Vol.6 Issue 1, Pg.: 20-26
January 2018

INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

A NEW APPROACH IN ASPECT- ORIENTED REQUIREMENT ENGINEERING FOR AGILE SOFTWARE DEVELOPMENT

Ardashir Shahmoradpour[1], Soheil Afraz[2]

1: Department of Computer Engineering, Ardabil Science and Research Branch, Islamic Azad University, Ardabil, IRAN
2: Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, IRAN.

Abstract: - Employing agile methods instead of traditional methods in software development is increasing strongly. Agile methods as a reaction to inherence problems of agile methods, promise new goals and values. Unlike traditional methods, agile methods are trying to establish an agile (not rigid) development process in software companies and organizations results in the company and customer satisfaction. Comprehensive Agile adoption is not an overnight and quick process because of their focus on individuals rather than processes. So far, there is no acceptable tool for assessing Agility of companies, therefore, this research proposed an Agility assessment model to evaluate the Agility of software companies based on universal metrics in aspect-oriented requirements engineering. Although employing a quantitative measure for evaluating a qualitative phenomenon is subject to criticisms, it is so helpful to improve and enhance the Agility degree of companies in agile software development process. In this research, agile practice instead of agile methods were used as the underpinning of the assessment model. Hence, the proposed model is usable in all the companies, even those who partially adapted to a specific agile method.

Keywords: Aspect-oriented, Agility, Agile Software Development, Agile Methods, Requirements Engineering

1. Introduction

Designing software-dominated systems to be adaptable to change has become increasingly critical when building user driven systems. While it is possible to define an initial set of requirements up front, adapting to user needs is necessary and often requires significant scope change. Incorporating multiple design paradigms into the system development process enables requirements to continue to change throughout the life of a system while allowing users to start taking advantage of existing functionality. A key insight is the benefit of applying the right paradigm given the stage of system development. Switching between paradigms during system development constitutes a novel hybrid approach to system engineering. This paper describes a case study where a hybrid approach clearly contributed to the achievement of challenging system development goals.

2. Literature Survey

Software dominates most new system development Software is critical for systems to remain competitive, but software dominated systems present a new set of system engineering challenges. Most recently, the debacle with healthcare.gov exemplifies the magnitude of the issues faced by most systems engineers of software systems. One major issue is the introduction of new requirements – a.k.a. requirements creep. However, they are a fact of system life. One way software system engineering deals this inevitability is spiral development, however the spiral methodology often leads to undocumented systems. The use of traditional methods results in systems that are well structured but don't allow room to grow. Conversely, the use of agile methods results in systems with room to grow but no well-defined path or end point. Reviewed 15 methods of requirements gathering and design development. Among them was Agile. The resulting development can provide products for customer use, but the software embodies no planning to accommodate future functionality and growth. They suggest development of the mission model and mission verification and validation to test the systems integration before the actual system development and propagation of issues that might originate at the component lever or user level. The fact of the matter is that software intensive systems are here to stay. Major problems can ensue during both requirements gathering and integration phases without the adoption of software-specific system engineering processes. For both phases, traditional and agile methodologies exist for addressing such problems, but our thesis is that a better solution comprises a hybrid approach. The Hybrid approach tends to help navigate requirements creep and systems integration issues not only by providing a flexibility to accommodate change, but also by incorporating structure and plans for future increments. In this paper, we present an application of the hybrid approach to a large-scale data processing and analysis system. Design and implementation of this system took place over two years using a mix of traditional and agile system engineering approaches.

3. Large scale data processing and analysis system case study

This detailed analysis looks at those improvement of a framework that began for those high-keyed objective from claiming taking done large portions divergent sorts of data, transforming What's more dissecting them, Furthermore giving work to them with clients for further examination. Same time those high-keyed objectives of the framework were situated in the recent past fill in on the framework started, those arrangement might have been to definition from claiming natty gritty objectives Furthermore necessities might have been to happen iteratively Similarly as the framework created. Those group creating the framework performed a large portion about this worth of effort. Framework improvement took two a considerable length of time.

$$\text{Agility} = \sum_{1}^{44} (P_i * W_i)/range$$

Three major turning points furnished those group chances will worth of effort straightforwardly for wind clients for critical sums about information accessible. All around those two a considerable length of time of framework development, those less group connected lessons learned, goals, Furthermore recognized client needs with determine the best framework building methodologies to utilize provided for the condition. The eagerness of the cooperation will alter with evolving condition permitted them to shift the middle of separate programming framework building methodologies likewise suitable to those periods of the system. The Emulating segments describe the advancement of the framework building methodologies as the less group balanced with an arrangement of lessons discovered and the framework developed.

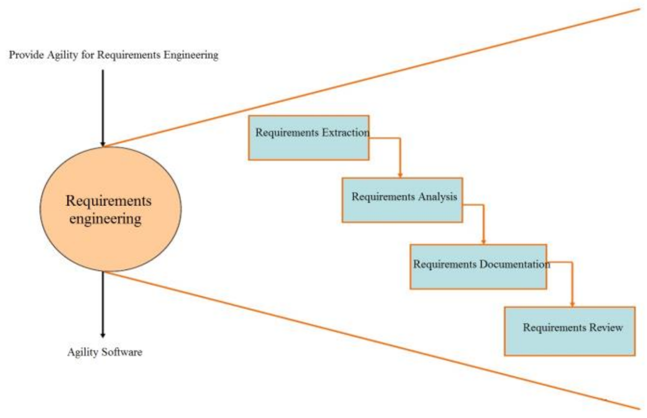
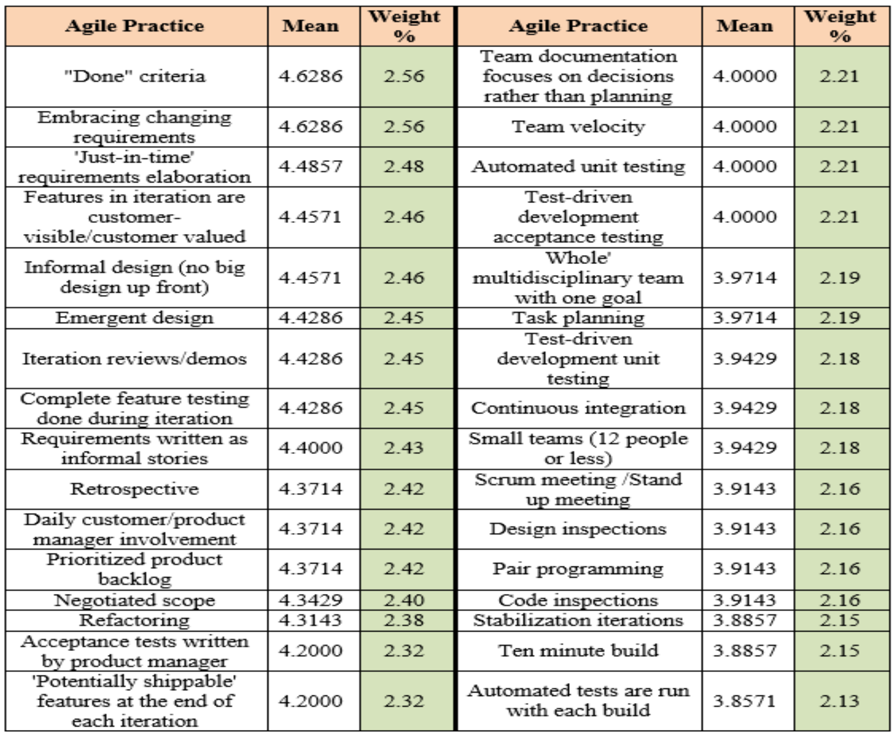


Figure 1. Our method diagram

4. Agile Project Management Methodology

Similarly as restricted of the universal methodologies, alarmed approach need been presented as a endeavor on aggravate product building adaptable and productive. With 94% of the associations working on alarmed in 2015[3], it need ended up a standard of venture administration. Those historical backdrop about alarmed might a chance to be followed again to 1957: In that the long haul Bernie Dims dale, john von Neumann, herb Jacobs, What's more Gerald Weinberg were utilizing incremental improvement systems (which need aid presently known as Agile), fabricating programming to IBM Furthermore Motorola. Although, not knowing how should arrange those approach they were practicing, they the greater part acknowledged obviously that it might have been unique in relation to those Waterfall from various perspectives [4]. However, the up to date alarmed methodology might have been authoritatively acquainted for 2017, when an aggregation from claiming 35 product improvement experts met on Talk elective venture administration methodologies.



Agile Practice	Mean	Weight %	Agile Practice	Mean	Weight %
"Done" criteria	4.6286	2.56	Team documentation focuses on decisions rather than planning	4.0000	2.21
Embracing changing requirements	4.6286	2.56	Team velocity	4.0000	2.21
'Just-in-time' requirements elaboration	4.4857	2.48	Automated unit testing	4.0000	2.21
Features in iteration are customer-visible/customer valued	4.4571	2.46	Test-driven development acceptance testing	4.0000	2.21
Informal design (no big design up front)	4.4571	2.46	Whole' multidisciplinary team with one goal	3.9714	2.19
Emergent design	4.4286	2.45	Task planning	3.9714	2.19
Iteration reviews/demos	4.4286	2.45	Test-driven development unit testing	3.9429	2.18
Complete feature testing done during iteration	4.4286	2.45	Continuous integration	3.9429	2.18
Requirements written as informal stories	4.4000	2.43	Small teams (12 people or less)	3.9429	2.18
Retrospective	4.3714	2.42	Scrum meeting /Stand up meeting	3.9143	2.16
Daily customer/product manager involvement	4.3714	2.42	Design inspections	3.9143	2.16
Prioritized product backlog	4.3714	2.42	Pair programming	3.9143	2.16
Negotiated scope	4.3429	2.40	Code inspections	3.9143	2.16
Refactoring	4.3143	2.38	Stabilization iterations	3.8857	2.15
Acceptance tests written by product manager	4.2000	2.32	Ten minute build	3.8857	2.15
'Potentially shippable' features at the end of each iteration	4.2000	2.32	Automated tests are run with each build	3.8571	2.13

Figure 2.Agile practices weight in agility assessment model

Hosting an acceptable dream of the flexible, lightweight Furthermore team-oriented product improvement approach, they mapped it out in those manifestly to alarmed product advancement. Pointed at "uncovering preferred routes of Creating software", the manifestly obviously specifies those key standards of the new approach: Through this fill in we have come to value: people and collaborations in techniques Also instruments working programming in far reaching documentation client coordinated effort In contract arrangement reacting to change over taking after an arrangement.

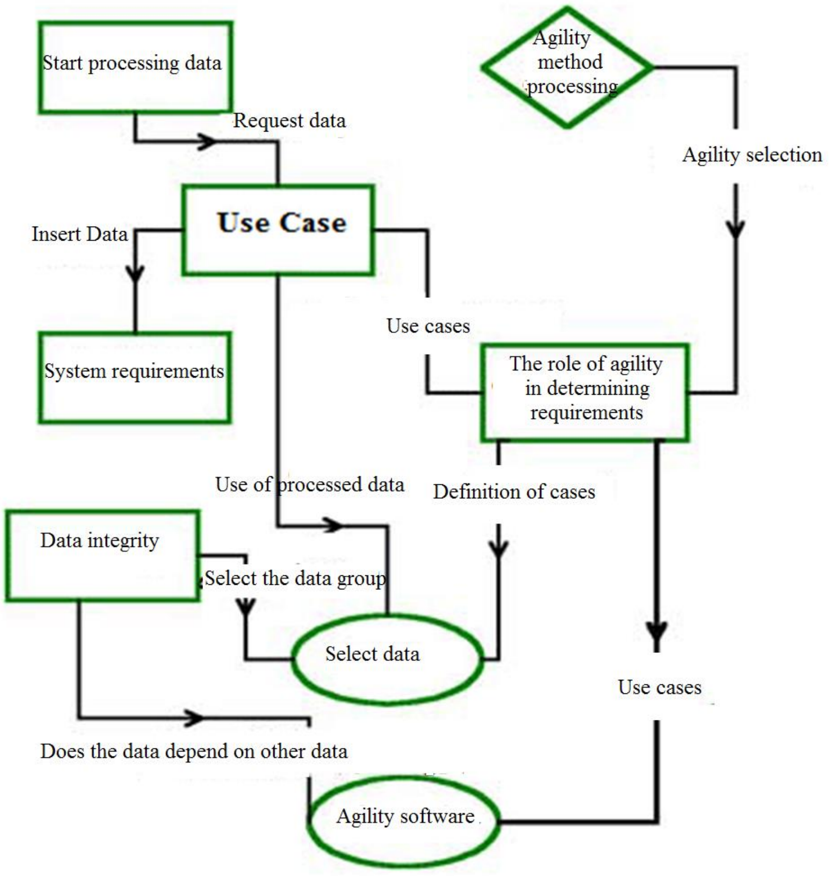


Figure 3.Agile Development Cycle

5. <u>Lessons learned and future guidelines</u>

The requisition of a mixture methodology of the design, development, verification, and acceptance of the framework in this detailed analysis permitted those frameworks with settle same time permitting for new prerequisites later over improvement. Eventually Tom's perusing looking after adaptability On paradigms, the less group might have been ready to adjust of the distinctive necessities Also tests from claiming every point of reference. Clinched alongside addition, those readiness of the less group on include new artifacts will help aide framework improvement enabled versatility similarly as concentrate moved towards accepting the framework. Same time those mixture approach functioned great In known to this system, there are a number about upgrades necessary on take more full focal point of the separate building paradigms. Same time there might have been a few level about client inclusion starting with the starting of the program, there might have been no committed set of clients until the third point of reference. This committed it was troublesome will establish a benchmark and focus Advance Throughout and the middle of turning points. To addition, the group didn't consider those effects a methodology required relative on framework soundness alternately the sway a methodology required looking into general improvement What's more joining pace.

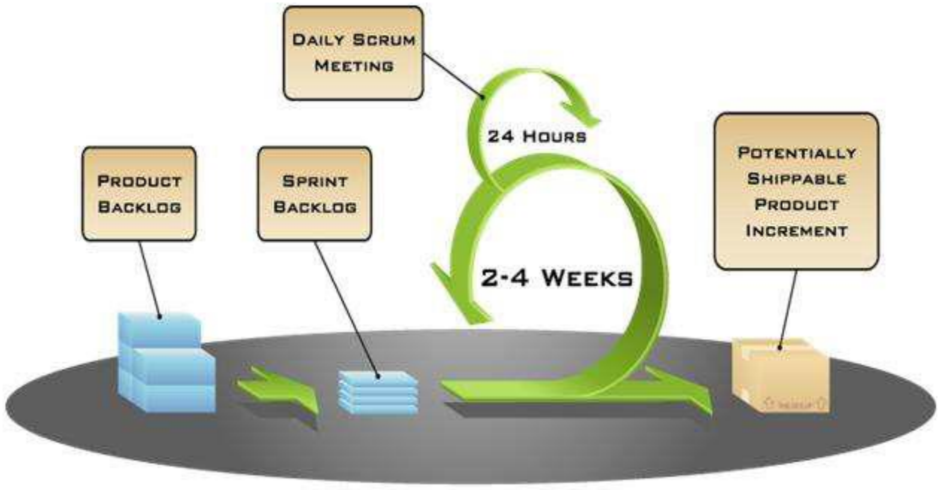


Figure 4.Scrum Framework

This got to be reasonable throughout breakthrough 3, The point when those less group produced more new features done a provided for sprint over Might realistically be tried. This prompted a large portions in length weekends Furthermore a decrease for Dependability of the benchmark. On recover, those cooperation included a mix just sprint the place they permitted no new Characteristics Furthermore set tighter controls set up will standardize those workload over sprints. So as on that's only the tip of the iceberg fully take advantage of a mixture approach, arranging during the starting from claiming framework plan ought happen will think through what amount of adaptability may be vital for those degree of the framework and introductory proposals to The point when over the methodology will movement paradigms. This arranging might lay out magic proposed move points, the thing that the extension of the framework is between each move point, what's more entryway quickly the framework will must alter to transform on meet deadlines. On do this effectively, groups ought to further bolstering to create entryways and measurements on to focus what achievement takes a gander in Furthermore The point when another approach may make suitable. A percentage measurement that the less group connected in this zone is: rate for joining relative will development, rate from claiming improvement relative should prerequisites and design, Also framework Strength Throughout coordination. These measurements the sum search for bottlenecks in the procedure that need aid initial indicators of the possibility compelling reason for An altered alternately totally distinctive approach, contingent upon the seriousness of the issue. This methodology might permit groups should respond both All the more rapidly Furthermore for All the more thinking ahead will evolve needs.
To determine the effectiveness of agile software, some of the factors are:

True Positive (TP): The total number of patterns that are properly recognized, namely, predicted, is agility to determine the requirements and it really has happened.

True Negative (TN): The number of predicted loads of agility will not affect the determination of requirements, and it does not really work.

False Positive (FP): The number of unpredictable loads that agility affects the determination of requirements, but it actually has an effect.

False Negative (FN): The number of loads expected, that is, agility does not affect the determination of requirements, but has not actually happened.

Vol.6 Issue 1, Pg.: 20-26
January 2018

Table 1. calculate performance

$Accuracy(\%)$	$Specificiy(\%)$	$Sensitivity(\%)$	Methods
90	89/9	92/3	Our Method
70/2	80	84/3	Agile software development methodology
71	76/4	83/6	User-centric software development

Those techniques about displaying another approach on building those requirements-oriented necessities for the improvement for alarmed product utilizing different calculations and models have brought about this hazard about utilization Also it might not a chance to be time permits to utilize one hundred percent of the calculations alone Also special case depended with respect to them. Acknowledging the vitality of the liable What's more attempting should minimize our extraction errors, we need took those examination produced on the systems accessible in this issue As far as accuracy, characteristic Also affectability and need aid demonstrated in the outline Similarly as indicated clinched alongside. Over essence, the reason for utilizing the new approach with engineer requirements-oriented prerequisites to that advancement from claiming alarmed product may be on gatherings give those suitable data for expansive databases, which infers that our suggested technique in this investigation is that's only the tip of the iceberg dependable over different routines.

6. Conclusion

In this chapter, we outlined our proposed approaches to the two issues listed in Chapter IV. In the approach the proposal for the first phase focuses on the utilization of the capabilities of engineering and soft requirements
Agile software, we have argued that by relying on these capabilities we can solve problems related to the change of concept and speed highly on the impact of engineering on the need to develop agile software development. This particular approach when the change of concept follows certain relationships, it is very good. The strong point of this approach is that if the relationships mentioned change dynamically over time the agent is able to adapt to new conditions in a short time and learn new relationships.

7. REFERENCES

[1] D. Cohen, M. Lindvall, and P. Costa, "An Introduction to Agile Methods," Advances in computers, vol. 22, pp. 1-22, , 2015.
[2] K. Beck, A. Cockburn, R. Jeffries, and J. Highsmith. (2001, July 2013). Agile manifesto. Available: http://www.agilemanifesto.org
[3] T. J. Gandomani, H. Zulzali, A. A. A. Ghani, A. M. Sultan, and M. Z. Nafchi, "Obstacles to moving to agile software development; at a glance," Journal of Computer Science, vol. 2, pp. 224-22., 2013.
[4] L. Williams, "What agile teams think of agile principles," Communications of the ACM, vol. .., pp. 41-42, 2014.
[5] M. Cohn, Succeeding with Agile: Software Development Using Scrum. Boston, MA: Addison-Wesley Professional, 2015.
[6] Chromatic, Extreme Programming Pocket Guide. Sebastopol, CA: O'Reilly Media, 2013.
[7] J. A. H.ghsmith, Agile Software Development Ecosystems. Boston, MA: Addison- Wesley Professional, 2012.

[8] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," Computer, vol. 34, pp. 124-122, 2013.
[9] R. W.W., "Managing the development of large software systems: Concepts and techniques," in WESCON, 2014, pp. 1-2.
[10] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, 2nd ed. Boston, MA: Addison-Wesley Professional, 2016.
[11] B. Boehm, "Get ready for agile methods, with care," Computer, vol. 3. pp. 24-22, 2013.
[12] B. W. Boehm, "Spiral model of software development and enhancement," Computer, vol. 21, pp. 21-42, 2016.
[13] R. L. Glass, "Agile versus traditional: Make love, not war," Cutter IT Journal vol. December, pp. 12-1., 2013.
[14] A. Cockburn and J. Highsmith, "Agile software development: The people factor," Computer, vol. 34, pp. 131-133, 2012.
[15] J. Coldewey, "Agile software development - Introduction and overview," Agile Entwicklung Web-basierter Systeme: Eiführung und Überblick, vol. 44, pp. 234-24., 2016.
[16] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review”, Information and Software Technology, vol. 50, no. 9–10, pp. 833–859, 2017.
[17] G. Hanssen, D. Šmite, and N. B. Moe, „Signs of Agile Trends in Global Software Engineering Research: A Tertiary Study“, IEEE Sixth International Conference on Global Software Engineering Workshop, pp. 17–23, 2016.
[18] B.A. Kitchenham and S. Charters, “Procedures for Performing Systematic Literature Review in Software Engineering,” EBSE Technical Report version 2.3, EBSE-2007-01, Software Eng. Group.
[19] B.A. Kitchenham, R. Pretorius, D. Budgen, P. Brereton, M. Turner, M. Niazi, and S. Linkman, “Systematic literature reviews in software engineering – A tertiary study”, Information and Software Technology, vol. 52, no. 8, pp. 792–805, 2017.