which lead to more unnecessary costs as Fig. 8 & 9 shows. In addition to that, the design provides implementation guidance for a unit of requirements (XP uses user stories) as it is written nothing less, nothing more. The design of extra functionality (because it will be needed later) is discarding. Agile methodologies use simple tools to keep the simplicity. They do not elaborate in using complex and detailed tools. For example XP uses Class-Responsibility-Collaborator (CRC) model [10] .

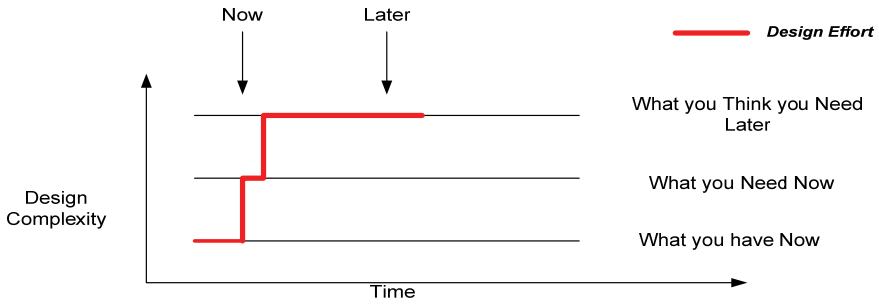

Fig. 8. The complexity of design for future.

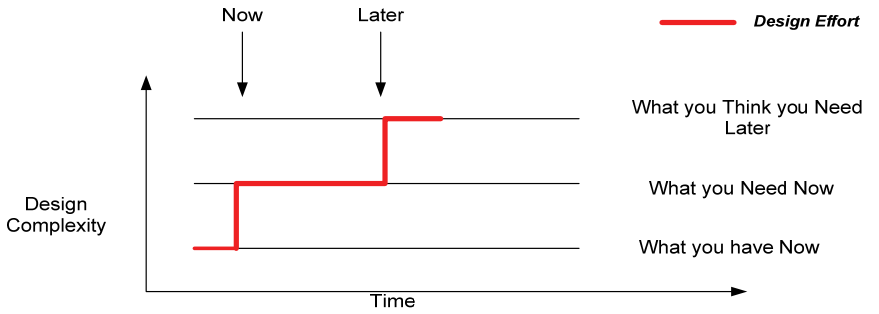

Fig. 9. The complexity of design for today.

If a difficult design problem is encountered, agile methodologies recommend the immediate creation of an operational prototype of that portion of the design (it is called in XP spike solution). The intent of that is to lower risk when true implementation starts and to validate the original unit of requirement. Agile encourages refactoring technique (Fig. 10). Refactoring is a reorganization technique that improves, simplifies and maximize the efficiency of the design (or code) of a component without changing its function or behavior. When software is refactored, the existing design is examined for redundancy, unused design elements, inefficient or unnecessary algorithm, poorly constructed or inappropriate data structures or any other design failure that can be corrected to yield a better design [2,10-11].

4. Agile / Planned Methodologies and Human Resources

Two main aspects should be taken in considerations, to investigate the effect of development methodologies on the human resources:

- Are the development methodology people orientated or process orientated?
- Communications among the members of development team.

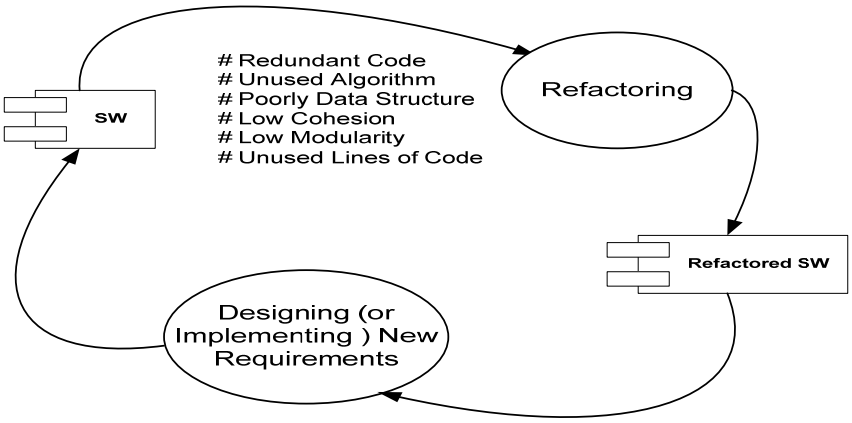

Fig. 10. Refactoring.

4.1 *People Orientation or Process Orientation*

Planned methodologies consider the people as replaceable parts, and available in various types (analysts, coders, test engineers, managers...). The individuals aren't so important; the concentration is only on the processes and on the roles that will execute these processes[1]. "However people are not static and predictable components. They are very highly non-linear variable and the most important factor in software development. COCOMO Cost Model and COCOMO II cost model show that 10:1 is the effects of personnel capability, experience, and continuity[12]. For that reason "agile development focuses on the individuals and interactions over processes and tools" [4] "They mold the process to specific people and teams. That means the process molds the needs of the people and team, not the other way around" [2]. Agile tries to build projects around motivated and high morale individuals, and give them the environment, courage, respect, trust and support the need to get the job done [4,10]. Figure 11 shows how the agile has high orientation to the people, whilst the planned methodologies orient toward the processes [2,12].

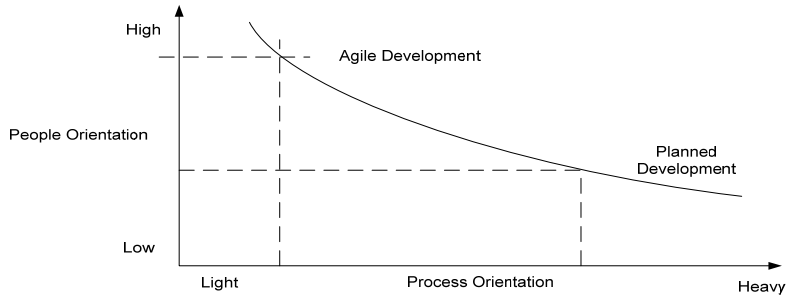

Fig. 11. People /process orientation in planned and agile methodologies.

4.2 *Communications Among the Members of Development Team*

Communication is very important value. It causes the success or failure of a software project, if it is / or is not applied in an efficient way. Communication could be achieved formally, through documentation, or informally through face to face meetings, and continuous discussions among the team members. Planned methodologies relay a lot on formal communication. A lot of work and efforts are consumed, to implement a lot of heavy analysis, design, and quality and test documents. For example to start any stage of waterfall model, it is necessary to have bunch of documents that are implemented by previous stages as Fig. 12 shows.

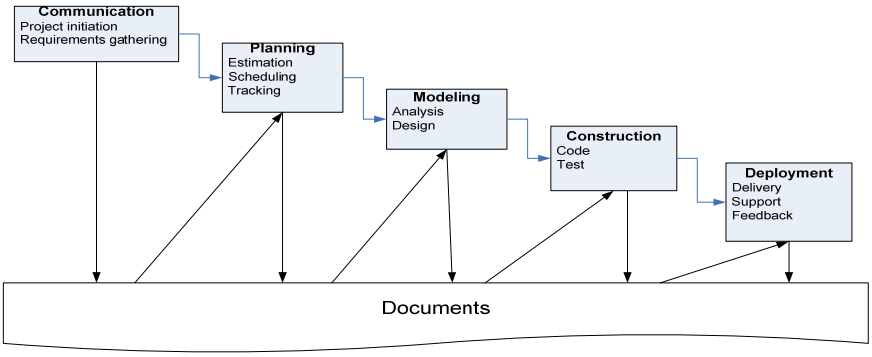

Fig. 12. Formal communication in the planned methodologies.

Agile methodologies rely heavily on communication through tacit, interpersonal knowledge. Knowledge is gathered through the team reviews. It is shared across the organization as experienced people work on more tasks with different people [12]. Agile considers the customer an important member of development team - on-site customer is one of XP

12 practices - the customer priorities, point of view and feedback are the main guider that drive the construction effort.

5. Development under Dynamic Market

Software project is considered under the condition of dynamic market, if it is under time-to-market pressure, with the volatility of requirements. Time-to-market is defined as to deliver a product to a market early. The sooner a market is penetrated, the earlier product sales and the revenues start [13]. The selection of development methodology is the corner stone to avoid many problems associated with dynamic environment. Table 1 shows the failure factors that could face the development team when they develop under dynamic conditions; also it shows how XP (most popular agile method) and waterfall (most popular planned method) react to these factors [3].

From the Table 1, it could be noticed that XP has more practical reactions to dynamic market factors. The reasons behind this are that "agile develops and deliver software in an incremental fashion, which achieve higher flexibility and better satisfy actual customer requirements. Incremental approach has many advantages over the traditional planned approach. Firstly, requirements can be prioritized so that the most important ones are delivered first and benefits of the new system gained earlier. Consequently, less important requirements are left until later and so if the schedule or budget is not sufficient the least important requirements are the ones more likely to be omitted. Secondly, it means that customers receive part of the system early on and so are more likely to support the system and to provide feedback on it. Thirdly, being smaller, the schedule/cost for each delivery stage is easier to estimate. The fourth advantage is that the user feedback can be obtained at each stage and plans adjusted accordingly. Lastly, perhaps most importantly, an incremental approach allows for a much better reaction to changes or additions to requirements" [1]. Also from Table1 it could be noticed that most failure factors centered on the rapid changes of requirements during development process. The major effect of changes in requirement is the cost that is spent on fixing the defects. It is very expensive to fix a change in requirements especially in the late phases of planned methods as it is seen in Fig. 13 [2]. Fixing errors increase exponentially the later they are detected in the development lifecycle because the artifacts within a serial process build on each other [14].

Table 1. Dynamic market problems and waterfall, XP reactions.

Failure factor	Waterfall reaction	XP reaction
Unclear project Objectives (lack of a project mission)	Waterfall model does not tackle especially this problem. The project team should stay on the specification phase, until project objectives are clarified	The customer participates in the weekly planning sessions, and checks that what is planned is consistent to what is expected. (on-site customer)
Underestimation of project size, complexity, novelty.	The waterfall model does not tackle especially this problem. The project team may need to replan the whole project.	One of the main features of XP is a continual planning during all the phases of the project. This controls the problem of the poor estimation of the project size.
Extreme project (high speed, high change)	Waterfall does not have enough flexibility to deal with such situations.	XP is targeted for extreme projects.
Project execution incomplete requirements/ specs (poorly defined parts), lack of user input	Waterfall model does not tackle especially this problem. The project team should stay on the specification phase, until project objectives are clarified	The customer participates in the weekly planning sessions, and checks that what is planned is consistent to what is expected. (On-site customer).
Unstable (volatile) requirements	Waterfall model does not tackle especially this problem. Frequent and/or late changes are not welcome	Welcoming changes is the true nature of XP. The project is redefined on weekly basis.
Poor requirements management	There should not be many changes at all, since the uncertainties are supposed to be resolved at the first stages	This is a part of XP planning game. However, because of the nature of XP development, there is not much formal change management.
Project redirected (profound changes of the schedule/ functionality/ resources)	The pure waterfall model cannot adapt well to major midcourse changes. The lifecycle must usually be restarted	The customer can present new specifications (new user stories) on the weekly meetings.
New, immature software technology	There is a high risk the project will be cancelled. Waterfall assumes a mature, stable environment	Often this does not match well with the XP philosophy of 'quick planning' and 'simple design'. The Infrastructure is assumed to be doable on the fly.
Project cancelled	The project cannot show any results (except documentation) since no working software is available before the integration stage	The customer can cancel the project any time on hir will. What has been achieved to that point can be taken into use
Unattractive Software release (wrong, obsolete or missing features)	The release is built according to the initial requirements phase. If that phase was conducted poorly, the resulting release is likely to be unattractive	This is tackled with XP planning game. The customer selects the features to be implemented.
Integration difficulties	Waterfall model directs to integrate the whole system in one shot, which often leads to integration difficulties.	One of the main XP features is the continuous integration. Therefore there are not much integration difficulties.

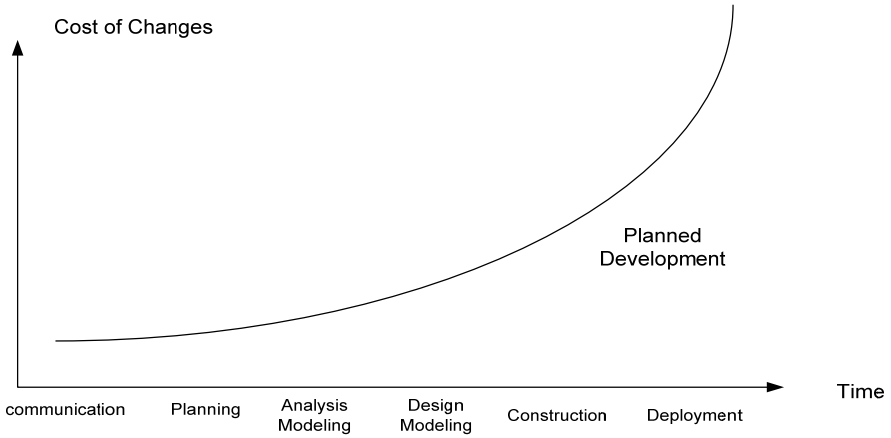

Fig. 13. cost of changes in the planned methodologies.

In agile methodologies the effect of changes of requirements is minimized as Fig. 14 shows. And controlled by depending on implementing requirements in small releases. The changes of requirements during small period of time seldom happen , and if they do so they are immediately prioritized by project stakeholders, and added to the requirements stack in the appropriate increments .

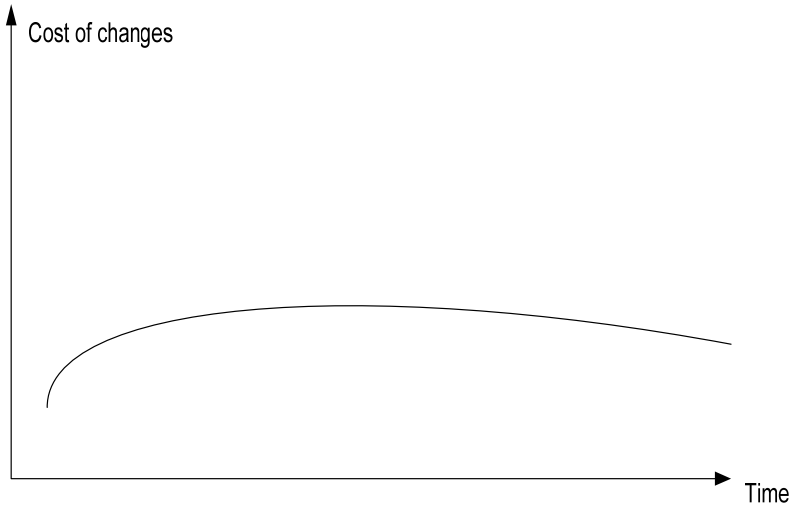

Fig. 14. cost of changes in the agile methodologies.

6. Conclusion

When the agile methodologies are compared to traditional approaches, there is much less up-front planning. Agile methodologies are more adaptive and less predictive than traditional software planned methodologies. There is less emphasis on documentation. They rely more on tacit knowledge and oral communication than on written documentation. Requirements engineering happens throughout the process. Even though the majority of requirements engineering is performed at the beginning of each release of the project, there is no longer the notion of a requirements engineering stage. Agile methods value people over processes and depend on the fact that having competent people on a project is more crucial for its success than the development process. Agile methodologies improve a team's capability of dealing with changing requirements. Thus for a project following the agile practices, the slope of the cost of change curve is no longer exponential, but rather linear or even logarithmic.

References

[1] **Fowler, M.,** *The New Methodology* (2005), available on line, www.martinfowler.com/articles/newmethodology.html

[2] **Roger, S.,** *Software Engineering a Practitioner's Approach*, McGrow-Hill International Edition (2005).

[3] **Petri, K.** and **Maarit, L.,** How to steer an embedded software project: tactics for selecting the software process model , *The Journal of Information and Software Technology*, **47**: 587–608(2004).

[4] *Agile manifesto*, available on line http://agilemanifesto.org

[5] *The Wikipedia*, the free encyclopedia, available on line http://en.wikipedia.org/wiki.

[6] **Lee, M.,** *Just-in-Time Requirements Analysis – The Engine that Drives the Planning Game*, available on line, www.agilealliance.org/system/article/file/1007/file.pdf

[7] **Mrenak, G., Poston, T.** and **Schein, J.,** *Agile Software Development Methods* (2004), available on line, http://www.topdownsoftware.com/Practitioners/Life_Cycle_Models/AgileMethods.html

[8] **Ambler, S.**, *Agile Requirements Change Management* (2006), Available on line, http://www.agilemodeling.com/assays/Agile RequirementsChange Management.htm

[9] **Fowler, M.,** *Is Design Dead* (2005), available on line, www.martinfowler.com/articles/designdead.html

[10] **Beck, K.,** *Extreme Programming Explained – Embrace Change*, Addison Wesley Longman, Inc, Reading, Massachusetts (2000).

[11] **Fowler, M.,** *Refactoring*(2005), Available on line, http://martinfowler.com/distributedcomputing/refactoring.pdf

[12] **Turner, R.** and **Boehm, B.,** People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods, Cross Talk, *The Journal of Defense Software Engineering*, **16** (2003),*www.janes-defence-weekly.com*.

[13] **Pfeiffer, S.,** Requirements Engineering for Dynamic Markets, *Thesis for the Degree of Master of Science in Software Engineering*, Department of Electrical and Computer Science, Calgary, Alberta (2003).

[14] **Ambler, S.**, *Examining the Agile Cost of Change Curve* (2006), available on line, http://www.agilemodeling.com/assays/ExaminingtheAgileCostofChangeCurve.htm

مقارنة بين منهجية تطوير البرمجيات القائمة على التخطيط ومنهجية التطوير الذكي والخفيف والسريع تحت ضغط تغير متطلبات السوق

محمود كامل، وإبراهيم عبدالمحسن البديوي، ومبارك الرشود
كلية الحاسبات وتقنية المعلومات – جامعة الملك عبد العزيز
جـــدة – المملكة العربية السعودية

المستخلص. تتمثل المهمة الأكثر تحديا لتطوير برمجيات المشاريع المختلفة في ديناميكية السوق، حيث يلعب عنصر الوقـــت تحـــدياً أساسياً في نجاح المشروع ومنافسته للمشاريع الأخرى. ويمثل عدم استقرار متطلبات الزبون تحدياً آخر له نفس التأثير، ولذلك تلعـــب منهجية تطوير البرمجيات دوراً أساسياً للتغلـــب علـــى التحـــديات السابقة.

يقوم هذا البحث بمقارنة بين منهجيـــة التخطـــيط (Planned Methodology) بمنهجية التطوير الذكية والخفيفة والـــسريعة (Agile) ولبيان مميزات وعيوب كل من المنهجيتين، حيث تمثل المنهجيـــة الأولى الإغراق في مراحل التحليل والتصميم ثم الانتقال إلى التنفيذ والاختبار مما قد يؤخر تسليم المشروع، وقد يؤدي هذا التأخر إلى تغير متطلبات الزبون في ظل ديناميكية السوق فضلاً عـــن عـــدم وضوح واستقرار متطلبات الزبون في المرحلة الأولـــى مـــن أي مشروع. بينما تمثل المنهجية الثانية التعاون بين الزبون ومبـــرمج المشاريع أثناء المراحل المختلفة لتطوير البرمجيات فـــضلاً عـــن اختيار المتطلبات الأكثر إفادة في بداية المشروع والبدء الفوري في تنفيذها وتطوير البرمجيات على هيئة إصدارات متكاملة.