Balancing Agility and Discipline for Using Hybrid Methodologies in Large-Scale Software Projects BALANCING AGILITY AND DISCIPLINE FOR USING HYBRID METHODOLOGIES IN LARGE-SCALE SOFT…

Conference Paper · June 2018

CITATIONS
0

READS
87

3 authors, including:

Maryam Yosefi

3 PUBLICATIONS **1** CITATION

SEE PROFILE

Seyed Mohammadbagher Jafari

University of Tehran

89 PUBLICATIONS **521** CITATIONS

SEE PROFILE

BALANCING AGILITY AND DISCIPLINE FOR USING HYBRID METHODOLOGIES IN LARGE-SCALE SOFTWARE PROJECTS

[1]MARYAM YOSEFI, [2]SEYED MOHAMMADBAGHER JAFARI*, [3]SARA NAZIFIFARD

[1]Taali Institute of Higher Education, Iran, [2,3]Faculty of Management & Accounting, University of Tehran, Iran
Email: [1]maryam.yosefi93@gmail.com, [2]sm.jafari@ut.ac.ir (*Corresponding Author), [3]s.nazifi@ut.ac.ir

Abstract: Hybrid methodologies can be applied in the software industry, particularly, in business sectors to cope with dilemmas in large-scale software projects. Rapid change and increasing software criticality drive organizations to balance between agility and discipline in software development projects. In order to utilize the benefits of both approaches in large-scale software development projects, integration of agile and non-agile development methodologies via a hybrid methodology seems to be necessary. In this research, firstly a comprehensive comparison of two methods has been done. Then, the characteristics and differences of both traditional and agile methods have been analyzed. The most important agile and discipline factors have been introduced for effective evaluation of large-scale software development projects. This will provide clear insight of project circumstances and help software developers to increase the degree of agility and discipline of the project.

Index terms: Large-scale software projects, Agility, Discipline, Hybrid methodologies

I. INTRODUCTION

Disciplined development methods such as Waterfall, V model, and Rational Unified Process (RUP) have been prevailing in software industry since mid1990s, but these methods are mainly focused on extensive planning and documentation and team expertise which is only appropriate for medium and large-scale projects. With the gradual passage of time, the disciplined software development methods have been embedded in agile software development methodologies. The disciplined methods are discussed in several articles [1-4]. Over the last decade, agile methods have been in the notice of the software industry due to their unequaled features, such as quick response to changing requirement, pragmatic documentation and agility [2, 5]. The important characteristics of agile methods are the incremental development style, a cooperative development, straightforward and adaptive [2]. Actually, structured methods and agile methods are compatible rather than being combinable [6]. Large-scale software development is a radically increasing phenomenon in modern software development environments. At the same time, traditional and agile methodologies and combinations of those are being used in the industry. Despite the existing deficiencies, the traditional methods are still commonly used in industry, particularly, for large-scale projects[7]. The driving force of this wide utilization of the traditional methods comes from their straightforward, methodical, and structured nature [8], also their capability to provide predictability, stability, and high assurance [9]. In spite of the potential benefits of the agile methods, many organizations are reluctant to give up their traditional methods and switch to agile methods due to several issues. Based on the literature review applying agile methods in large-scale is a challenging issue [7, 10-12]. One solution for this issue is creation of a new hybrid method by combining traditional and agile methods[7, 13]. Hybrid methodologies usage is associated with specific features of agile and traditional methods in software development projects, mainly in large-scale projects [10]. There is a lack of research regarding the reasoned to use hybrid methodologies [7, 14-16]. In order to fully understand why hybrid methodologies should be used, more research is required [14]. To this aim, this research provides an extensive comparison of both agile and structured methods, which is presented based on project and method characteristics. Applying hybrid methodologies in large-scale projects is in relation to the compatibility of principles of agility and discipline. This research introduces associated factors and key dimensions affecting method selection in large-scale software projects.

A. Comparing traditional methods with agile methods

The disciplined methods were prevalent in the software industry until the mid1990s. Since then, the traditional methods have been replaced by agile software development methods mostly in small-scale and proportionally simple projects. This phenomenon is mainly due to the structured methods' shortcomings, including a slow adaptation to rapidly changing business requirements, and inclination to be over budget and behind schedule [17-19]. The structured methods also have failed to provide drastic improvements in productivity, reliability, and simplicity [7, 20]. The main differences between these two methodologies are shown in table 1.

Table 1-Comparison of disciplined and agile methodologies

Project/ methodology properties	Agile methodologies	Disciplined methodologies	Sources
Fundamental hypothesis	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change	Systems are fully specifiable, predictable and can be built through meticulous and extensive planning	[16, 17, 21]
Project size	Small/ medium	Large	[10, 17, 22-27]
Requirements	Largely emergent, with rapid changes	Very stable, known in advance	[16, 17, 21]
Team size	Small, usually less than 10	Large, more than 10	[17, 24, 25]
Mindset and emphasis	Innovative, people oriented	Disciplined, process-oriented	[23, 25]
Project management mode	Decentralized	Autocratic	[25]
Documentation	Low	Heavy, light and abstract	[22, 25, 28]
Planning and control	Limited, internalized plans; qualitative control	Comprehensive upfront planning, documented, process-oriented, quantitative control	[10, 23, 29-31]
Development cycle	Unlimited, numerous iteration	Limited tied and bound	[10, 27]
Return on investment	Early stage	At the end of the project	[10, 27]
Technology	Favors object-oriented technology	No restriction	[21]
Success measurement	Business value	Conformation to plan	[27, 28]
Architecture	Design for current and predictable requirement	Design for current requirements	[17, 22, 27, 28]
Project cycle	Guided by tasks or activities	Guided by product features	[17, 21]
Domain	Unpredictable/ exploratory	Predictable	[24]
Primary Goals	Rapid value, responding to change	Predictability, stability, high assurance, high safety	[17, 22, 27, 28]
Quality control	Permanent control or requirements, design and solutions	Difficult planning and strict control	[16, 30]
Team and Role assignment	Self-organizing teams encourages role interchangeability	Individual-favors specialization	[16, 17, 21, 22, 27, 31]
Testing	Every iteration, permanent testing, executable test cases define requirements and testing	After coding is completed, difficult and late testing, documented test plans and procedures	[10, 16, 30]
Development direction	Easily changeable	Fixed	[10, 16]
Feedback mechanism	Easily accessible	Not easily accessible	[21]
Risk	Unknown risk, major impact	Well understood risk, minor impact	[10, 22]
Continuous learning	Embraced	Not frequently encouraged	[31]
Development model	The evolutionary-delivery model	Life cycle model (waterfall, spiral or some variation)	[21, 28, 30, 31]
Fault detection	Easy	Difficult	[10]
Reusability	Easily accessible	Hardly accessible	[10]
Design flexibility	Easily accessible	Not Easily accessible	[10]
Cost of restart /refactoring/ remodeling	Not expensive	Expensive	[10, 16, 17, 22, 27]
Communication	Informal	Formal	[16, 21, 30]
Perspective to change	Change adaptability	Change sustainability	[25, 27]

Project/ methodology properties	Agile methodologies	Disciplined methodologies	Sources
Organizational culture	Organic (flexible and participative, encourages social cooperation), targeting small and medium organizations	Mechanic (bureaucratic, high formalization), targeting large organizations, command and control	[16, 21, 25, 28, 29]
Approach	Adaptive	Predictive	[21, 27, 30, 31]
Customer requirements	Interactive input	Detailed and defined before coding/ implementation	[10, 16, 17, 22, 26, 28]
Customer collaboration and satisfaction	Dedicated on-site customers; focused on prioritized increments, high	As needed customer interaction, focused on contract provision, low	[10, 16, 25]
Additional abilities required from developers	Interpersonal abilities and basic knowledge of the business	Nothing in particular	[10, 16]
Developers	Agile, knowledgeable, collocated, and collaborative	Plan oriented, adequate skills, access to external knowledge, pre-structured team	[10, 16, 17, 22, 26]
Customer	Dedicated, co-located collaborative, representative, authorized, committed, knowledgeable	Minimal commitment, not co-located and empowered	[10, 16, 17, 22, 26]
Knowledge management	Explicit	Tacit	[10, 30, 31]
Team physical location	Predominantly collocated	Predominantly distributed	[28, 31]
Client involvement	High, Critical and continuous	Low, Important, usually only at the analysis phase of the project	[21, 28, 31]
Team involvement	Essential	Not essential	[31]
Management style	Leadership-and-collaboration, decentralized	Command and control, autocratic	[21, 22, 27, 31]

B. Hybrid approaches in large-scale software projects

While the majority of software developers claim agility method is desirable by the developers, there are also those who enjoy working within hybrid methodologies, claiming it gives them increased control over the project and provide both flexibility and rigidity [7, 14, 16]. Hybrid solutions are often more successful than other methods in large-scale software projects [13, 32-36]. By introducing traditional/agile hybrid methods, large organizations can take advantage of some benefits of agile development without abandoning the discipline provided by traditional methods [7, 10, 11, 15].

II. DISCIPLINE AND AGILITY FACTORS FOR EVALUATION OF LARGE-SCALE SOFTWARE DEVELOPMENT PROJECTS

Discipline is the basis for all profitable attempts. Discipline creates well-organized history, and experience versus agility is equivalent to discipline. Where discipline ingrains and reinforces, agility releases and discovers. Agility applies history to rectify new environments to react and adapt, due to take advantage of unexpected opportunities[37]. Although agile methodologies are broadly accepted in software development. But it's not suitable for large client based projects. New hybrid agile methods which combine the philosophy of waterfall, iterative waterfall, prototype, spiral as well as pure agile is recommended for large-scale software projects [38]. Using hybrid methodologies might be due to an increased stream of agility adoption[7, 14, 15]. These issues and related factors can be categorized in four key dimensions namely project, dynamism, organizational culture and process. Following sections will provide extensive knowledge.

A. Project

Applying only agile methods are not suitable for large-scale. Agile methods significantly reduce the amount of documentation, also have not been sufficiently tested for mission/safety-critical projects [7]. As project size, complexity and number of developers' increase, the less agility is applicable. In fact, with highly volatile environment it is better to take advantages of agile merit. If these situations coexist providing a hybrid approach seems better to manage the situation [9, 11, 33, 34, 39]. By rating project's environmental, agile, and plan-driven risks, developers choose an appropriate method to mitigate the overall project risk during development [9, 10, 34, 40]. Factors correlate agility and discipline in project dimension summarized in table 2.

Table 2-Project

Dimension	Factors associated with applying hybrid methods	Source
Project	Project size (small, large, meduim) & complexity	[9-11, 41]
	Project objectives & priorities	[10, 11]
	Criticality & risk	[6, 9, 10, 16, 33, 34, 39, 42]
	Project type (new, renovation, upgrade)	[11]
	Project duration	[10, 41]
	Business model	[15, 43]
	Number of developers and teams	[9, 42]
	Scope	[11, 41, 43]
	Nature of project interdependencies (sequential, reciprcal)	[10, 16]
	Budget and schedule controling& project tracking	[41, 43]

Table 3- Process

Dimension	Factors associated with using hybrid methodologies	Source
Process	Agile& non-agile Requirement engineering teqniques	[5, 6, 15, 16, 41, 43]
	Requirement documentation	[5, 6, 16]
	Requirement tracking	[5, 6, 41]
	Testing practices and tools (agile& non-agile)	[12, 15, 41, 45]
	Defect tracking	[12, 41]
	Outsourcing	[10]
	Sufficient amount of documentation	[6, 10, 15, 43, 44, 46]
	Up-front & incremental designing	[10, 15, 44]
	Techniques (agile & none-agile)	[15, 41]
	Reuse with refactoring	[10]

B. Process

Using an incremental life-cycle model is critical because the source of many conflicts observed are in charge of the all up-front thinking that comes with the single-increment waterfall model. Incremental thinking is fundamental to the coexistence of the two agile and structured methods especially in large-scale [44]. Agile practices require full integration of testing and development, which can change high-level product-release policies and organizational structures. In traditional methods, once a feature-complete version is sent to the quality assurance group, it sends back a vast stream of defects of varying importance. The team restores the most urgent defects first, agile methods improve this process with fixing every defect as soon as it has been discovered at every iteration, this is because the team should releases software to its customers, and releases shouldn't, in principle, contain known bugs [12]. Factors in relation to process dimension are summarized below in table 3.

C. Organizational culture

Organizational culture is the main dimension in agility and discipline factors [47, 48]. Organizational culture aspect of agility in large organizations is the emulation of different organizational values and cultures related to agile culture and agile values [49]. Certain conditions extend to make compatible the agility and structured approach in software development projects [6, 9, 11, 33, 34, 39]. The project leaders are unanimous that the projects are likely to confront significant changes in the user requirements [10]. Organizational culture factors summarized below in table 4 are as balancing factors of agility and discipline in large-scale projects.

Table 4-Organizational culture

Dimension	Factors associated with using hybrid methods	Source
Organizational culture	Team size (small, large)	[10, 41]
	Developer & customer collaboration	[10, 44, 46]
	Team distribution (co-located & distributed teams)	[11, 41, 50]
	Agile Competency	[15, 42, 46]
	Motivated trusted individuals	[10, 46]
	Identifying different stakeholders	[50]
	Welcome changing requirements	[10]
	Self-organizing teams	[46]
	Identifying & managing developers	[11]
	Knowledge management support	[16]
	Sufficient face-to-face meeting	[10, 46]

D. Dynamism

Detailed plans and big design upfront best suited for stable environments, but the simple design and continuous refactoring are excellent for extremely changing environments. Agile methods can be successful only with talented individuals who favor many degrees of freedom. Applying flexible and sufficient extent of pair programming and flatter hierarchies with controlled delegation are suitable for large-scale projects [11]. Ability to cope with existing and emerging technical challenges such as evolving Internet and Web capabilities, distributed and mobile operations, agent coordination, and multimode virtual collaboration. Large organizations should complement whatever agile or discipline methods balancing options they chase with continuous effort to improve staff competency, and communication capabilities [9]. Dynamism factors are illustrated in table 5.

Table 5-Dynamism

Dimension	Factors associated with using hybrid methodologies	Source
Dynamism	Changes in top management	[9, 43]
	Technolgy changes & organization unanticipation	[42]
	Requirements inclination to change during the development process	[42, 43, 46]
	Managing changing priorities	[41, 44]
	Short Internal & external release with layered approach	[10, 46]
	Agile maturity	[41, 44]
	Volatility of environment	[10]
	Flexible pair programing	[10]
	Moderate hierarchical structure with controlled empowerment	[10]

III. CONCLUSION

To sum up, hybrid methodologies have increasingly been utilized in relatively large-scale projects. Neither agile nor the disciplined approaches provide the best approach. Indeed, the two seem to contradict but several researchers stated that large-scale software projects need both agility and discipline. Both approaches are sufficiently compatible to utilize advantageously their benefits in such projects. In order to fully understand why and when hybrid methodologies should be used, current study provided a comprehensive comparison between disciplined and agile methods. Key dimensions and factors affecting the combination of principles of agility and discipline also has been introduced, which is the reason for hybrid methodologies usage in large-scale projects.

REFERENCES

[1] N.M.A. Munassar, A. Govardhan, A comparison between five models of software engineering, IJCSI, 2010, 5, pp. 95-101.

[2] G. Rasool , S. Aftab, S. Hussain, D. Streitferdt, eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects, Journal of Software Engineering and Applications, 2013, 6, p. 446.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, J. Warsta, Agile software development methods: Review and analysis, in, VTT Finland, 2002.

[4] D. Truex, R. Baskerville, J. Travis, Amethodical systems development: the deferred meaning of systems development

methods, Accounting, management and information technologies, 2000, 10, pp. 53-79.

[5] N. Ganesh, S. Thangasamy, Issues identified in the software process due to barriers found during eliciting requirements on agile software projects: Insights from India, 2011, 16.

[6] L.T. Heeager, The Agile and the Disciplined Software Approaches: Combinable or Just Compatible?, in: Information systems development, Springer, 2013, pp. 35-49.

[7] J. Cho, A hybrid software development method for large-scale projects: rational unified process with scrum, Issues in Information Systems, 2009, 10, pp. 340-348.

[8] A. Fruhling, G.-J.D. Vreede, Field experiences with eXtreme programming: developing an emergency response system, Journal of Management Information Systems, 2006, 22, pp. 39-68.

[9] B. Boehm, R. Turner, Observations on balancing discipline and agility, in: Agile Development Conference, 2003. ADC 2003. Proceedings of the, IEEE, 2003, pp. 32-39.

[10] J.B. Barlow, J. Giboney, M.J. Keith, D. Wilson, R. Schuetzler, P.B. Lowry, A. Vance, Overview and guidance on agile development in large organizations, Communications of the Association for Information Systems, 2011, 29, pp. 25-44.

[11] D. Batra, W. Xia, D. VanderMeer, K. Dutta, Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry, Communications of the Association for Information Systems, 2010, 27, p. 21.

[12] D. Talby, A. Keren, O. Hazzan, Y. Dubinsky, Agile software testing in a large-scale project, IEEE software, 2006, 23, pp. 30-37.

[13] A. Mandal, S.C. Pal, Achieving agility through BRIDGE process model: an approach to integrate the agile and disciplined software development, Innovations in Systems and Software Engineering, 2015, 11, pp. 1-7.

[14] E. Burman, Agile in action: Hybrid methodologies in practice, 2015.

[15] A.Q. Gill, B. Henderson-Sellers, M. Niazi, Scaling for agility: A reference model for hybrid traditional-agile software development methodologies, Information Systems Frontiers, 2016, pp. 1-27.

[16] M. Stoica, M. Mircea, B. Ghilic-Micu, Software Development: Agile vs. Traditional, Informatica Economica, 2013, 17, p. 64.

[17] B. Boehm, Get ready for agile methods, with care, Computer, 2002, 35, pp. 64-69.

[18] G. Washington, Using risk to balance agile and plan-driven methods, IEE Computer Science, 2003.

[19] P. Grünbacher, M. Halling, S. Biffl, S. BIFFL, B.W. Boehm, Integrating collaborative processes and quality assurance techniques: experiences from requirements negotiation, Journal of Management Information Systems, 2004, 20, pp. 10-30.

[20] F.P. Brooks Jr, The mythical man-month (anniversary ed.), 1995.

[21] S. Nerur, R. Mahapatra, G. Mangalaraj, Challenges of migrating to agile methodologies, Communications of the ACM, 2005, 48, pp. 72-78.

[22] A. Aitken, V. Ilango, A comparative analysis of traditional software engineering and agile software development, in: System Sciences (HICSS), 2013 46th Hawaii International Conference on, IEEE, 2013, pp. 4751-4760.

[23] T.J. Gandomani, H. Zulzalil, A.A.A. Ghani, A.B.M. Sultan, M.Z. Nafchi, Obstacles in moving to agile software development methods; at a glance, Journal of Computer Science, 2013, 9, p. 620.

[24] S.C. Misra, Adopting agile software development practices: success factors, changes required, and challenges, in, Carleton University Ottawa, 2007.

[25] M. Awad, A comparison between agile and traditional software development methodologies, University of Western Australia, 2005.

[26] G. Aslam, F. Farooq, A comparative study on Traditional Software Development Methods and Agile Software Development Methods, 2011.

[27] G. Kumar, P.K. Bhatia, Comparative analysis of software engineering models from traditional to modern methodologies, in: 2014 Fourth International Conference on Advanced Computing & Communication Technologies, IEEE, 2014, pp. 189-196.

[28] A. Moniruzzaman, D.S.A. Hossain, Comparative Study on Agile software development methodologies, arXiv preprint arXiv:1307.3356, 2013.

[29] B. Fitzgerald, An empirical investigation into the adoption of systems development methodologies, Information & Management, 1998, 34, pp. 317-328.

[30] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and software technology, 2008, 50, pp. 833-859.

[31] K. Conboy, S. Coyle, X. Wang, M. Pikkarainen, People over process: key people challenges in agile development, 2011.

[36] M. Rahmanian, A Comparative Study on Hybrid IT Project Managment, International Journal of Computer and Information Technology, 2014, 3.

[39] B. Boehm, D. Port, A.W. Brown, Balancing plan-driven and agile methods in software engineering project courses, Computer Science Education, 2002, 12, pp. 187-195.

[40] S.F. Eng, Agility and Discipline, 2014.

[41] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices, Empirical Software Engineering, 2010, 15, pp. 654-693.

[42] V. Vinekar, C.W. Slinkman, S. Nerur, Can agile and traditional systems development approaches coexist? An ambidextrous view, Information systems management, 2006, 23, pp. 31-42.

[43] B. Boehm, R. Turner, Management challenges to implementing agile processes in traditional development organizations, IEEE software, 2005, 22, pp. 30-39.

[44] P. McMahon, Bridging agile and traditional development methods: a project management perspective, CrossTalk: The Journal of Defense Software Engineering (May 2004), 2004.

[45] C. Gil, J. Diaz, M. Orozco, A. De La Hoz, E. De La Hoz, R. Morales, Agile Testing Practices in Software Quality: State of the Art Review, Journal of Theoretical and Applied Information Technology, 2016, 92, p. 28.

[46] L. Barroca, H. Sharp, D. Salah, K. Taylor, P. Gregory, Bridging the gap between research and agile practice: an evolutionary model, International Journal of System Assurance Engineering and Management, 2015, pp. 1-12.

[47] G. Mikulėnas, R. Butleris, An approach for constructing evaluation model of suitability assessment of agile methods using analytic hierarchy process, Elektronika ir Elektrotechnika, 2015, 106, pp. 99-104.

[48] A. Debate, Agility through, 2003.

[49] M. Laanti, Characteristics and Principles of Scaled Agile, in: international conference on agile software development, Springer international publishing, 2014.

[50] N. Azizi, M.A. Taqi, Applying Agile methodologies within the context of traditional project governance:-A study of the Volvo Group experience, 2015.