Egyptian Computer Science Journal (ISSN-1110-2586)
Volume 40 – Issue1 January 2016

Paired Scrum for Large Projects

Fatma A. El-Licy
Department of Computer and Information Sciences,
Institute of Statistical Studies and Research
Cairo University, Giza
Why2fatma@yahoo.com

Abstract

Software has become part of all aspects of our lives, and organizations are increasingly conceiving extremely large and complex software projects. Agile development is a method of building software by empowering and trusting people, acknowledging change as norm, and promoting constant delivery of small pieces of working code. Yet, for large project, scaling Agile Scrum framework requires multi level management to coordinate the efforts and integrate the smaller pieces of the working code. An iterative paired Scrum framework is proposed to manage the development process of large project while exploiting the benefits of Agile methodology. Top-down requirement categorization and modulation is achieved to allow for the short cycle of the sprint in Agile Scrum framework. Requirement modularity while employing Scrum development methodology will, not only, help managing large project but also facilitating and acceleration its deployment. The different categories of the requirements constitute the Sprint modules, which iteratively reformed into production architecture.

The objective of this research paper is to establish a framework model that pairs modularity with Scrum methodology. The framework employs three levels of process development and management to facilitate and minimize deployment efforts, without, compromising the agility characteristics of the Agile Scrum framework.

Keywords: *Software Engineering, Agile Development, Scaling Scrum Framework, Scrum for Large projects, Requirement Modulation, Testing and Deployment.*

1. Introduction

Agile approaches embrace task simplicity, fast turnaround for smaller pieces of working code and human interaction over email/document/process oriented interaction. Simpler tasks make it harder for developers to misunderstand what the code is *meant* to do [1, 2].

The most popular approach of Agile today, is Agile with Scrum [3, 4]. Each characteristic of this approach has been selected to support agility. For example short development ‘sprints’

are used rather than the 18-month cycles common to traditional Waterfall development [5, 6, 7].

In scrum, requirements are broken down into ***Sprints*** which are one-to-four week coding cycles with specific tasks to be performed (not necessarily features to be developed). For large project, however, number of sprints could be hard to manage as individual tasks given limited number of developers. Several approaches were presented in the literature to improve the scalability of Agile approach in general, focusing in deferent aspects of software engineering practices, [8, 9,10,11, 12, 13,14] and Scum in particular [8, 9, 15, 16, 17, 18].

This paper proposes a framework model that handles large project as an abstraction of sets of functionalities with the aim to promote testing and accelerate products deployment.

This paper is divided into seven major sections. In addition to the introductory section, section 2, presents the proposed framework, section 3 illustrates the levels of development progress and section 4 presents the three management levels. Section 5 presents some related research articles, while, section 6 presents framework model evaluation and discussion. Finally, section 7, presents the conclusion.

2. Proposed Framework

The concept underlying the proposed framework is modularity that enforces the idea of divide and concur that promotes simplicity. The customer's proposed project (requirement), not necessarily formal, is categorized into several modules which specify the load of the project that, approximately, achieves customer goals. A draft of these categories is established and agreement of initial priorities is accepted by the project members during the pre-launching phase. For large project, each category is assigned to modules management team that device the given category into a set of modules. Each module is converted to a set of stories to be developed by the sprint developing teams, who will be the owners of the associated module. They build the story source codes incrementally [19]. Categories' owners are responsible for integrating, testing and deploying developed modules. Integration of a module's components is performed at a lower level, where the testing and deployment are also delegated to the actors of this level. The responsibility and roles are interrelated as will be discussed in the following subsections. Figure 1, illustrates the proposed framework which extend the project progress as a fountain of refinement flow of customer proposal into a complete product.

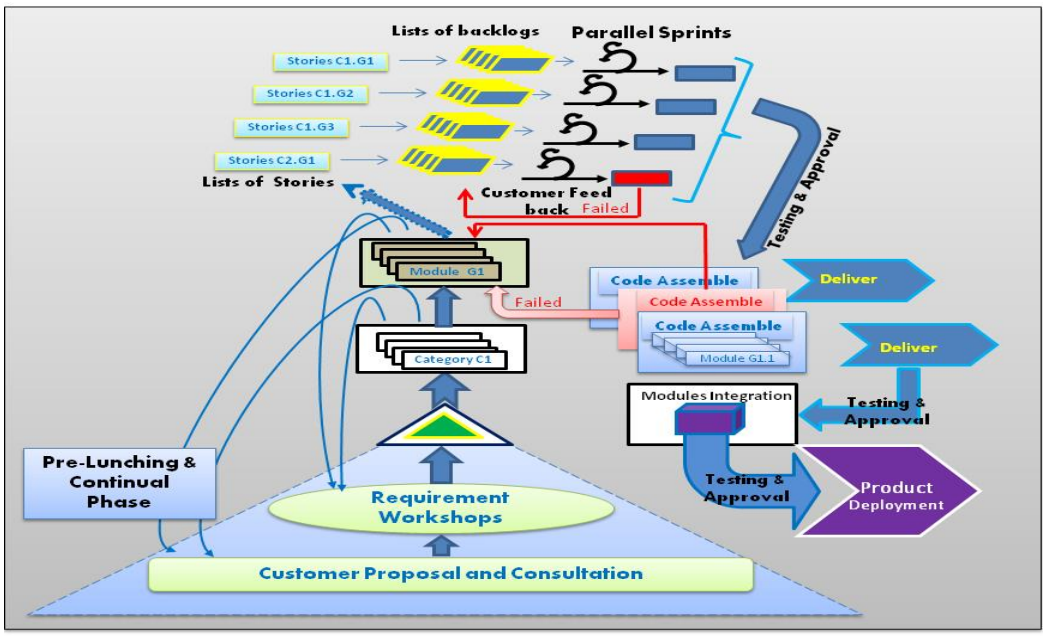

Figure 1: Proposed framework for Paired Scrum

2.1 Pre-Launching Phase: Requirement analysis

Collaborative requirement technique is adapted so that all project members are involved and participating in analyzing customer proposal. Project team collaboration in requirement analysis facilitates a project-wide sense of ownership – and also communicates a common understanding of what features need to be built. Collaborative requirement analysis produces more robust specifications.

Pre-launching conference involves the developers, Business Analyst (BA), Quality Control personals (QC), security engineers, Release Engineers(RE) and stakeholders (SH), is arranged. The project proposal is first presented by the customer in his own language, followed by the Project Manager (PJM), who frames the proposal in more technical speech. At this conference, no promises are made from either party. However, all members will study the proposal and prepare an initial view of its components, the risks involved, the security and vulnerability issues. As shown in Figure 2, several workshops will follow to specify the project's categories, the security and vulnerability issues associate with each category and the initial estimations of the required tools, equipment and man power.

Project requirement specification is discussed through several workshop iterations. A typical workshop will have at most 4 monitoring members. Each will offer his unique perspective, and through discussions they will identify edge cases, undefined requirements, opportunities, potential reuse and mainly a set of product categories. The developers may share their own concerns and flag some implementation issues, though. The customer should provide reasonable illustration and feedback for all concerns and queries. Similar workshops are carried out as part of the paired scrum cycles for refining requirement and merging changes.

The triangular shape in Figure 1 represents the categorization criterion, presenting security, production environment and peoples.

2.1.1 Project Commitment Meeting:

Define abstractions for the requirements as a set of categorized modules that specify the overall objectives of the project. This meeting should include all members of the project with customer agents, requirement engineers, project manager, program managers, developers, testers, security & quality control manager, Release Engineers and stakeholders. This meeting should conclude with the set of recommendations that specifies the making components of the project.

- Project component assignment of the modules
- Quality/security personals assignment for each project component
- define the working environment, therefore the required tools and platforms
- Define verification & security strategies, and working conventions

2.2 The Heist Level: L-Three {Category Level}

Whenever the project manager, stakeholders and the customer reach an agreement, this level presents briefs for each category (already agreed upon during the pre-launching phase). The categorization criterion was based upon:

- The size of the data and its dependencies
- The level of user interaction with the product
- The application domain
- whether it was Local, distributed or cloud based product

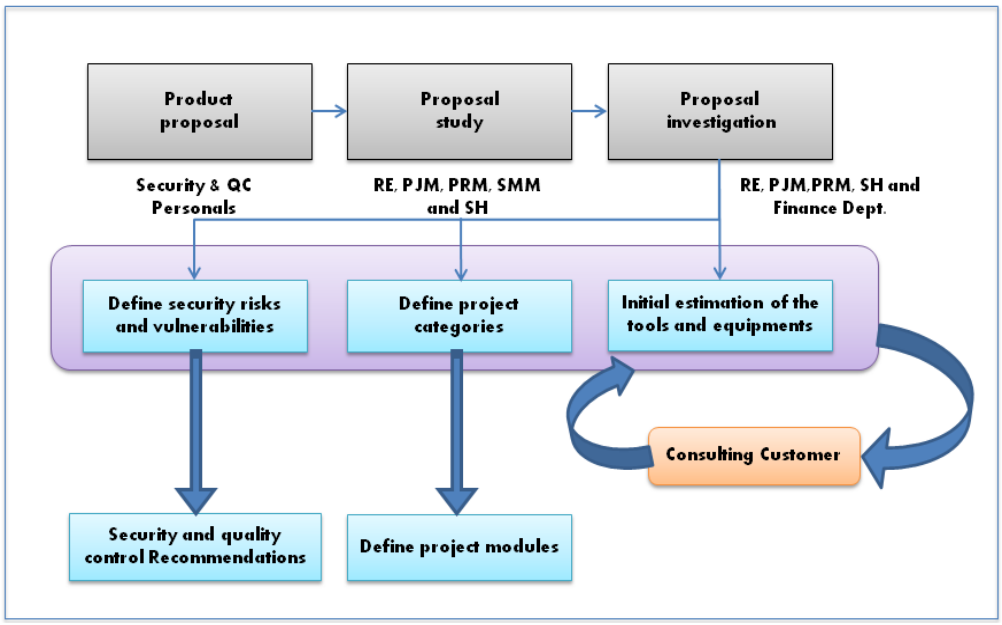

Figure 2: Two-to-four week preparation for project launching

The presented briefs for the categories should include:

- The category purpose
- A skeleton of test cases concerning the production environment
- A skeleton of security enforcing test cases covering production environment vulnerabilities and critical issues
- Other preconditions essential for each of the specified categories

The roles in this level are:

1. Business analyst ***BA***
2. Requirement engineering group ***RE***
3. Project manager ***PJM***
4. Product manager ***PRM***
5. Scrum module managers ***SMM***
6. Set of testers and security teams ***SQT***
7. Release management engineers ***RME***
8. Operation team

Quality assurance and security teams deal with the project at this level as black bags of purposes that should comply to the production environment and its security policies. Their main job is guaranteeing a successful deployment of the developed product after each release. Release Engineer and operation team set firewall policy and patches requirements for deployment environment. A release is assumed after the approval of a completed version.

2.3 Second Level: L-Two {Module Level}

A given Category is classified onto several modules. This initial hierarchical classification of the requirements assumes the leading production software architecture, which will evolve with Paired Scrum iterations into a concrete structure. The actors of this level are responsible for specifying modules, defining the interface for each and their level of interaction with the production environment. Customer consultation and feedback are assumed and essential here and always. This level deals with the classified modules, as black boxes, that constitute the project view.

With the help of Scrum Module's Manager, each Sprint Master is responsible for generating stories for the assigned module. Module specification, as part of requirement specification, must be pushed into development when they have been reviewed and accepted. Parallel efforts are assumed at this level to prepare stories for several modules. The roles in this level are:

- Product Manager ***PRM***
- Scrum Module Managers ***SMM***
- Sprint Masters ***SMA***
- Security teams & Quality control and Testers, (Security, Quality and Testing) ***SQT***

The responsibilities of the members of this level include:

1- Define the modules included in each category
2- Define the interfaces between modules within each category
3- Set and run the test cases concerning the defined interfaces
4- Set and run the security enforcing test cases covering the vulnerabilities and critical issues
5- Define other preconditions essential for each module (priorities, time dependencies, flow control, …etc)
6- Specify and run Quality Control tests includes fuzzing and exception testing.
7- Assembling the developed sprint build

At the end of sprint period, Scrum Module's team assembles the delivered source codes (typically from several sprint teams working in parallel) compile and run the associate testing. Normally, the specific Module's Scrum manager (who defined the module at an earlier stage) is responsible for receiving and assembling his/her module components.

2.4 First level: L-One {Components Sprinting Level}

At the sprint developing level is a release plan which highlights what functionality the team is planning to deliver for the next several iterations. Prior to Sprint Planning the Sprint Master forecasts the amount of story points that the team is expected to complete in the upcoming sprint.

Often this is based on the prioritized functionality and the amount of functionality that a team can produce in iteration ("velocity"). For the most current iteration, the team produces a detailed plan that includes the tasks needed to deliver the planned functionality. Code is typically committed near the end of the sprint.

The protocol relies on daily 'scrum' meetings, where all developers meet face-to-face to discuss what they will be working on that day. Each sprint focuses on iterative improvements rather than complete feature delivery. This ensures that only functional code modules are checked in — unlike Waterfall where every feature tends to get crammed in on deadline [6].

This multi-level planning allows the team to adjust their approach to account for changes in priorities, changes in the team, better approaches, and techniques learned during previous iterations, because they wait until right before they are going to do the work to determine how they will do it.

For large project, this level includes several sprint teams working in parallel to develop separate modules and obtain on site customer approval. During development; unit testing, static analysis and regression testing (whenever due) are performed.

Top-down development and release enhancement is achieved through two Scrum processes across the presented levels. S-M Scrum (L-One → L-Two) constitutes the delivery of potential components, whilst, M-C Scrum (L-Two →L-Three) constitutes the step-wise integration and deployment of potential Modules. This double scrum is called ***Paired Scrum*** process, Figure 3, illustrates the Paired Scrum process as viewed by of the proposed framework.

3. Three Level of Development Progress

Modules managers (at L-Two), Project Release managers, as well as L-One Sprint Master and Sprint teams are coordinating to transform customer proposal into a deliverable working product. Developers commit their source code for testing. Those codes that passed unit and regression testing are handled by L-Two for assembly and testing. The set of potential successful modules are thereafter delivered for integration and committed to L-three for deployment. Figure 3, shows project development progress as Paired Scrum processes.

3.1 L-One Developing Role

Pair programming of XP (Extreme programming) method is blended with the protocol of building the code by employing (QA and code vulnerability) a tester with each sprint group. The tester distributes his/her effort among sprint team developer as a pair programming. Being directly involved with the developers, he/she can highlight code vulnerabilities, and induce secure practices and overcome the social gab between developers and testers. He must be collaborating member in the L-two level.

Like the continuous integration of XP, the team performs a local (in the developing environment) unit test and regression testing, before committing the verified code. Yet, unlike XP, (in which the verified code is committed to the continuous integration mainline), the verified code is pushed into L-two for assembly and further testing. Of course, Regression testing is mandated whenever a change of the requirement takes place.

3.2 L-Two Developing Role

Level two adopts modularity with the protocol of assembling the source codes developed in L-one. In addition to automated unit tests, performed at L-one, a build server is utilized to implement *continuous* processes of applying quality control. The committed source codes are merged and assembled into their associate modules, test cases for each module are specified as test suits to account for the committed codes, and added to the testing suites pool.

In addition of running the unit and integration tests, this level run additional static & dynamic tests, measure and profile performance, extract and format documentation from the source code and, most important, reports binaries and meta data to repository artifact. This continuous application of quality control aims to improve the quality of software, and to reduce the time taken to deliver it. This is very similar to the original idea of integrating more frequently to make integration easier, only applied to QC processes.

Also, as most software uses a combination of open source and commercial code to supplement what the in-house development team builds. Keeping these supplementary components patched in the specified module is just as necessary as fixing issues in the developed code. The presented framework, promotes continuous testing and stepwise integration, rather than Continuous Integration. Step wise modules integration is performed whenever two or more modules are fully completed, in which case, modules are treated as black boxes, for which interfaces are tested for consistency, validation and type dependencies

verification. The completed Modules are packaged with its source files, binaries and metadata for delivery to L-three for deployment. Product manager can, however, choose to deliver uncompleted set of modules (which is the typical case in Agile methodology) as a prototype, to obtain an early customer's feedback and register a delivery.

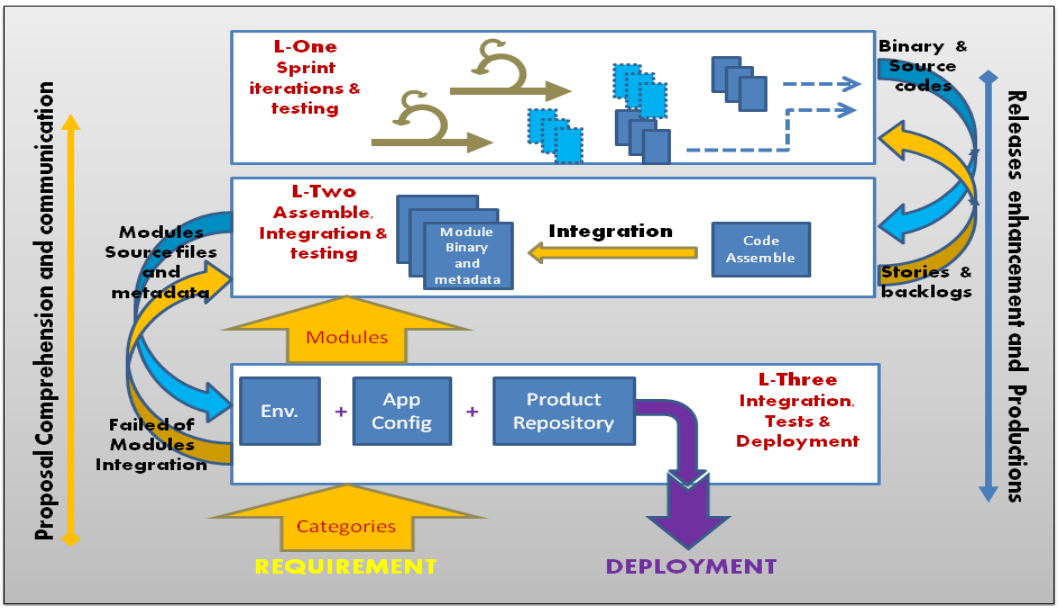

Figure 3: Paired Scrum Iterations (L-1→L-2 & L-2→L-3) for large projects

3.3 L-Three Developing And Deployment

A virtual machine or server platform is dedicated for the purpose of integration, testing and deployment. The dedicated machine is configured to emulate the production or user environment. Categories Configuration files are either prepared or developed for the to-be-delivered modules. Patches and fixes might be needed to tweak the environment or/and to select specific configuration files.

At this level groups of completed modules are integrated into their associate categories. Configuration of each category is tuned to account for all the delivered module components. Modules binaries are pulled for testing, while unfinished modules (belong to any of the current categories) are replaced by stumps.

Again all previous test suites are exercised beside modules integration testing and environment compatibility testing. Replacing dummy interfaces and stumps for unfinished modules, allows for exercising the completed modules in the assumed product package. Therefore, binaries and metadata for a successful release are available in the production line of the repository artifact and can be easily linked onto a new release to constitute a release package.

Stakeholders, BA, and release engineers may choose to carry on the development process vertically (depth wise) or horizontally (breadth wise). Vertical development process involves iterations for module completion and successive releases. The selected Modules' stories and backlogs are distributed for several parallel scrum sprints and being assembled, tested and delivered for deployment. L-one and L-tow are dedicated for completing all functionalities of one or more module, while L-three sets up the environment and fix applications configuration for production release.

Horizontal development process involves iterations of enhanced modules releases. Modules' stories and backlogs are distributed for several parallel sprint teams and being assembled, tested and delivered for deployment. L-one and L-tow are dedicated for developing enhancements of previous features of several modules, while L-three sets up the environment for version's release.

4. Three Level of Management

Project management is delegated to three interrelated actors in the development site, namely: Sprint management, Module management and Category management. The main focus is to, smoothly, build effective deliverable software that achieves customer proposal and comply with the production environment.

4.1 Sprint Management

Scrum makes the team responsible for managing their sprint. Scrum manager collaborate with the developing team to select a backlog to plan for the coming sprint. After that initial selection is made, it is up to the team to figure out how to do the work at hand. The team decides how to turn the selected requirements into an increment of potentially shippable product functionality. The team devises its own tasks and figures out who will do them. The selected backlogs are registered to the specific module in the issue tracker.

Issue tracker is basically a project management tool designed to integrate directly into the development process. In Agile development user stories are entered directly, broken down into features, and broken down again into specific developer tasks/assignments. Detected bugs also go into the issue tracker. This is the central tool for tracking the status of the development project — from earliest concepts to updates to production bugs.

Release Engineers is responsible for furnishing the Integrated Development Environment (IDE) where developers write code. It typically consists of a source code (text) editor, a compiler or an interpreter, a debugger, and other tools to help programmers write code and build applications.

Release Activities At The Sprint Level: Code management and version control is essential for the assembly and production of the intended (verified) version. Therefore metadata is generated for each completed piece of code. Source files Metadata, at sprint level completely define their names, versions, locations in the project (e.g. C1.G2.S1.A3.1) and its IDE i.e., compilers, testing tools, feed data, testing data set,…etc.), which will help in testing, assembling and integrating the developed code into modules and facilitates its deployment.

Release Engineers, with their integrating efforts, tracks the changes in the revision control system. They link the product components and push the finished features, into modules and releases.

4.2 Module Management

Project Managers, Release engineers, Quality control and scrum masters are deeply involved in managing Project modules. Scrum master make a place for the registered backlogs (decided at the sprint planning meeting) in the module, which is viewed as a subclass in the specified module. Thereafter, QC prepare the specific test suites (of the general test cases), while release Engineers collaborate with each module manager to define module's configurations. Also, as most software uses a combination of open source and commercial code to supplement what the in-house development team builds. Patching these supplementary components in the specified module is just as necessary as fixing issues in the developed code. Security patches in supporting commercial and open source platforms are incorporated into the sprint as tasks.

Version Control System will manage constantly changing code, which is mostly sets of many text files, which are worked on by the parallel sprint teams. A source code management tool [20, 21] keeps track of all changes and handles checkout, check-in, branching, forking, and otherwise keeping the code consistent and manageable. Whichever tool is used, the collection of code it manages is typically referred to as a *source code repository*.

Release activities at module level: Source files and configuration parameters are essential for module release and integration with other modules. The completed (finished and verified) set of source codes and the supplementary components patches are assembled and compiled into binary. A set up for the associated test cases is pulled for integration and static testing. Whilst, QC perform modules integration testing, automated fuzz testing [22] and acceptance testing. A successful module will have its metadata stored in the artifact repository; otherwise, a failed report should report the failed test cases. The metadata of source files at this level define all source files involved in the module, their version, including the supplementary patches, and module's configuration parameters. Figure 4, shows a typical release activity at the module level.

4.2.1 S-M Paired Scum Meeting

Module's teams are involved in workshops to diverse categories into modules and integrate new or changed requirement into the developing stream. They are responsible, also, for assembling sprint codes into those modules. During the sprint planning, sprint master registers the selected backlog (to be developed in the upcoming sprint). Therefore, Paired scrum meeting is due for Module team to prepare the required testing data set, the interface active parameters and fix stumps for any uncompleted task relevant to that specific module. As several sprints are running in parallel, Module's team prioritize the tasks and decide which integrations to be performed for the coming release.

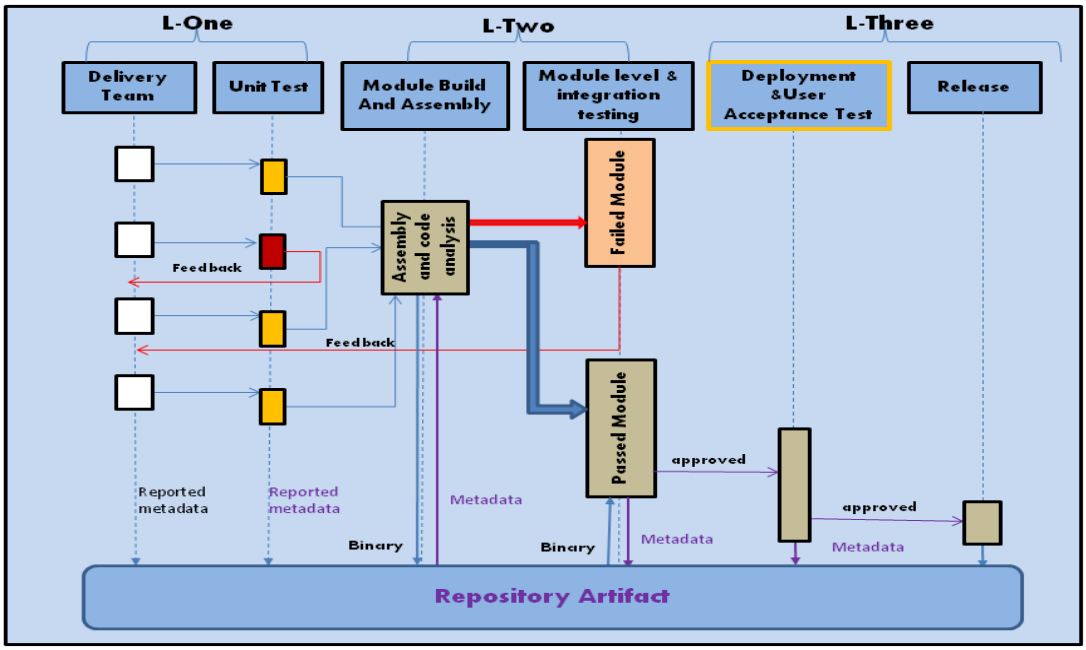

Figure 4: Development Progress stages and release activities.

4.3 Deployment Management

Deployment implies moving a product from a temporary or development state to a permanent or desired state. Deployment process consists of several interrelated activities with possible transitions between them. Deployment, in the proposed framework, however is a stepwise process that can be performed at each completion of a release. The completed release constitutes an upgraded version of a previous release. It consists of the set of approved module's files presenting either completed features or, merely, functionalities and tasks.

Manager's main concern is to emulate the production environment operation to comply with the customer environment. Therefore they initiate a Skeleton structure for the packaged product, refining it along the way as they gain better comprehension of the requirements. They recommend the application of fixes and patches for releases deployment. Tracking changes in the configuration management is a task performed by the release engineering. QC and operation personals manage the generation of acceptance, load/stress and performance tests to determine the sufficiency and capacity of the released software. Figure 5, illustrates the deployment process activities and the roles of project members.

4.3.1 M-C Paired Scum Meeting

Category Team is busy in furnishing a production virtual machine for the purpose of module integration, testing and deployment. An M-C Paired Scrum is due whenever Product manager and stakeholder choose to deliver a production release. Their agenda is to generate a check list for the preconditions, security and testing aspects essential for a successful releasing operation. At this level of abstraction requirement changes have no significant effects. Yet, this level is amenable for environment and security policy changes.

5. Related Work

In his paper, Sutherland [9] investigated the problem of globally distributed Agile teams and presented recommendations for the best practices of project management with outsourced teams. Zheng [11] applied Agile development methodology for implementing a business process management system. The author examined and confirmed Agile's effectiveness, and its positive impact on the organization. Koteska et. al, [14] proposed a concept of integrated agile software testing in a large scale project. Their research explained the software testing process technologies and principles for agile software testing and identified the best practices for testing software products in agile development process. Paasivaara et. al, [15] presented a case study of a successful use of agile practices in small distributed projects. They described how agile practices based on scrum had been applied to a mid-sized distributed software product development program. Cho [16] presented a hybrid model, in which he combined the Rational Unified Process with Scrum to maximize the strengths of both conventional and agile methods, while trying to suppressing the weakness of each approach. Lima et. al, [17] in their paper showed how Scrum agile software project management methodology has been deployed and adapted to the model of software project management of a research and development laboratory. Tomanek et. al, [18] executed a survey with the objective to understand how project managers manage risks in the agile software development projects. As a final result of this paper the authors proposed to extend the conceptual framework for managing agile software development projects, based on Scrum and PRINCE2, by risk management aspects.

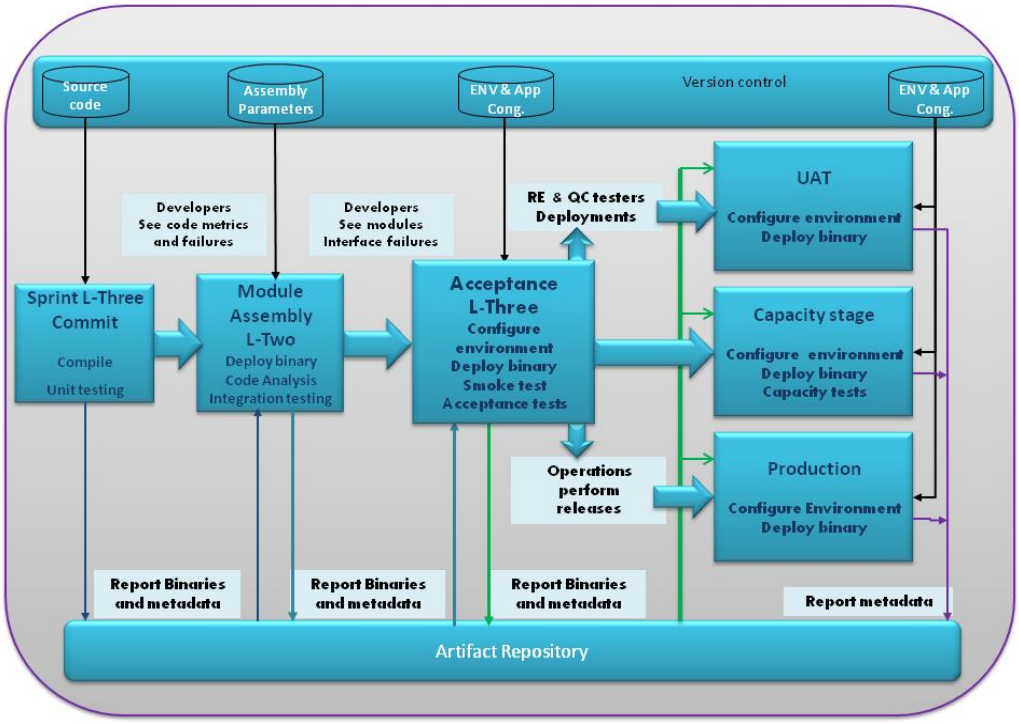

Figure 5: Deployment process activities

6. Evaluation and Discussion

The established framework proposes nested processes for product management and development. It employs the sprinting stage of Scrum approach in the inner nesting loop of two abstracted Paired Scrum process. At the outer perimeter is the first paired scrum process that delivers an abstraction of the requirement as set of modules. The delivered modules are devised into stories backlogs in the second Paired scrum process, to be the feeds for sprint developing team. This is constitutes the iterative top-down requirement analysis. Whilst, the Production development process progressed from the code pieces at L-One (bottom) up to assembly and integration (L-2 & L-3). By the end of the first sprint, the delivered source code is traversed backward into the Paired scrum process to be assembled into modules. The outer Paired scrum process accepts the traversed modules for integration and release.

The proposed frame work allows for both vertical and/or horizontal development cycle. This means that, it is scalable for small, medium as well as large projects. In Vertical development cycle, L-one develops a complete module functionalities that is assembled and Released at L-two, and may be deployed by L-three. Whereas, horizontal development cycle permits development of portions of several modules, which may be released as general prototypes of the involved modules.

Involving QC personals, at the early stages of development, as pair programming with the sprint team promotes production quality and induced security and vulnerabilities awareness among the developers. While Involving modularity with the protocol of assembling the generated code shifts the release process and refine software product into a structured architecture.

Automatic continuous testing allows for the early detection of bugs and faults, which can be mended faster while the code is still fresh in the developer memories. Stepwise integration blended with continuous testing permits the release of unfinished modules with compromising neither its security nor its efficiency.

The modularity of project components facilitates the generation of test cases up front, whilst the concrete test suits are generated step wised, parallel with the sprint cycles. This multilevel approach balances the work load for both QC and testers.

Pre-launching stage may consume several weeks, yet it is compensated for in many other project facets. The categories defined in this stage specify the Environment features and Configurations that provide for furnishing the production environment, which accelerated the releases and deployment processes. While, requirements modulation promoted collaboration and induced initial understanding of the requirements.

Project members are acting as Scrum development team with members involved in one or more level of the development process. Release Engineers have the big responsibility of lubricating and smoothing the transitions between the development levels.

The author claims that, skilled and experienced project staff is essential for a success implementation of the proposed framework, which may be considered as drawback of the framework.

7. Conclusion

Agile software development methods can provide a shorter development cycle, higher customer satisfaction, lower bug rates, and quicker adaptation to rapidly changing business requirements. Like the conventional method, the agile methods also have weaknesses, including 1) significant reduction of documentation and heavy dependency on tacit knowledge, 2) insufficient test for large-scale, mission-critical, and safety-critical projects, and 3) inadequacy for highly stable projects. This paper presents a new framework model for developing large project using Agile Scrum framework. The model blends top-down requirement modularity with Scrum methodology to maximize the strengths of conventional agile methods, while Integrating testing, providing flexibility for products releases and accelerating deployment. Modularity is utilized to provide a skeleton for the production architecture, while Scrum process is employed at the very primitive level of sprint production and Paired through a three level of the project development progress to offer project management. Bottom up developing process is achieved for module integration, product releases and deployment.

References

[1] Dyba, T., & Dingsoyr, T. Empirical studies of agile software development: A systematic review. Information and Software Technology, 2008; 50(9-10), 833-859.

[2] Muhammad Amir, Khalid Khan, Adnan Khan and M.N.A. Khan, "An Appraisal of Agile Software Development Process." International Journal of Advanced Science and Technology, 2013, vol.58, pp.75-86. http://dx.doi.org/10.14257/ijast.2013.58.07

[3] Cohn, M. "Succeeding with agile: software development using Scrum." Addison-Wesley Professional, 2009.

[4] Ken Schwaber, "Agile Project Management with Scrum." Microsoft Press, 2004. ISBN:073561993x

[5] Awad, M., "A Comparison between Agile and Traditional Software Development Methodologies." Master Thesis, Western Australia University; 2005.

[6] M. Steven Palmquist, Mary Ann Lapham, Suzanne Miller, Timothy Chick and Ipek Ozkaya, "Parallel Worlds: Agile and Waterfall: Differences and Similarities." Technical Note, Software Solutions Division, software engineering Institute, 2013.

[7] Naga Sri Morampudi, Gaurav Raj, "Evaluating Strengths and Weaknesses of Agile Scrum Framework using Knowledge Management." Inter. Journal of Computer Applications vol. 65, no.23, 2013, pp. 1-6.

[8] Peter Schuh, "Integrating Agile Development in the Real World", Charles River Media, 2005.

[9] Jeff Sutherland, "Distributed Scrum: Agile Project Management with Outsourced Development Teams." In Proc. of 40th Hawaii International Conference on System Sciences, *Petersburg, Russia*, 2007.

[10] Petersen, K. and Wohlin, C., "A Comparison of Issues And Advantages In Agile

And Incremental Development Between State Of The Art And An Industrial Case." Journal of Systems and Software, vol. 82(9), 2009, pp. 1479-1490.

[11] Gongyao Zheng, "Implementing A Business Process Management System Applying Agile Development Methodology: A Real-World Case Study." Bachelor Thesis, *Informatics & Economics*, Erasmus School of Economics of the Erasmus, Universities Rotterdam, May, 2012.

[12] Hung-Fu Chang and Stephen C-Y. Lu, "Toward the Integration of Traditional and Agile Approaches." International Journal of Advanced Computer Science and Applications, vol. 4, no. 2, 2013, pp. 9-13.

[13] Veracode "Secure Agile Development" edited by Chris Pepper, 2014. http://creativecommons.org/licenses/by-nc-nd/3.0/us/

[14] Bojana Koteska and Anastas Mishev, "Agile Software Testing Technologies in a Large Scale Project."

BCI'12, Novi Sad, Serbia, September 2012.

[15] Maria Paasivaara, Sandra Durasiewicz and Casper Lassenius, "Using Scrum In A Globally Distributed Project: A Case Study," Software Process: Improvement and Practice, special issue: Global Software Development: Where Are We Headed? vol. 13(6), 2008, pp. 527–544.

[16] Juyun Cho, "A Hybrid Software Development Method For Large-Scale Projects: Rational Unified Process With Scrum." Issues in Information Systems, vol. X, no. 2, 2009, pp. 340-348.

[17] Igor Ribeiro Lima, Tiago de Castro Freire, Heitor Augustus Xavier Costa, "Adapting and Using Scrum in a Software Research and Development Laboratory." Revista de Sistemas de Informação da FSMA,

no. 9, 2012, pp. 16-23. http://www.fsma.edu.br/si/sistemas.html

[18] Martin Tomanek and Jan Juricek , "Project Risk Management Model Based On Prince2 And Scrum Frameworks**."** International Journal of Software Engineering & Applications (IJSEA), vol.6, no. 1, 2015, pp. 81-88.

[19] L. Samuelis, Cs. Szabo, "On the role of the incrementality principle in software evolution." Egyptian Computer Science Journal, vol. 29, no. 2, 2007, pp. 107-112.

[20] Open Source version control systems. Available at:

http://www.smashingmagazine.com/2008/09/the-top-7-open-source-version-control-systems/

[21] Version Control software. Available at:

http://www.sitepoint.com/version-control-software-2014-what-options/

[22] D. Aitel. "The advantages of block-based protocol analysis for security testing,." 2002.

http://www.immunitysec.com/downloads/advantages of block based analysis.html.