I.J. Mathematical Sciences and Computing, **2015, 4, 21-33**
Published Online November 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2015.04.03

***Available online at** http://www.mecs-press.net/ijmsc*

Data- and Workflow Customer-Oriented Software Process Models

Yazan Al-Masaf'ah[a], Ali M. Meligy[b] and Alaa S. Farhat[b]

[a]*Faculty of Information Technology, Middle East University for Graduate Studies, Jordan*
[b]*Dept. of Mathematic & Computer Science, Faculty of Science, Menoufya University, Egypt*

Abstract

This paper presents a dataflow model to control the flow of data in each phase of a customer-oriented software process model. In addition, we suggest a workflow model to describe the transaction between the model phases, and a role model to govern the personnel participation and roles. Our goal is to develop models that involve the customer frequently and effectively during project development. Testing the models using CHAOS Report and shows that our models are capable of achieving this goal.

Index Terms: Dataflow Models, Workflow Models, SDLC, RUP, JAD

1. Introduction

Customer trustworthiness is the most important factor in the success of a software project, trustworthiness or dependability "essentially means the degree of user confidence that the system will operate as they expect and the system will not 'fail' in normal use [13]. Software development processes that focus on rapid delivery and considering the customer changes in requirements are essential. Extreme Programming (XP) provides principles for guiding projects and relies on the participants for its success [14], [2,3] and [8]. Scrum concentrates on how the team members should function in order to produce the flexibly in a constantly changing environment [12]. Rational unified process (RUP) is a well-known software engineering process that provides a disciplined approach to assigning tasks and responsibilities within a development organization [7]. All the needs of every stakeholder are considered in the Inception phase, alongside with critical use cases to be used, the candidate architectures of the system, and the schedule and cost of the entire project. [1].

This model enhances customer satisfaction and the quality of the delivered software. The motivation was the fact that customers change their requirement too often and the supplier understanding of a customer requirement is not always right. This makes the product loops into iterations until it reaches a certain level that satisfy most of the customer needs. These iterations will lengthen the implementation time, and hence increase

* Corresponding author. Tel.:
E-mail address: ameligy@ugs.edu.jo

its cost, as well as, reducing customer and supplier profit. We have suggested solving these problems by developing a new software life cycle that enhances the customer involvement in the process, and thus reducing the main effects that slow the software development process, and add to its cost. More details are included in [11]. The new software development life cycle (SDLC) focuses on enhancing the customer participation in the software production processes, relying on his knowledge in the environment, practices, communication methodology, and structure of the hosting environment.

2. Related Work

Agile Software Development has strong relation with the desired model since it focuses on customer participation as well as individual communication. Hence, major agile development methodologies (Scrum, Feature Driven Development, Dynamic System Development Method and Adaptive software development) along side with the resent studies on them, are described in detail in this section. Roles in the Scrum are distributed between scrum master, product owner, scrum team, Customer and management [12]. Customer participation is on the tasks related to the Backlog items for the system being involved or enhanced. It has been noted that Scrum teams with Scrum masters seems to scale naturally specially where strong technical leadership is applied [4].

Feature Driven Development (FDD) was first reported by Coad et,al, [5] it does not cover the entire software development process, but rather focuses on the design and building phases [12]. FDD provides the methods, techniques and guidelines needed by the project stockholders to deliver the system. Rational unified process (Rup) is a well-known software engineering process that provides a disciplined approach to assigning tasks and responsibilities within a development organization [7]. Masaf'ah presents Dynamic System Development Method (DSDM). DSDM consist of some phases such as feasibility study, business study, design and implementation. The feasibility study phase is mainly concerned with the technical ability to build the required software, judging the domain of the project as well as, deciding whether to use DSDM or not. The business study involves workshops where a sufficient number of customer's experts are gathered to be able to consider all relevant aspects of the system including the requirements and the effected business process. Another two outputs are the architecture definition and the prototyping plan.

Adaptive software development (ASD) focuses on the problem in developing complex, large systems. The method aims to provide a framework with enough guidance to prevent projects from failing into disorder [10]. Joint Application Design (JAD) is used in the System Development life Cycle to collect business requirements while developing new information systems for a company. It consists of a workshop where "knowledge workers and IT specialists meet, sometimes for several days to define and review the business requirements for the system [9].

3. The Proposed Model

Our proposed model contains five phases:

- Customer Preparation Phase.
- Requirement Engineering Phase.
- Design and Development Phase.
- Testing Phase.
- Closure Phase.

The customer preparation stage determines the nature and the scope of the development, as well as provides basic understanding to the customer team about the nature of software projects and software development. This stage should include a series of activities that covers the following areas:

1. Feasibility study of the project.
2. Defining stakeholders, project managers, and project team
3. Customer team training.
4. Defining the project business goals and objectives.
5. Producing and signing of the project charter.

The major role in the second phase is for the customer, where they make the election of the requirements, the analysis and validation of these requirements will be done by the supplier, and the definition will be done by both parties as well as the specification.

For the requirement engineering phase in particular, the model will use a RUP-like process –i.e. a process that follows the RUP flow to build the requirement of the system. The validation process tries to answer the following:

- Validity: Does the system provide the functions which best support the customer's needs?
- Consistency: Are there any requirements conflicts?
- Completeness: Are all functions required by the customer included?
- Realism: Can the requirements be implemented given available budget and technology?
- Verifiability: Can the requirements be checked?

In the third phase the designers of the supplier will design the software architecture based on the developed behavior tree and the Software Requirement Specification SRS. The main part in this phase is for the supplier, but the customer team that was trained in previous phases will start working at this phase. This phase includes six processes:

1. Behavior tree analysis: The development team will start the development process by analyzing the BT that was produced from the requirement phase.
2. Software architecture: The architecture of a software system refers to an abstract representation of that system. The system architecture will based on the system BT, considering various subtrees, which will make the architect work easier.
3. Integration protocol definition: During this activity, a protocol will be created to identify the coding scheme and the protocol that should be followed by the developers for subtrees integration.
4. Teams distribution: Depending on the architecture results, the development team will be divided into several -but smaller- teams. Based on the size of work, the software manager will distribute the subtrees over these teams, where related subtrees will be assigned to a single team.
5. Subtree coding: During this activity the developers will start coding the subtrees where successfully coded requirements will be moved to the testing phase. The customer team, on the other hand, will create a communication channel between various project members and teams trying to explain the requirements.
6. Subtree Integration coding: If a subtree is tested successfully, a development team specialized in integration will glue the subtree code to the other parts of the system that are already developed.

Upon completion of all the subtrees integration, the system will be sent for acceptance testing. The customer team will work closely with all teams trying to explain the requirement to developers. In the testing phase is a separate phase which is performed by a different team after the implementation is completed.

There are four types of tests in this phase:

- Internal testing
- Subtree testing
- Application testing

- Acceptance testing

Internal testing deals with low-level implementation, where each function is tested. This test is conducted by a developer on a single node -on the behavior tree- or a group of connected nodes. Subtree testing deals with testing a single sub tree. This could test the interaction of many functions but impound the test within one subtree. Application testing deals with tests for the entire application. The application must successfully execute all scenarios before it is ready for general customer availability. Acceptance tests are black box system tests. Each acceptance test represents some expected result from the system. A node is not considered complete until it has passed its acceptance tests.

Closure phase is the final phase of the proposed model, it contains the following activities:

- Delivering the final product to the customer.
- Delivering the project documentation to the customer.
- Performing training for the customer operational team.
- Project Evaluation.

4. The Dataflow Model (The Model Main Tool)

The data flow of the proposed model is based on the behavior tree: the behavior tree contains three parts (figure 1):

1. Root node: in the below figure, R0 represents a root node for the whole tree, R1, R2 and R3 represents root nodes for its derived subtrees. Also R2.1 could be considered as root node for the nodes R2.1.1 and R2.1.2.

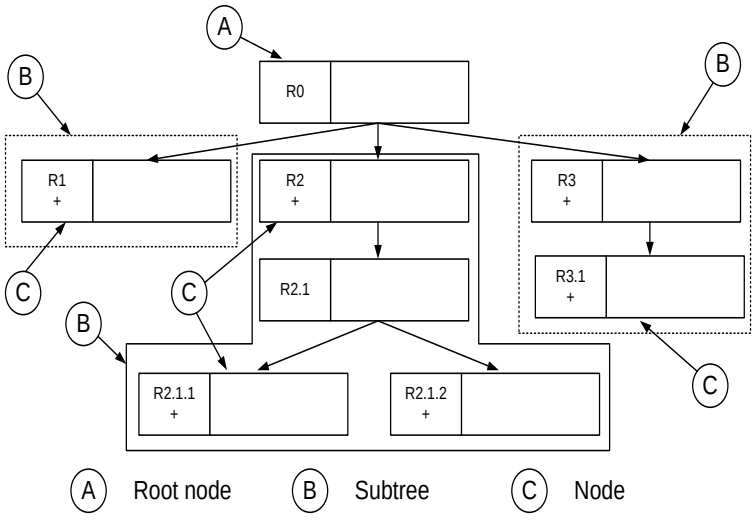


Fig. 1. The model behavior tree parts

2. Subtree: any part of the tree that can be categorized as related requirements, the sub tree will be named by the root node of it, in the below example, the selected sub trees are named as ST1, ST2 and ST3 respectively. If R2.1 is considered as a root node, its sub tree will be ST2.1.
3. Node: a single requirement that is represented on the behavior tree in one block only.

The behavior tree will represent only functional requirements that are well defined on the SRS. A coloring scheme should be adopted by the project stakeholders to indicate the status of a certain node, four colors should be defined, each will represent one of following:

1. Node is in requirement engineering phase.
2. Node is in the development phase.
3. Node is in the testing phase.
4. Node is ready.

Figure 2 shows the life cycle of a single node. A node could go back and forth between the project phases, once it's ready, it can not move any more. A subtree passes through the same process as a single node.

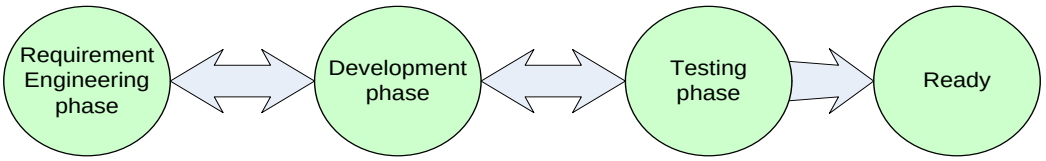


Fig. 2. Data flow of a sub tree and a single node

5. The Workflow Model (Transaction between Phases)

Based on behavior tree nodes status, the phases of the proposed model are interleaved; for example, while a certain subtree are in development phase, another subtree in the same project could be in the testing phase and another one may be in the requirement phase. The transaction between phases is based on the calling system – shown on figure 3; the calling system defines the authority of a decision on a certain phase, as well as, governs the project transaction between phases.

The proposed model suggests three main types of calls:

- Customer call: A customer initiated request to move a single node, a subtree, or the whole behavior tree from a phase to another.
- Supplier call: A supplier initiated request to move a single node, a subtree, or the whole behavior tree from a phase to another.
- Unified call: A customer/supplier initiated request to move a subtree or the whole behavior tree from a phase to another.

Calls can be major or minor. In major calls the whole behavior tree is moved to another phase, while in minor calls, a subtree or a single node is moved. Every phase on a project has certain types of calls, based on the authority in this phase. In the customer preparation phase a major unified call will be performed if both parties -the customer and supplier- reached an agreement to proceed with the project. This call will officially indicate the beginning of the requirement engineering phase.

In the requirement engineering phase, the following calls could be issued:

- Customer major call: Upon completion of this phase for the first time, the instance tree will complete and all nodes have the same status the customer will issues a major call to move the whole instance tree to the design and development phase.
- Customer minor call: A single node or a subtree could be sent back to this phase from the design and development phase, or the testing phase, they will be sent to design and development phase again, and their corresponding status will be changed accordingly.

The authority in the design and development phase is in the hands of the supplier, who may issue the following calls:

- Supplier minor call: if any defects are found with a single requirement (node) or with a subtree, the supplier makes a call to send the defected node(s) back to the requirement engineering phase.
- Supplier major call: if the whole application is finished –every node in the behavior tree is satisfied- the supplier will issue a major call to declare the end of the design and development phase and to require application and acceptance testing, the whole instance tree status will change accordingly.

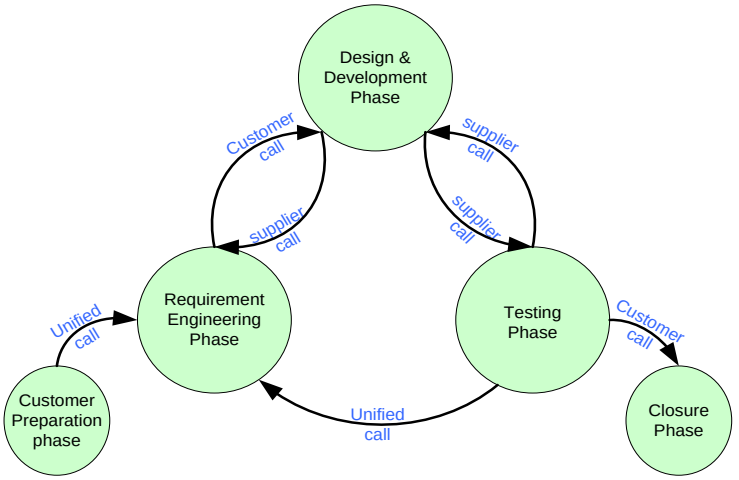


Fig. 3. The calling system

In the testing phase the following calls can be initiated:

- Supplier minor call: the supplier will be responsible on internal and subtree testing, if a nodes or a subtree fails to pass on of these tests, it will be sent back to design and development phase with development status.
- Unified minor call: if a defect is found in the supplier understanding of subtree, the customer can agree to repeat the requirement engineering process for this part again, and the status of the contained nodes will changed as well.
- Unified major call: if the system does not satisfy one of the basic principles of software development, like dependability or security, during the application testing process, the whole instance tree could be sent back to the requirement phase, repeating the to return the whole project starting from the Requirement phase, all the nodes statuses will be initialized to the requirement phase status.
- Customer major call: upon successfully completing the application testing with the supplier, the customer will mark the whole instance tree as ready, and the acceptance test will take place.

No calls are conducted through the closure phase since the development process will be finished by then.

6. The Role Model (Major Roles and Responsibilities)

There are different roles in the proposed model for different tasks and purposes during the process and its practices. These roles are described as follow:

- Customer and Supplier stakeholders: The managers from the customer and supplier end, that are responsible of declaring the preparation and the closure of the project. The stakeholders will monitor the project through weekly progress reports initiated from the supplier and the customer project managers.
- Supplier project manager: The model do not define any customized tasks for the Supplier project manager, other than controlling the customer project manager role and involving him in some of the project management activities, and initiating the supplier calls and unified calls from supplier end.
- Customer project manager: Works alongside with the supplier project manager main tasks include:

  - Creating the project charter with the supplier project manager.
  - Explaining the customer requirements to the design and development team of the supplier in the design and development phase.
  - Communicating with the customer to handle any urgent requests from the supplier.
  - Selecting the customer testing team for the acceptance testing, and performing unit, subtree and application testing with the supplier testing team.

The customer project manager will assess the supplier project manager in performing the following tasks:

1. Analysis and design of objectives and events.
2. Risk Management
3. Organizing the work
4. Defining the products of the project
5. Issues management
6. Defect prevention
7. Communicating to stakeholders

- Technical writer: Responsible of documenting the JAD meetings, writing the requested documents from the project manager, and building the SRS. Having technical writers can offer several benefits:

  - Technical writers are skilled information gatherers, ideal for eliciting and articulating customer requirements.
  - Technical writers can better assess and plan documentation projects and better meet customer document needs.
  - Technical writers know how to determine the questions that are of concern to the user or customer regarding ease of use and usability.
  - Technical writers involved early and often in the process, can become an information resource throughout the process, rather than an information gatherer at the end of the process.

- Software manager: The development manager is responsible of selecting and organizes teams for each subtree or group of subtrees. He is also responsible on the integration plan between each group of requirements, by building a basic structure for the integration that all the teams should follow to guarantee smooth integration of subtrees.
- Process controller: A supplier repetitive that is responsible of monitoring that the model process is followed, he is responsible of controlling and guiding the project members to follow the model process.
- Tree builder: Responsible in building the behavior tree and tracking the status of each node as well as defining the nodes that moved at each calls.
- Testing manager: Responsible of performing and managing all the required tests, and providing feedback to the customer and supplier project managers.
- Consultant group: This group contains two types of personals that participate in the project in the requirement phase:

- Customer consultants who are experts in a certain products that the needed software need to integrate with it.
- Third party consultants who are owners of a product that the requested software need to integrate with.

Any contact between the customer and the supplier representatives will be done through the project manager. The role and communication model between the customer and the supplier is shown in figure 4.

7. Contributions, Conclusions and Future Work

The contribution of the presented software development methodology will be illustrated in two main perspectives. At the outset, the benefits that the model has presented will be argued. Then, the effectiveness of the model will be measured.

A. Tree-based Data Flow

In addition to its effective rule in the requirement phase, in terms of defect detection, representing the system in a behavior tree could help in the following aspects:

- Simplifies tracing of requirement –i.e. the phase of each node.
- Simplifies nodes (requirement) reporting between the customer and the supplier.
- Modulating the system into subtrees that can be developed simultaneously.
- Simplifies the selection of testing cases in the acceptance test, and the expected behavior will be obvious.

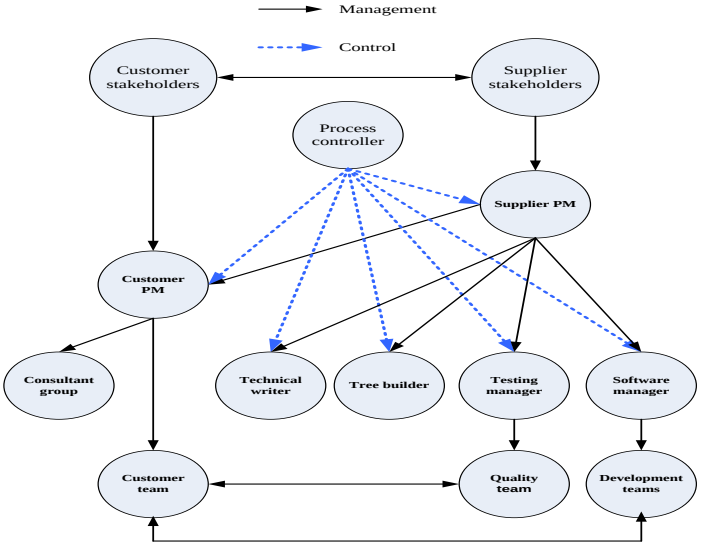


Fig. 4. The role and communication model

B. Call-based Work Flow

Both the customer and the supplier along with the whole development life cycle will benefit from the proposed calling system:

- It clearly identifies the decision maker in each phase of the project.
- The responsibility comes with the decision; hence, call maker will make deep analysis before issuing any call.
- Controlling the process flow.
- A clear declaration of moving nodes between phases to make all the teams synchronized.

Interactive Customer and Supplier Teams

The customer and the supplier will work as a single team in the project. Benefits of this method include:

- The customer will have a better understanding in the value of change in requirements.
- Any delay in project delivery will be justified from the customer perspective.
- Simplifies the testing for the customer team.
- Flexible adaptation to the software after delivery.

The ultimate goal of this method is to make sure of gaining satisfaction of the customer, which will lead to long term profit of the supplier. The roles and responsibilities are distributed between the customer and the supplier, which will do well to both ends. For example, having a customer project manager will:

- Increase and guarantee customer's team dedication to project.
- Speed up deliveries from customer end, like requirement clarifications.
- Making sure that the supplier project manager and his team is fully focused in the project goals.
- Making sure the project time plan is followed and gives early alarm in case of a proper delay.
- Identify risks from the customer end.
- Reporting the project progress to the customer end stakeholders.

The Model Effectiveness

Measuring the effectiveness of the proposed model needs a reliable and trusted method that comprehensively analyzes the factors of success and failure in projects. We will measure our model effectiveness based on comparisons with the CHAOS report and the CMMI model.

CHAOS Report Confirmation

One of the measuring references for this model will be the "Standish Group CHAOS Report 2004. The objectives of CHAOS research are to document the scope of application software development project failures, the major factors for failure, and ways to reduce failure. (Hartmann 2006)

For purposes of the study, the projects were classified in the report into three resolution types:

- Project success: The project is completed on-time and on-budget, with all features and functions as initially specified.
- Project challenged: The project is completed and operational but over budget, over the time estimate, and offers fewer features and functions than originally specified.
- Project impaired: The project is canceled at some point during the development cycle.

The resolution of projects in 2004 is illustrated in figure 5. Figure 6 shows the change in projects resolution from 1994 until 2004.

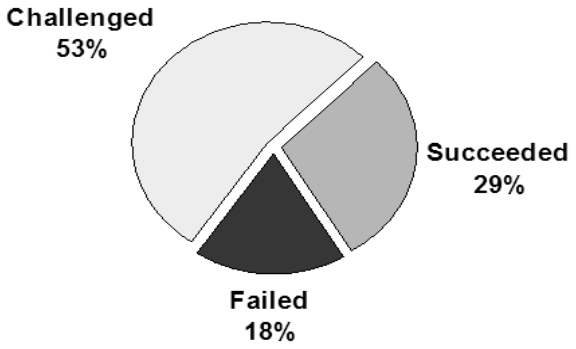
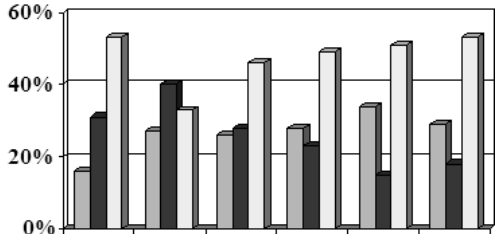


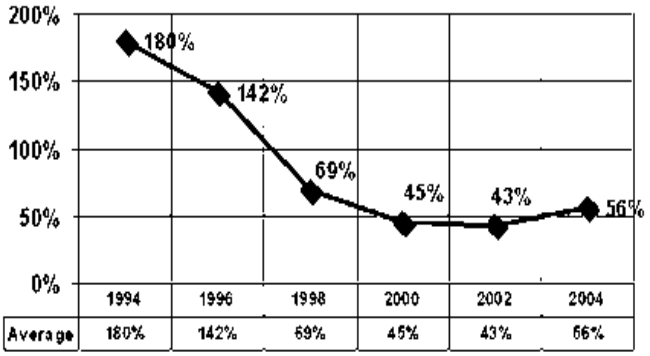
Fig. 5. CHAOS 2004 projects resolution



	1994	1996	1998	2000	2002	2004
Succeeded	16%	27%	26%	28%	34%	29%
Failed	31%	40%	28%	23%	15%	18%
Challenged	53%	33%	46%	49%	51%	53%

Fig. 6. Change in projects resolution (1994-2004)

Figure 7 shows the average percentage of cost overrun over 1994 till 2004, and figure 8 illustrate the average percentage in the time overrun during the same period of time. The most important aspect of the research is discovering why projects fail. To do this, The Standish Group surveyed IT executive managers for their opinions about why projects succeed. The three major reasons that a project will succeed are user involvement, executive management support, and a clear statement of requirements.



	1994	1996	1998	2000	2002	2004
Average	180%	142%	69%	45%	43%	56%

Fig. 7. Average percentage of cost overrun (1994-2004)

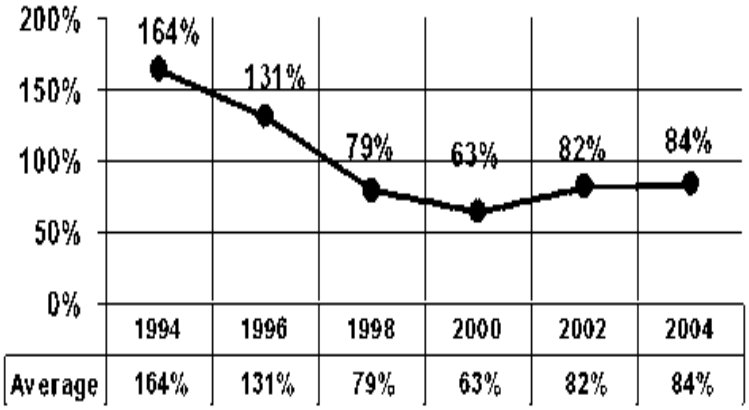


Fig. 8. Average percentage of time overrun (1994-2004)

There are other success criteria, but with these three elements in place, the chances of success are much greater. Without them, chance of failure increases dramatically. The project top ten success factors was:

1. User involvement 15.9%
2. Executive management support 13.9%
3. Clear statement of requirements 13.0%
4. Proper planning 9.6%
5. Realistic expectations 8.2%
6. Smaller project milestones 7.7%
7. Competent staff 7.2%
8. Ownership 5.3%
9. Clear vision & objectives 2.9%
10. Hard-working, focused staff 2.4%

Other 13.9%

The survey participants were also asked about the factors that cause projects to be challenged. Top ten factors for Project failure was:

1. Lack of user input 12.8%
2. Incomplete requirements & specifications 12.3%
3. Changing requirements & specifications 11.8%
4. Lack of executive support 7.5%
5. Technology incompetence 7.0%
6. Lack of resources 6.4%
7. Unrealistic expectations 5.9%
8. Unclear objectives 5.3%
9. Unrealistic time frames 4.3%
10. New technology 3.7%

Other 23.0%

Opinions about why projects are impaired and ultimately canceled ranked incomplete requirements and lack of user involvement at the top of the list. The main factors for impaired projects are:

1. Incomplete requirements 13.1%
2. Lack of user involvement 12.4%
3. Lack of resources 10.6%
4. Unrealistic expectations 9.9%
5. Lack of executive support 9.3%
6. Changing requirements & specifications 8.7%
7. Lack of planning 8.1%
8. Didn't need it any longer 7.5%
9. Lack of IT management 6.2%
10. Technology illiteracy 4.3%

Other 9.9%

Major factors that affect software development can be derived by analyzing the results from the above figures; it can easily noted that the most effective factors in project -listed from the most important to the least important- are:

1. User involvement.
2. Clear statement of requirements.
3. Executive management support.
4. Realistic expectations.
5. Staffing.
6. Proper planning.
7. Clear vision & objectives.
8. Technology.
9. Smaller project milestones.
10. Ownership.
11. Didn't need it any longer.
12. Unrealistic time frames.
13. Lack of IT management.

The results amazingly comply with the process of the proposed model; from it intensive focus on customer involvement, passing by the long and comprehensive requirement phase, along side with enhancing the management role –the stakeholders- in projects. The model also peruses realistic expectations by involving customer in the development process, and clearly identifies the role of each project team member –the staff- in the process. It can be noticed that, the main factors that affect projects are actually the points of focus of the developed model.

Conclusions and Future Work

It was initially assumed that the outcome model aims to enhance the customer contribution to the development process, as well as trying to increase the customer side participation in project management activities. The model tried to accomplish this by building:

- A software development model
- A role model
- A dataflow model
- A work flow model

All of the above models identified clearly the parts of the customer and the supplier and the boundaries between them.

Along side with this, deep and thorough analysis must be done to check the ability of proposed role, dataflow and work flow models.

References

[1] Abrahamson, P., Salo, O., Ronkainen, J., Warst, J. (2002), *Agile Software Development Methods, Review and Analysis.* VIT Publications 478.
[2] Beck, K (1999), *Embracing Change with Extreme Programming*. IEEE Computer 32(10).
[3] Beck, K (1999), Extreme Programming Explained: Embrace Change. Addison -Wesley.
[4] British Broadcasting Corporation (2007), Scaling Product Ownership, Agile Conference 2007.
[5] Coad, P., Levebvre, E. And De Luca, J. (2000), Java Modeling In Color With UML: Enterprise Component And Process. Prentice Hall.
[6] David Norton (2007), Agile Essence: Dynamic System Development Method, Gartner (G00150567).
[7] de Barros Paes, Carlos Eduardo Hirata, Celso Massaki (2007), *RUP Extension for the Development of Secure Systems*, Fourth International Conference on Information Technology.
[8] Gray, A. Jackson, A. Stamouli, I. Shiu Lun Tsang (2006), *Forming successful extreme programming teams*, Agile Conference 2006.
[9] Haag, Cummings, Mccubbrey, Pinsonneult, and Donovan (2006), Information Management Systems, for the Information Age. Phase 2: Analysis. Mcgraw- Hill Ryerson.
[10] Highsmith, J.A (2000) Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing.
[11] Masaf'ah, Y. A Customer-Oriented Softwrae Development Life Cycle, Master Thesis, Middle East University for Graduate Studies, Amman, Jordan, 2008.
[12] Palmer, S.R. And Felsing, J. M. (2002), a Practical Guide To Feature Driven Development. Upper Saddle River, NJ, Prentice Hall.
[13] Schwaber, K. and Beedle, M. (2002), *Agile Software Development with Scrum. Upper Saddle River*, NJ, Prentice Hall.
[14] Somerville (2007), *Software Engineering 8th edition*, Addison Wesley.
[15] Talby, D., Hazzan, O., Dubinsky, Y. and Keren, A. (2006*), Agile software testing in a large-scale project*, IEEE Software (Volume: 23, Issue: 4).

Authors' Profiles

Yazan Al-Masaf'ah he received the M.S. degree in computer Science at the Middle East University for Graduate Studies, Jordon in 2008.

Ali M. Meligy is a professor of computer science at the Menoufia University in Egypt. Previously, he was the head of computer science and information technology departments at Al-Hussein Bin Talal University in Jordan. His research interests include parallel processing and applications, distributed systems, Petri nets, and reuse-based software engineering.

Alaa S. Farhat She received the Bachelor's and M.S. degrees in Computer Science at the University of Menoufia, Egypt in 2010 and 2014, respectively**.**