

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/323770902>

THE FOUR (4) PRIMARY MODELS OF SOFTWARE DEVELOPMENT METHODS

Article · March 2018

CITATIONS

0

READS

406

3 authors, including:



Eka Arriyanti

STMIK Widya Cipta Dharma, Samarinda, Indonesia

4 PUBLICATIONS 4 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Fuzzy likert scale to measure perception [View project](#)



Data Mining Optimization [View project](#)



THE FOUR (4) PRIMARY MODELS OF SOFTWARE DEVELOPMENT METHODS

Eka Arriyanti

STMIK Widya Cipta Dharma, Indonesia

Rufman I. A. Efendi and Hoga Saragih

STMIK Eresha, Indonesia

Discipline is an important attitude in developing process (system or software). Undiscipline in developing process, sometimes is caused by confusion in understanding what the right model or method for the right project. Hence, the developers often ignore the right steps in developing process or the students often do not care on the system development methods, whereas the development methods or models are very important, specifically for managing schedule and budget as the work needs of project. Ignoring may be no significant problem for small projects, but yes for more than small ones. The problem root of this confusion is apparently caused by various theories on system development methods and software development models with all of those similarities and differences. There is no firm concept in explaining the excellence and the weakness of the software process models or methods existed. Hence, students or researchers should do comparative study on them before doing research on implementation of a new method or model. This research tries to unite that various theories to be a firm concept on models or methods with getting back them to their mother concepts. Analyzing for taking conclusion used deductive and inductive technic and with grouping based on model name, life-cycle type, early model, model advanced, excellence, weakness, project scale-relative (PS-R), and budget-basic estimation. Therefore, the four (4) primary models of software development methods are Linear Sequential (Waterfall), Incremental Development (ID), Reuse-Oriented (Rapid Application Development ; RAD), and Rational Unified Process (RUP).

Keywords: Model-method, Software development, System development, Process discipline.

Introduction

Software is a product of technical engineering in computer sciences. The technical engineering process shall take the development steps due to project scales and needs. In small scale projects, ignoring development steps or choosing a software development step tend to classical model (waterfall), may not influence the compatibility between software produced and development needed. However, if software developments are about engineering systems or project scales more than small, then developers can not ignore the steps (method) of system or software development.

In fact, many developers often ignore the methods of system or software development. They tend to work directly to construction step of software and it ends fast. Another fact, students often have confusion with various theories on system or software development methods or software development models. With

all similarities and differences of their steps, which methods or models are for which projects ?. This question implies advice, “Just do what software needs, what system needs may follow”. That is certainly not right for academic atmosphere. Because if that is, then why do the experts always emphasize the importance of knowing needs of system or software before constructing software ?. What may the students or developers keep the fix of their projects with among needs of system or software, schedule, and budget ?.

The practices could not be blamed just like justify the academicians. A link should be built. This is for instance in previous research that the comparative study on the excellence and the weakness of the software process models or methods existed due to their name and life-cycle type, was done before researching the implementation of Rational Unified Process (RUP) method. Scientifically, that study is a procedure. To know about how a specific method or model for system or software development be implemented is needed information on how previous methods or models have been implemented. And in its benefit of research, students or researchers become to know firmly for each method or model while developers, also students, may use it as the following table to decide which models will be used for project.

This paper of research titled “The Four (4) Primary Models of Software Development Methods” is to firm the concept on models or methods of software or system development after getting back their various theories to their mother. Grouping is a result of comparative study in advance. Easy deciding the chosen models of software development methods, scheduling, and budgeting are expected for developers and students.

Literature Review

Software Development Methods (SfDM)

SfDM is for differentiating acronym in theory from SDM (System Development Methods). Nowadays, it is very difficult to separate software projects with system projects. Demands on software compatibility, either mass or specialized productions, need studying on system needs. SfDM as like as SDM, evolved from their classical methods, indeed following a cycling steps in literatures. Rainer Jr. and Cegielski (2011), Laudon and Laudon (2012), and Irani and Love (2008) in system (information) literatures, prefer say life-cycle models than development methods. They say that all life-cycle models descript project process of system or software.

In software (engineering) literatures, SfDM is called software process, a set of related activities that leads to the production of a software product (Sommerville, 2011). The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed (Presmann, 2010). In software (project) literatures, SfDM (or SDM) can not be spilled by project scales and development needs. Both subjects are influenced by engineering, scientific, mathematics, and economic disciplines for quality softwares.

Models of Software Development Methods (MSDM)

MSDM is a process description of SfDM. Schach (2007) and Sommerville (2011) called it as Software Process Models (SPM). Schah says that SPM is the ways software produced while Sommerville (2011) is the simplified representations of software process. MSDM is what software development models purposed.

Project Scales of SfDM or SDM

Concluded from RPL (2010) and Laporte, Chevalier, and Maurice, (2013), there are 3 (three) project scales of SfDM or SDM which be measured by number of system analysts and programmers. They are :

1. Small project ; needs 1-4 analysts and or programmers ; duration is about less than 2 months.
Analysts and programmers are same ones or analysts and programmers are different ones.
2. Middle project ; needs 2-3 analysts and 2-5 programmers ; duration is about 2-8 months.
3. Big project ; needs 2 analysts or more and 5 programmers or more ; duration is about more than 8 months.

Mother Concept

Mother concept is not a classical model. Mother concept is a term used in this paper to call the early model of a method that from it is advanced the next model, which is significant different from the previous one, on software process or system development description. Classical model is generic framework activities of software process or system development. Generic framework activities of software process are communication, planning, modeling, construction, and deployment (Presmann, 2010). Generic framework activities of system development are analysis, design, implementation, testing, and evaluation (SDLC, 2015).

Research Method

General

The general research method purposed is begun by doing comparative study among SDM, SfDM, and MSDM theories in about 21 literatures (2005-2014), analyzing the theories for finding conclusion on the mother concepts with deductive and inductive technic, and grouping the mother concepts based on model name, life-cycle type, early model, model advanced, excellence, weakness, project scale-relative (PS-R), and budget-basic estimation. Model name is name of MSDM based on their mother concepts. Life-cycle type is nature of cycling steps. Early model is figure of mother concepts. Model advanced is figure example of mother concepts advanced. Excellence is comparative statements of MSDM's excellence. Weakness is comparative statements of MSDM's weakness. PS-R is relative size of projects based on number of system analysts, programmers, and duration. Budget-basic estimation is estimation of project worker fee based on MSDM.

Specific

The specific research method purposed is to determine the estimation of project worker fee based on MSDM for finding a pattern of budget-basic estimation. It is :

Since,

small project ; $1 \leq \text{analysts and or programmers} \leq 4$; duration < 2 months

middle project ; $2 \leq \text{analysts} \leq 3$ and $2 \leq \text{programmers} \leq 5$; $2 \leq \text{duration} \leq 8$ (months)

big project ; $\text{analysts} \geq 2$ and $\text{programmers} \geq 5$; $d > 8$ months

Let A = analysts

P = programmers

N = number of analysts and or programmers

d = duration of project scale in hour

Then,

$$\begin{aligned} N &= A + P + (A \cdot 1 \cdot P) && \text{for small project} \\ N &= A + P && \text{for middle and big project} \end{aligned}$$

Hence,

$$\text{small project ; } [1 + 1 + (1)] \leq N \leq [4 + 4 + (4)] \quad ; \quad d < 2 \text{ months} = 2 \times 30 \times 24 \text{ (hours)} \\ = 1440 \text{ hours}$$

$$; \quad 3 \leq N \leq 12 \quad ; \quad d < 1440 \text{ hours} \quad (1)$$

$$\text{middle project ; } (2 + 2) \leq N \leq (3 + 5) \quad ; \quad 2 \leq d \leq 8 \text{ (months)}$$

$$; \quad 4 \leq N \leq 8 \quad ; \quad 1440 \leq d \leq 5760 \text{ (hours)} \quad (2)$$

$$\text{big project ; } N \geq (2 + 5) \quad ; \quad d > 8 \text{ months}$$

$$; \quad N \geq 7 \quad ; \quad d > 5760 \text{ hours} \quad (3)$$

Then,

Let f = fee of an analysts or a programmer per hour

p = estimation of working proportion based on MSDM

$m = N_{\text{minimum}}$ per project scale

E = estimation of project worker fee based on MSDM

Then,

$$E = p \times d \times f \times N \times \frac{1}{m} \quad (4)$$

And then,

$$E \geq p \times 1440 \times f \times 3 \times \frac{1}{8} = 1440fp, \text{ if only small project has } N_{\text{min.}} = 3 \text{ and } d < 1440$$

Hence,

$$E \geq 1440fp \text{ (fee unit ; IDR, USD, GBP, etc.)} \quad (5)$$

And hence,

$$\text{small project ; } 1440fp \leq E < 5760fp \quad (6)$$

$$\text{middle project ; } E = 5760fp \quad (7)$$

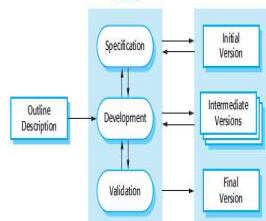
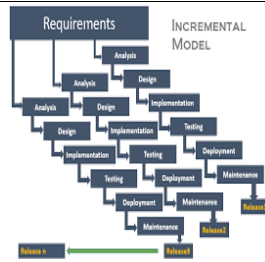
$$\text{big project ; } E > 5760fp \quad (8)$$

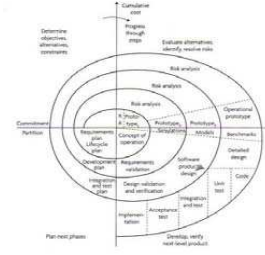
Results

Results of this research are said by table following. To read the table specifically for column 7 and 8 ; column 7 refers to number (1), (2), or (3), number of analysts and or programmers, and duration for each projects scale. Sequence of number indicates priority ; column 8 refers to number (6), (7), or (8), estimation of project worker fee based on MSDM for each project scale. This column explains column 7.

for the projects until its ending passed. If there is an error which is not known from beginning, then it will be a big problem, because the projects should be repeated from beginning again.

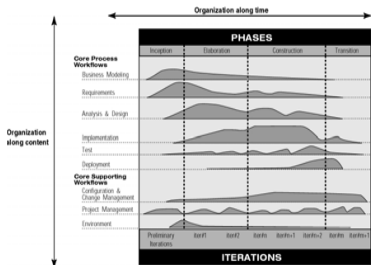
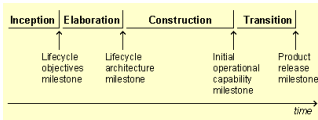
4. No efficient (developers often do unnecessary delay because members of the project team should wait for the other teams to finish their work due to a sequential flow).
-

	1	2	3	4	5	6	7	8
2	Incremental Development (ID)	Classic-iterative (evolutionary) ; dynamic (agile) ; prototyping	 <p>$P_{\text{Analysts}} = \frac{3}{12}$$P_{\text{Programmers}} = \frac{8}{13}$</p>	 <p>INCREMENTAL MODEL</p>	<ol style="list-style-type: none">1. There is a good communication between developers and customers, so developers work better in fixing customers' needs.2. Customers actively participate in developing so that the implementation becomes easier, because user candidates know what they want.3. More efficient.4. Adaptable.5. Timeless.6. Developers and users will easier understand and react on to each risk of the evolution level because of software works during the	<ol style="list-style-type: none">1. Sometimes, customers do not see or aware that the software existed has not showed the whole quality of software and they have not thought the maintenance ability for long time.2. Developers tend to work faster for ending the project. Generally, they prefer use simple algorithm and programming language for prototyping, the blueprint. They do not care the real product.3. Customer-computer interaction may not reflect good design technic.	<p>(2)</p> <p>(7) ; $E_{\text{Analysts}} =$ $E_{\text{Programmers}} =$ $E ;$ $E =$ 1329.2308f</p> <p>(3)</p> <p>(8) ; $E_{\text{Analysts}} =$ $E_{\text{Programmers}} =$ $E ;$ $E >$ 1329.2308f</p>	





- process.
7. Consistent in following classical life-cycle steps and bringing it in to iterative framework.
8. The process needs direct consideration on to technical risk. So that, it will reduce the risk before being serious problem.
9. Each iteration will be evaluated in ending of the iteration. Hence, the minus factor will be detected soon.
10. Flexible, be recommended for independent deadline organization.
4. Difficult to make sure customers that the evolutionary approach can be controlled.
5. The process needs the rational risk estimation, the major risk. If it is not found and managed, then it will be a seriously problem. The time is long to make the rational risk estimation paradigm being an absolute risk.
6. Each iteration is a complete cycle and the next iteration has to wait for the previous one (classic-iterative).

	1	2	3	4	5	6	7	8
4	Rational Unified Process (RUP)	Modern-OO ; iterative ; dinamik (agile)	<div>Static structure :</div> <div></div> <div>Dynamic structure :</div> <div></div> <div>$P_{Analysts} = \frac{4}{4}$<div>= $p_{Programmers}$; their work portions are same along the 4 phases (inception, elaboration, construction, transition)</div><div>Comparison of iteration number in each phase :</div><div>Inception : Elaboration : Construction : Transition</div><div>1 : 3 : 5 : 1</div><div>n : 3n : 5n : n</div><div>n = project scale variable (Arriyanti, 2014)</div></div> <td><div>1. Flexible for big or small scale project of software or system development because it is planed due to time and cost.</div><div>2. The formal cycle is defined by 4 phases, but they can still be executed together.</div><div>3. Consistent in following classical life-cycle steps and bringing it in to iterative frameworks.</div><div>4. Detecting and overcoming the risk earlier.</div><div>5. There is a good communication among developers, customers, and managements, so that developers</div></td> <td><div>1. Because of size and its heterogeneity, RUP can be complex process. Hence, its implementation adapts the development condition after considering system entirely, not adapts the architecture entirely.</div><div>2. The architecture should be understood as a process model, not only as a static model.</div></td> <td><div>(3) (8) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $E > 5760f$</div><div>(2) (7) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $E = 5760f$</div><div>(1) (6) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $1440f \leq E < 5760f$</div></td>	<div>1. Flexible for big or small scale project of software or system development because it is planed due to time and cost.</div> <div>2. The formal cycle is defined by 4 phases, but they can still be executed together.</div> <div>3. Consistent in following classical life-cycle steps and bringing it in to iterative frameworks.</div> <div>4. Detecting and overcoming the risk earlier.</div> <div>5. There is a good communication among developers, customers, and managements, so that developers</div>	<div>1. Because of size and its heterogeneity, RUP can be complex process. Hence, its implementation adapts the development condition after considering system entirely, not adapts the architecture entirely.</div> <div>2. The architecture should be understood as a process model, not only as a static model.</div>	<div>(3) (8) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $E > 5760f$</div> <div>(2) (7) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $E = 5760f$</div> <div>(1) (6) ; $E_{Analysts} =$ $E_{Programmers} =$ $E ;$ $1440f \leq E < 5760f$</div>		

-
- can work better in determining the needs of organization.
6. Realistic ; the software engineering process and system development that is compatible with the real condition of process and organization and suitable with development capability.
 7. Uniformity ; having UML tool and the others of Rational which can be used either integratedly or separatedly.
 8. Understandability ; the codes resulted can be organized in to the class related with real problems. Hence,
-

-
- it is easier to understand.
 - 9. Stability ; the program code resulted is relatively stable because it approaches real problems.
 - 10. Reusability ; possible in reusing codes so it is to speed the time of software development.
-

Conclusions

Due to this research, the conclusions are:

1. The four (4) primary models of software development methods are Linear Sequential (Waterfall), Incremental Development (ID), Reuse-Oriented (Rapid Application Development ; RAD), and Rational Unified Process (RUP).
2. E ; Estimation of project worker fee based on MSDM are :

$$E = p \times d \times f \times N \times \frac{1}{m} ; \quad p = \text{estimation of working proportion based on MSDM}$$

d = duration of project scale in hour

f = fee of an analysts or a programmer per hour

N = number of analysts and or programmers ;

$$N = A + P ; \quad A = \text{analysts}$$

P = programmers

m = N_{minimum} per project scale

and

$$E \geq 1440\text{fp (fee unit ; IDR, USD, GBP, etc.)}$$

Implications

Due to table of results, if the four (4) life-cycle types now are Classic-non iterative ; static, Classic-iterative (evolutionary) ; dynamic (agile) ; prototyping, Modern-based on component (object-oriented ; OO), Modern-OO ; iterative ; dinamik (agile), then the next may be Modern-OO; prototyping.

References

1. Arriyanti, E. (2014) *Design and Implementation of the Language Center Information System Used RUP Method*. Thesis. STMIK Eresha : Jakarta, Indonesia.
2. Baars, W. (2006) *Project Management Handbook*. Ver. 1.1. DANS : Hague, USA.
3. Begetis, I. (2010) *Combining Design and Engineering Methodology for Organizations with the Rational Unified Process*. Thesis. Delft University of Technology : Delft, Netherlands.
4. Gouws, J. and Gouws, L.E. (2006) *Fundamentals of Software Engineering Project Management*. Feed Forward Publications : Australia.
5. Irani, Z. and Love, P. (2008) *Evaluating Information System, Public and Private Sector*. Elsevier-Butterworth-Heinemann : Burlington, USA.
6. Janacek, G.J. and Close, M.L. (2011) *Mathematics for Computer Scientists*. Ventus Publishing AVS : www.bookboon.com
7. Laporte, C.Y., Chevalier, F., and Maurice, J-C. (2013) *Improving Project Management for Small Projects*. ISO Focus, www.iso.org/isofocus+. Date accessed 9th October 2015.
8. Larman, C. (2005) *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed., Addison Wesley Professional : New Jersey, USA.
9. Laudon, K.C. and Laudon, J.P. (2012) *Management Information Systems, Managing the Digital Firm*. 12th ed., Prentice Hall : New Jersey, USA.

10. Leppänen, M. (2013) *Information Systems Development Methods*. 13th Outline.
11. O'Docherty, M. (2005) *Object-Oriented Analysis and Design, Understanding System Development with UML 2.0*. John Wiley and Sons Ltd. : Sussex, England.
12. Phillips, J.R. (2006) *Team-RUP : An Agent-Based Simulation Study of Team Behavior In Software Development Organizations*. Thesis. Auburn University : Alabama, USA.
13. Pressman, R.S. (2010) *Software Engineering, a Practitioner's Approach*. 5th ed., McGraw-Hill : New York, USA.
14. Rainer Jr., R.K. and Cegielski, C.G. (2011) *Introduction to Information Systems, Supporting and Transforming Business*. 3rd ed., John Wiley and Sons Inc. : Hoboken, USA.
15. RPL (2010) *Perencanaan Rekayasa Perangkat Lunak, Kategori Ukuran Proyek*. (Software Engineering Planning, Project Scales Category) <http://ral-fikom.blogspot.co.id>. Date accessed 9th October 2015.
16. RSWP (2011) *Best Practices for Software Development Team*. Copyright 2001, Rational The Software Development Team : Massachusetts, USA.
17. Schach, S.R. (2007) *Object-Oriented and Classical Software Engineering*. 7th ed., McGraw-Hill : New York, USA.
18. SDLC (2015) SDLC, Circular Arrow Diagram. www.conceptdraw.com. Date accessed 9th October 2015.
19. Sommerville, I. (2011) *Software Engineering*. 9th ed., Pearson : Massachusetts, USA.
20. Valacich, J. and Jessup, L. (2008) *Information Systems Today, Managing in the Digital World*. 3th ed., Pearson : New Jersey.
21. Whitten, J.L., Bentley, L.D., and Dittman, K.C. (2009) *Metode Desain dan Analisis Sistem*. (System Analyze and Design Method) 6th int. ed., ANDI-McGraw Hill Edu. translator team : Jakarta, Indonesia.