Indian Journal of Science and Technology, Vol 9(28), DOI: 10.17485/ijst/2016/v9i28/97806, July 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Coordination in Different Software Development Lifecycles: A Systematic Review

W. A. W. M. Sobri[1], S. S. M. Fauzi[1*], M. H. N. M. Nasir[2], R. Ahmad[2] and A. J. Suali[1]

[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia; shukorsanim@perlis.uitm.edu.my
[2]Faculty of Computer Science and Information Technology, University of Malaya, Malaysia.

Abstract

Coordination plays a significant role in Software Development Lifecycle (SDLC) models. However, coordination varies between SDLC models. Relatively, little research has been conducted to analyze coordination in different SDLC models. This paper presents a Systematic Literature Review (SLR) of coordination in different SDLC models. Among the range of SDLC models, the most important and popular ones are the waterfall, spiral, agile, RAD, V and incremental models, which have been investigated on how coordination works in these models.

Keywords: Coordination, Software Development, Software Engineering Project, Software Project

1. Introduction

The use of coordination in software projects has already successfully rendered projects of good quality. Coordination is defined as: "*when multiple actors pursue goals together, they have to do things to organize themselves that a single actor pursuing the same goals would not have to do. We call these extra organizing activities coordination*"[1]. In this research, coordination is examined through the coordination between software developers during the software development stage in software engineering projects.

A Software Development Lifecycle (SDLC) model can be described as a process which begins at the phase of feasibility analysis right up to maintenance[2]. There are various SDLC models examined throughout the software development process. The most important and popular models are the waterfall model, spiral model, agile model, Rapid Application Development (RAD) model, V-model and incremental model.

As a matter of fact, different SDLC models have their own way in developing software development projects, therefore, coordination along the development period will differ between each SDLC model.

Table 1. Keywords used in this study

Category	Keywords
Coordination	Software developer coordination Developer coordination Coordination Software developer co-operation Developer co-operation Co-operation Software developer collaboration Developer collaboration Collaboration
Model	Waterfall model Spiral model Agile model Rapid Application Development (RAD) model V-model
Development Environment	Software Development Software Project Software Engineering Project

This paper explores how coordination takes place in different SDLC models. Despite a plethora of SDLC models available nowadays, this paper discusses only the popular and vital models. These include the waterfall, spiral, agile, RAD, V and incremental models.

*Author for correspondence

The paper has been structured as follows. The method is described in the first section. The next section presents results of the study, followed by the discussion section on coordination that occurs in selected software development lifecycle models. The final section infers the conclusions for this paper.

2. Method

2.1 Research Questions

With respect to providing a more comprehensive review on the research stated, the following research question is addressed:

RQ1: *How does coordination occur in different Software Development Lifecycle (SDLC) models?*

2.2 Searching and Keywording Criteria

The database search engines used in this study are:

- Emerald Insight (http://www.emeraldinsight.com/).
- ResearchGate (http://www.researchgate.net).
- Google Scholar (https://scholar.google.com.sg/).
- Springer (http://www.springer.com/gp/).
- IEEE Xplore (http://ieeexplore.ieee.org).
- ACM Digital Library (http://dl.acm.org/dl.cfm).
- Elsevier Science Direct (http://www.sciencedirect.com/).
- Wiley Online Library (http://onlinelibrary.wiley.com/).

The keywords used in this study are listed in Table 1 below.

2.3 Screening of Papers

Next, a previous study conducted in regard to the research studied was screened thoroughly. Inclusion and exclusion criteria were included.

The criteria for Inclusion (I) and Exclusion (E) are as follows:

I1. The related papers were issued with well-defined research questions, search and keyword criteria.

E1. Abstracts, organization studies, articles, presentations.

The papers were screened step by step. Firstly, the researcher scrutinized the title, abstract, introduction as well as content of the papers, because the keywords associated with this study may not have been at the beginning of the paper. Of all the 80 papers screened by title, 46 papers were extracted from abstract screening and 46 papers by screening the introduction and content. The final inclusion papers stood at a total of 31 papers.

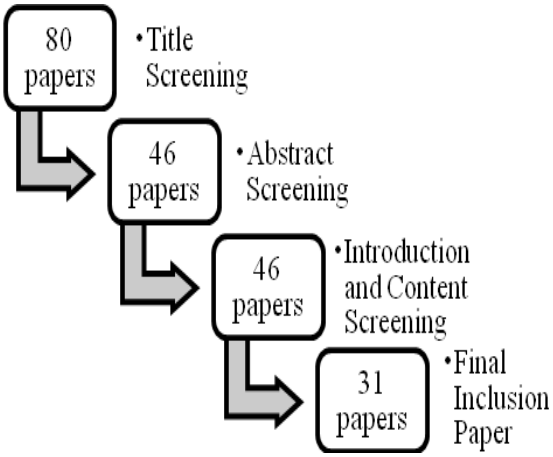

Figure 1. Steps in screening papers.

3. Results

3.1 Search Results

Appendix A displays the list of selected papers based on the search and keyword criteria.

The frequency of papers issued by year can be observed in Table 2. From 2002, papers associated with the study have shown a distinct improvement. This improvement can be traced to the presence of coordination in the success of project development. On the other hand, from 1995 until 2001 only a few papers that focused on coordination in SDLC models were successfully issued.

Table 3 classifies papers related to coordination in different SDLC models. Studies on coordination were mostly conducted in agile model.

4. Discussion

4.1 How does Coordination Occurr in different Software Development Lifecycles? (RQ1)

The various ways coordination takes place in different SDLC models are discussed as follows:

Table 2. Paper issued (by year)

Year	Percentage (%)	Frequency
1995	0	0
1996	3	1
1997	3	1
1998	0	0
1999	3	1
2000	3	1
2001	3	1
2002	7	2
2003	7	2
2004	3	1
2005	9	3
2006	9	3
2007	9	3
2008	3	1
2009	7	2
2010	7	2
2011	3	1
2012	14	4
2013	0	0
2014	0	0
2015	7	2
Total	100%	31 papers

Table 3. List of SDLC models

SDLC Model	Paper ID
Waterfall Model	(2) (3)(4)(5)(6)(7)
Spiral Model	(3)(2) (8)(9)(10)
Agile Model	(11) (12) (13) (14) (15) (16) (17) (18) (19) (20) (21)(22) (3)(2)(5)
Rapid Application Development (RAD) Model	(23)(24)(25) (26)(27)
V-model	(28) (29) (5)
Incremental model	(30)(31) (4)(32)

4.1.1 Waterfall Model

Waterfall is a traditional model. There are five phases typically used in a waterfall model which include requirement phase, design phase, implementation phase, testing phase and maintenance phase.

Waterfall model is well-known for its sequential development process[2–4]. As a matter of fact, sequential approach prevents people who are engaged in other phases to be involved in the current working phase[5]. In other words, each phase must be fully completed before moving on to the next phase. This model cannot return to the previous phase if the developer goes beyond the current phase. For example, if the developer has completed the requirement phase and is already at the design phase, any changes that exist in the requirement phase cannot be altered anymore. If users still need to change the requirements, they will have to wait until all phases are completed before they can proceed to another cycle. Thus, this indicates that low levels of interaction occurs between phases in waterfall model[7].

In addition, the software development project of waterfall model is structured through good documentation [6]. Thus for distributed teams, this is considered an advantage since they need to coordinate with each other to reduce the problems, especially with less communication between them. Every phase is documented in the waterfall model and full documentation can be seen at the last phase which is deployment[6]. With well documentation, the developer is able to work on his own by referring to design documentation of the product characteristics to be developed until the integration phase begins.

4.1.2 Spiral Model

Spiral model evolved from the waterfall model and is the combination of iterative and sequential approach[10]. Spiral model consists of four phases which are the requirement phase, design phase, construct or build phase and evaluation and risk analysis[9]. Spiral model is suitable for a large project where requirements are clear and fixed. Besides that, this model emphasizes more on risk assessment[3,8].

Coordination in spiral models usually can be seen at the end of each cycle when review takes place. Review process requires stakeholders to coordinate before the next cycle begins[2]. This entails collaboration between stakeholders to review products of previous cycles, besides planning for the next cycle and resources involved[2]. Nevertheless, this model stresses more on coordination through formal communication in the form of detailed documentation[8].

4.1.3 Agile Model

Agile model is another model featured in the software development lifecycle. There are five phases in an agile model which forms mini-incremental iteratively, including the planning phase, analysis phase, design phase, implementation phase and testing phase.

Agile model is suitable only for small teams. This is to maintain effectiveness of coordination among teams through informal communication practices[20]. Informal communication emphasized in this model is face-to-face

communication like face-to-face meetings, audio/video conferences or email conversations[12,15,17,22]. Coordination through informal communication is more important in agile models compared to formal communication because projects can proceed smoothly when developers can discuss if an issue arises[2,12,13,16]. Agile model is flexible as it adheres to changes in an attempt to fulfill client's needs[5,18]. Nonetheless, large teams are not as flexible as small teams in addressing changes[3,21].

The agile development approach emphasizes on coordination and collaboration to share tacit knowledge among team members[19]. In this case, tacit knowledge is retained by agile methods through regular face-to-face communication. With this intention, this model is able to practice knowledge sharing among teams while communicating. Agile practices such as eXtreme Programming (XP) and Scrum have had positive effects on communication involving development teams, thus contributing to productive software development. Regular meetings of Scrum is the best method in coordinating the teams[11,14]. Teams are able to communicate face-to-face in sharing knowledge, and deal with any arising issues during the development process[14].

4.1.4 Rapid Application Development (RAD) Model

Rapid Application Development (RAD) model is a concept in which high quality products can be developed rapidly[23]. Therefore, this model focuses on developing systems that are less complex where the framework is already in place[23,26]. The phases involved in this model are the business modeling phase, data modeling phase, process modeling phase, application generation phase and testing phase. Developers come up with products in parallel with products that have been broken down into modules and then combined into a functioning prototype. Software developers who implement this model use a prototype that will be improved over and over again based on feedback from users[26]. In the meantime, a software development team for each module is allocated a fixed time period to start development concurrently[23]. This is in accordance with the model name, known as Rapid Application Development model.

Coordination in Rapid Application Development (RAD) is focused on formal communication which requires documentation, unlike informal mode of communication[26]. RAD usually involves a small team size[24]. In RAD, communication is particularly important to maintain an understanding among the team members[25]. RAD teams consist of developers and users who are authorized to make design decisions, hence coordination among them is usually on a weekly, monthly or quarterly basis[27]. Users and developers interact either through formal or informal meetings[27].

4.1.5 V-Model

V-model is similar to waterfall model which is sequential, but each phase in the V-model will go through testing to make sure there are no errors. Phases involved are the requirement phase, system design phase, architecture design phase, module design phase and coding phase. Once requirement phase is completed, the developer will perform the test using acceptance testing. System design phase will be tested using system test while architectural design phase will be tested using integration test, followed by modular design phase which is tested using the unit testing. Lastly, coding phase will connect two halves of the V-model phase which were involved in software development and tested in each phase.

As for the V-model, although it is a traditional model, coordination is marginally better since the development team and testing team need to collaborate before moving on to the next phase[5]. Coordination is considered essential in V-models with mitigation plans like video or audio conference, face-to-face meeting and early prototyping taken to prevent any further miscommunication[28]. Meanwhile, in the test phase, formal communication is not always practiced because documentation serves more for large teams than small teams[29].

4.1.6 Incremental Model

Incremental model is an improved version of the waterfall model. The phases involved in incremental model are the requirement phase, design and development phase, testing phase and implementation phase. This model is called incremental as the project is developed based on systems that are divided into parts[4]. This model will not meet the overall requirement of users, but will complete the first requirement before fulfilling the next requirement. Each requirement that is met will go through each phase of the incremental model.

Coordination between stakeholders in incremental models occur iteratively when they are required to alternate in lending cooperation until the end product is

successfully delivered[31]. Coordination during implementation phase can be seen through team communication and collaboration such as walkthrough, peer review, project meeting and so on[30]. Incremental models are flexible, that is to say when there are modifications in requirements, this model allows changes to be made easily[4]. For this reason, developers need to coordinate through discussion in solving problems, and mutual agreement to keep updated on the progress[32].

The mapping shown in Table 4 shows that coordination occurs most frequently in agile model, V model and incremental model. There is a lack of coordination awareness in waterfall model and RAD model.

5. Conclusion

This paper explores coordination that occurs in different software development lifecycle models of current literature. Besides, this paper also maps research papers published by year. This research is important because it provides information on coordination that takes place in various SDLC models.

6. Acknowledgements

We are grateful to the Ministry of Higher Education for supporting this research, through Research Acculturation Collaborative Effort (RACE) grant.

7. References

1. Malone TW. What is coordination theory? Technology. 1988; 1–32.
2. Ruparelia NB. Software development lifecycle models. ACM Sigsoft Softw Eng Notes. 2010; 35(3):8.
3. Mujumdar A, Masiwal G, Chawan P. Analysis of various software process models. Int J Eng Res Appl. 2012; 2(3):2015–21.
4. Moløkken-Ostvold K. A comparison of software project overruns-flexible versus sequential development models. IEEE Transactions on Software Engineering 2005; 31(9):754–66.
5. Balaji S. Wateerfall vs v-model vs agile: A comparative study on SDLC. International Journal of Information Technology and Business Management. 2012; 2(1):26–30.

Table 4. Summary of coordination in SDLC models

SDLC Model	Phases				
Waterfall	Requirement	Design	Implementation	Testing	Maintenance
	X				
Spiral	Requirement	Design	Construct or Build	Evaluation and risk analysis	-
	X			X	
Agile	Planning	Analysis	Design	Implementation	Testing
	X	X	X	X	X
RAD	Business Modeling	Data Modeling	Process Modeling	Application Generation	Testing
	X				
V model	Requirement	System Design	Architecture Design	Module Design	Coding
	X	X	X	X	X
Incremental	Requirement	Design and Development	Testing	Implementation	-
	X	X	X	X	

6. Papadopoulos G. Moving from traditional to agile software development methodologies also on large, distributed projects. Procedia - Soc Behav Sci. Elsevier BV. 2015; 175:455–63.
7. Knauss E, El Boustani C, Flohr T. Investigating the impact of software requirements specification quality on project success. Product-Focused Software Process Improvement. 2009. p. 28–42.
8. Qumer A. Henderson-Sellers B. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Inf Softw Technol. 2008; 50(4):280–95.
9. Sakthivel S. Virtual workgroups in offshore systems development. Inf Softw Technol. 2005; 47(5):305–18.
10. Trammell CJ, Pleszkoch MG, Linger RC, Hevner AR. The incremental development process in cleanroom software engineering. Decis Support Syst. 1996; 17:55–71.
11. Ashraf M, Shamail S, Rana Z. Agile model adaptation for e-learning students' final-year project. Proc IEEE Int Conf on Teaching, Assess Learn Eng (TALE); Hong Kong. 2012 Aug 20-23. p. 18–21.
12. Turk D, France R, Rumpe B. Assumptions under lying agile softwar e development pr ocesses abstract 2. Overview of eXtreme programming- A representative agile process. J Database Manag. 2005; 16(4):62–87.
13. Ramesh B, Cao P, Mohan L, Xu K. Can distributed software development be agile? Commun ACM. 2006; 49:41–6.
14. Moe NB, Aurum A, Dyba T. Challenges of shared decision-making: A multiple case study of agile software development. Inf Softw Technol. 2012; 54(8):853–65.
15. Strode DE, Huff SL, Hope B, Link S. Coordination in co-located agile software development projects. J Syst Softw. 2012; 85(6):1222–38.
16. Paasivaara M, Lassenius C. Could global software development benefit from agile methods? IEEE Int Conf Glob Softw Eng (ICGSE '06); 2006. p. 109–13.
17. Dyba T, Dingsoyr T. Empirical studies of agile software development: A systematic review. Inf Softw Technol. 2008; 50(9-10):833–59.
18. Hansson C, Dittrich Y, Gustafsson B, Zarnak S. How agile are industrial software development practices? J Syst Softw. 2006; 79(9):1295–311.
19. Chau T, Maurer F, Melnik G. Knowledge sharing: Agile methods vs tayloristic methods. Proc 12th IEEE Int Work Enabling Technol Infrastruct Collab Enterp WET ICE 2003; Linz, Austria. 2003 Jun 9-11. p. 302–7.
20. Turk D, France R, Rumpe B. Limitations of agile software processes. 3rd Int Conf Extrem Program Agil Process Softw Eng (XP 2002); 2002.
21. Begel A, Nagappan N. Usage and perceptions of agile software development in an industrial context: An exploratory study. Proc - 1st Int Symp Empir Softw Eng Meas ESEM 2007; 2007. p. 255–64.
22. Sohaib O, Khan K. Integrating usability engineering and agile software development: A literature review. 2010 Int Conf Comput Des Appl ICCDA 2010; (Iccda). 2010.
23. Beynon-Davies P, Carne C, Mackay H, Tudhope D. Rapid Application Development (RAD): An empirical review. Eur J Inf Syst. 1999; 8(3):211–23.
24. Berger H, Beynon-Davies P. The utility of rapid application development in large-scale, complex projects. Inf Syst J. 2009; 19(6):549–70.
25. Howard A. A new RAD-based approach to commercial information systems development: the dynamic system development method. Ind Manag Data Syst. 1997; 97(5):175–7.
26. Beynon-Davies P, Mackay H, Tudhope D. It's lots of bits of paper and ticks and post-it notes and things...: A case study of a rapid application development project. Inf Syst J. 2000; 10(3):195–216.
27. Beynon-Davies P, Holmes S. Design breakdowns, scenarios and rapid application development. Inf Softw Technol. 2002; 44(10):579–92.
28. Sudershana S, Villca-Roque A, Baldanza J. Successful collaborative software projects for medical devices in an FDA regulated environment: Myth or reality? Proc - Int Conf Glob Softw Eng ICGSE 2007; 2007. p. 217–24.
29. Pyhajarvi M, Rautiainen K, Itkonen J. Increasing understanding of the modern testing perspective in software development projects. 36th Annu Hawaii Int Conf Syst Sci 2003 Proc; 2003. p. 10.
30. Chiang IR, Mookerjee VS. Improving software team productivity. Commun ACM. 2004; 47:89–93.
31. Robey D, Welke R, Turk D. Traditional, iterative, and component-based development: A social analysis of software development paradigms. Inf Technol Manag. 2001; 2:53–70.
32. Franken S, Kolvenbach S, Prinz W. CloudTeams: Bridging the gap between developers and customers during software development processes. Procedia - Procedia Comput Sci. Elsevier Masson SAS. 2015; 68:188–95.

Appendix A. Literature Review Studies

Citation	Author	Year	Topic
(2)	N. B. Ruparelia	2010	Software Development Lifecycle Models
(3)	Ashwini et al.	2012	Analysis of various Software Process Models
(4)	Kjetil et al.	2005	A Comparison of Software Project Overruns – Flexible Versus Sequential Development Models
(5)	S.Balaji and M.S. Murugaiyan	2012	Wateerfall vs V-Model Vs Agile : A Comparative Study on SDLC
(6)	G. Papadopoulos	2015	Moving from traditional to agile software development methodologies, also on large, distributed projects.
(7)	E. Knauss, C. El Boustani, and T. Flohr	2009	Investigating the impact of software requirements specification quality on project success
(8)	a. Qumer and B. Henderson-Sellers	2008	An evaluation of the degree of agility in six agile methods and its applicability for method engineering
(9)	S. Sakthivel	2005	Virtual workgroups in offshore systems development
(10)	C. J. Trammell et al.	1996	The incremental development process in cleanroom software engineering
(11)	M. Ashraf, S. Shamail, and Z. Rana	2012	Agile Model Adaptation for E-Learning Students' Final-Year Project
(12)	D. Turk et al.	2005	Assumptions Under lying Agile Software Development Processes
(13)	Balasubramaniam et al.	2006	Can Distributed Software Development Be Agile?
(14)	Nils et al	2012	Challenges of Shared Decision-Making: A Multiple Case Study of Agile Software Development
(15)	Diane et al.	2011	Coordination in Collocated Agile Software Development Projects
(16)	M. Paasivaara and C. Lassenius	2006	Could Global Software Development Benefit from Agile Methods?
(17)	T. Dybå and T. Dingsøyr	2007	Empirical studies of agile software development: A systematic review
(18)	C. Hansson et al.	2006	How agile are industrial software development practices?
(19)	T. Chau et al.	2003	Knowledge Sharing: Agile Methods vs. Tayloristic Methods
(20)	D. Turk, R. France, and B. Rumpe	2002	Limitations of Agile Software Processes
(21)	A. Begel and N. Nagappan	2007	Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study
(22)	O. Sohaib and K. Khan	2010	Integrating Usability Engineering and Agile Software Development: A Literature Review
(23)	P Beynon-Davies et al.	1999	Rapid application development (RAD): an empirical review
(24)	H. Berger and P. Beynon-Davies	2009	The utility of rapid application development in large-scale, complex projects
(25)	A. Howard	1997	A new RAD-based approach to commercial information systems development: the dynamic system development method
(26)	P Beynon-Davies et al.	2000	'It's lots of bits of paper and ticks and post it notes and things . . .': a case study of a rapid application development project
(27)	P Beynon-Davies and S. Holmes	2002	Design breakdowns, scenarios and rapid application development
(28)	S. Sudershana et al.	2007	Successful Collaborative Software Projects for Medical Devices in an FDA Regulated Environment: Myth or Reality?
(29)	M. Pyhajarvi et al.	2003	Increasing Understanding of the Modern Testing Perspective in Software Product Development Project

(30)	I. R. Chiang and V. S. Mookerjee	2004	Improving Software Team Productivity
(31)	D. Robey et al.	2001	Traditional, Iterative, and Component-Based Development: A Social Analysis of Software Development Paradigm
(32)	Sebastian et al.	2015	Cloud Teams: Bridging the Gap between Developers and Customers during Software Development Processes