Framework to Define Software Process Model in the Ethiopian Context

Tadesse Aregawi
tadesseare@gmail.com

Dagmawi Lemma
Department of Computer Science, Addis Ababa University, Ethiopia
dagmawil@yahoo.com

Abstract

The Federal Public Procurement Directive (FPPD), which is set by the Ministry of Finance and Economic Development (MoFED), is a directive for guiding and controlling public procurement process made by Federal institutes, including software procurement. The FPPD is not flexible enough to fit with flexible process models so it has to be revisited and shall create conducive environment to development stakeholders. Therefore, it is required to propose a framework to define a software process model that goes with the current procurement directive.

This paper surveys software process models in an effort to build an effective taxonomy that categorizes process models in terms of their functionalities and their interrelationships to the FPPD. The research, descriptive survey method, is carried out through a 4-steps process: data collection, data analysis, data representation, and data grouping, which involved a literature survey of existing process models.

Based on the survey, the software process models were evaluated statistically in view of FPPD and frameworks are introduced to address similarities and differences among software process models. Finally, the proposed solution is validated and directions for future research work are suggested.

Keywords: Software Process Model; Comparison Matrix; Data Grouping

1. Background

A software development process model is a description of the work practices, tools, and techniques used to develop software [1]. An appropriate software process model supports developers in creating software that meets customer requirements with low cost and less number of defects at an acceptable quality level.

A software process model tries to capture the main characteristics of the activities performed to develop software. A variety of models have been proposed. Most process models are defined with the objective of entertaining requirement change, early/on-time product delivery, and risk management.

In Ethiopia, there is FPPD set by MoFED which focuses on the promotion of proper implementation of public procurement proclamation, development of manpower capacity to carry out public procurement functions and strengthening of uniform application of procedures and standard bidding document in all public sectors at federal and regional level [2]. It allows services and goods to be purchased in a planned manner and within the allocated budget. Though the procurement directive is primarily applicable in governmental institutes, as the government is spending on information technology (IT) and continues to consume a large share of national budgets, its significance has become pertinent.

Likewise, the software procurement process, be it off-the-shelf or otherwise, has to meet requirements of the FPPD. As a result, the directive seems to have no flexibility to define software tenders in accordance to the variety of software process models. Hence, this paper investigates the strengths and weaknesses of existing software process models to examine their suitability in view of the FPPD and finally proposes a suitable framework to define a process model as a solution.

2. Related Work

According to [4], efforts have been made to tailor software process models to project requirements. The need comes from the ever changing nature of software process model. Software process modeling has undergone extensive changes in the last three

decades. This evolution resulted in a diversity of models that tended to suit software project requirements from different angles.

This paper investigates the main streams of process models in an effort to build taxonomy of super classes of process models and their interrelationships. For the software process model to be tailorable to its users' needs, it is necessary to develop an understanding of two important issues. First, it is important to understand the process and study how software development models impact software products. This requires surveying the literature on process models and studying the evolution of these models and their relationships. It is also important to assess the goals of these software process models, which is essential to evaluating them and identifying their differences. Second, it is also important to understand software user requirements. This is a relevant dimension in defining an evaluation method for process models and in determining criteria for matching project needs with process models.

The evolution of the models resulted in a diversity of models that tended to suit software project requirements from different angles. This investigation involved a literature survey of both existing process models and process modeling assessments. Based on this taxonomy, an eight-step evaluation [4] process was carried out. By utilizing manual and automated decision tables, a final assessment was presented. According to project management, a set of decision criteria was established to facilitate tailorability among process models. The paper concludes with the optimized selection of process models most successful in meeting a variety of software projects requirements. It was also concluded that process diversity could be the rationale for integration rather than differentiation as to capture project needs in adaptive and flexible combinations. Finally, several frameworks were presented to address similarities, differences, comparisons, and evolutionary aspects among software process models.

Another research [5] also tried to address the problems that impacted software development processes in Ethiopia. According to the findings in [5], the software development situation is mostly dominated by failure cases characterized by: unrealized benefits, unsatisfied users, substantial budget and time overruns far beyond expected, frustrated developers, etc.

The effort resulted in the development of a comprehensive methodical approach known as Reflective Steps. The approach evolved from the process of tailoring the Software Technology for Evolutionary Participative System Development (STEPS) model and complementary aspects of contemporary methods and processes based on the contextual issues identified. In the process, home-grown collaborative techniques and proven practices in the areas of business process redesign for software development and project management were incorporated. Software development approaches and software development situations and practices in Ethiopia were also included.

The overall purpose of this research is to explore the possibility of developing suitable approaches to address the software development challenges in Ethiopia by tailoring available methods and process models.

Though the researches do not give much focus to the procurement directive, selecting proper and appropriate process model that fits to project problems at hand is clearly discussed.

3. Methodology

3.1 Data Collection

The methodology used for this research is descriptive method and has required survey method research which the participants used to answer questions administered through interviews and questionnaires in addition to the information and data extracted through literature review.

3.1.1 Extracting Process Models from Literature, Questionnaire, and Interview Survey

In this step, a literature survey of software process models is conducted, questionnaires and interviews were also administered to extract the major streams in process models.

Due to lack of readily available empirical data regarding the software industry in the country,

selection of participants was guided by a list of participants of the "ICT exhibition and bazaar 2004 E.C", compiled by the Ministry of Communication and Information Technology (MCIT).

The list shows that there are about 26 private local companies that are engaged in software development projects. Ten companies were selected using purposive sampling technique. That is, attempts were made to ensure the inclusion of most of the private firms that were actively involved in the development and customization of software systems. Companies whose major business was not software development or customization were excluded.

Questionnaires were sent to all (hand delivered as all of the respondents were in Addis Ababa) and to expedite the process, arrangements were made to physically collect the completed questionnaires. The questionnaire was accompanied by an explanatory covering letter that stated the purpose of conducting the survey, and how the findings will be used. An overall response rate of 80% was obtained from software development companies.

The interviews and discussions conducted for collecting data for this study included structured one-to-one interviews with selected software engineers, managers of IT Departments in selected organizations and software development companies. In particular, selected individuals from those that had already participated in the questionnaire survey were given more chance in the interviews to qualify their responses through providing explanations or examples.

3.1.2 Extracting Process Models from Literature Survey

In this step, a literature survey of software process models is conducted to extract the major streams in process models. Most of the well known software process models with regard to problem solving features were explored [3].

3.2 Data Analysis

3.2.1 Establishing Frameworks for Similarities and Differences

In this step, similarities, differences, relations and rationales among process models and the FPPD are explored and frameworks are presented to address how process models share common goals and how they also have different characteristics.

Although several factors contribute to the formation or development of process models, all process models aim to achieve common goals and share general characteristics regardless of their degrees of success or accomplishment.

3.2.2 Characteristics of Process Models

The following characteristics of the software process models are drawn from [6]

Maintainability: Baseline requirements, stable designs, consistent coding standards, baseline regression tests, and adequate documentation contribute to software maintainability. Highly critical and complex systems often have demanding maintainability requirements as they are very concerned about "total lifecycle costs" - both during development and during operation. On the other hand, ad-hoc programs for experimental and concept validation purposes care little about down-stream effort and cost effects.

Application Domain: High-end applications include safety, reliability, security and availability and other critical systems; moderate applications include business and enterprise applications; low-end (low criticality apps) include beta and field trial services, proof-of-concept prototypes, and vanilla-flavored web-sites.

Size/Complexity: For simplicity, let us consider that software size and complexity are relatively linear with respect to each other. Let's assume that large, complex software projects exceed 500K LOC, small, simple projects are less than 10K LOC, and medium-sized projects range in between.

Requirements Uncertainty: This too is difficult to quantify. But let us assume that requirements are highly uncertain if they are only verbalized or if they have been expressed in a few pages with little input from users. Meanwhile, highly certain requirements have involved some up front studies, user inputs, possibly have been prototyped, and have been scrutinized by third parties. Again, moderately certain requirements are somewhere in between.

Progress Visibility: Visibility into progress can be achieved through demonstrations and through documentation and both have their shortcomings. Demonstrations are favored by many customers but if not supported by other measures of progress, can give the illusion of more progress than is actually being achieved. Documentation is harder to translate into real progress but simplifies contracting – hence it is favored by those with accounting and legal mindsets. For this discussion we will assume that ongoing and frequent demonstrations to customers and users accompanied by reasonable levels of documentation can provide high progress visibility. And infrequent demos plus scant documentation provide little or no visibility into progress.

User Involvement: User involvement can be with respect to developing requirements specifications, validating requirement specifications, inspecting prototypes, supporting detailed design and development, reviewing specifications, and accepting released products. Users heavily committed to supporting three or more of these areas can be regarded as "highly involved". Marginal support of only one of the areas can be considered "low" involvement.

Requirements Volatility: This factor is also difficult to quantify and requires a good understanding of the problem domain, especially with respect to the maturity of the customer, users, and the development team. If the application is unprecedented or if the problem is in an emerging area, stakeholders will not be sure what will change until parts of the system have been fielded. Often, but not always, requirements uncertainty will go hand-in-hand with requirements volatility (if they are uncertain, they could very well be volatile later on).

Time Constraints: Demanding market forces will put pressure on a software project to create and release software functionalities and features as early as possible. This will lead to lots of schedule pressure. High urgency, of course, needs to be moderated for mission-critical projects when it comes to human safety and financial transactions where errors cannot be tolerated.

Despite of these commonalities, these process modeling approaches were neither introduced in the same period of time nor belonged to same schools or researchers, nor based on same prospective or the same availability of enabling technologies. Moreover, these approaches were not facing the same nature of problems. Therefore, from our previous survey, we can infer that many of the differences among previous software process models solutions can be attributed to one or more of the following influencing factors:

3.3 Data Representation

3.3.1 Building the Comparison Matrix for Software Process Models

Comparison matrixes between explored process models based on the major drivers of each process model and FPPD and its features will be introduced to establish a foundation for the grouping process.

3.3.2 Characterizing Each Lifecycle Process

Each lifecycle process in terms of the criteria expressed [3] is characterized in Table 1. To simplify the analysis, it is chosen to use three-level attributes for each as follows:

H=High; M=Moderate; L=Low … these terms can be directly interpreted except for:

- Application Domain where H=safety/reliability-critical; M=business; L=prototypical/experimental
- Size / Complexity where H=large and complex; M=moderate; L=small and relatively simple

Table 1: Software process models and lifecycle process criteria comparison table

Lifecycle Process Criteria	*Process Model*				
	Waterfall	*Incremental*	*Iterative/Spiral*	*Evolutionary*	*Agile/XP*
Maintainability	L	H	H	L	H
Application Domain	M, H	M, H	M, H	L, M, H	L
Size/Complexity	L, M, H	M, H	M, H	L, M, H	L
Uncertain requirement	L	L	L	H	H
Progress visibility	L	M	M	M	H
User Involvement	L	M	H	M	H
Requirements Volatility	L	L	M	H	H
Urgency	L	L	M	H	H

Table 2: Process model and FPPD features matrix

FPPD Characteristics	*Process Model*				
	Waterfall/V model	*Incremental*	*Iterative/Spiral*	*Evolutionary*	*Agile/XP*
Rapid Planning	H	L	M	L	L
Late Delivery	H	H	H	L	L
Unrealistic Schedule	L	L	M	H	H
Changing requirement	L	H	H	L	H
Requirement volatile	L	M	M	M	H
Risk avoidance	L	H	H	L	H
Obsolete technology	H	M	M	L	L
Fixed allocated budget	H	M	L	L	L
Time overrun	H	M	M	L	L

3.3 Testing the Hypothesis

Table 3: Matrix for testing the hypothesis

FPPD Attributes	*Process models*					*Sum*
	Waterfall	*Incremental*	*Iterative*	*Evolutionary*	*Agile*	
High	5	3	3	2	4	17
Medium	0	3	6	0	0	9
Low	4	3	0	7	6	20
Sum	9	9	9	9	9	46

The hypothesis: Existing software process models are not applicable in the Ethiopian context as the result of FPPD; and procurement procedures should be considered while defining process model.

To test, we apply the statistical test called χ^2(chi square)

$\alpha = 0.01$ significance level

Ho = Process models are applicable to the context of FPPD (Process models are independent of FPPD)

H_1 = Process models are not applicable (Process models are dependent on FPPD) (3-1) (5-1) = 8 degree of freedom gives us a critical value of 20.09.

To reject H_1, we need $\chi^2 \geq 20.09$

5+3+3+2+4= 17, 3+6= 9, 4+3+7+6= 20

5+3+ 3+……. +7+6 =46

e_{11}= (16) (9)/46= 3.1 e_{21} = (9) (9)/46= 1.8 e_{31} = (20) (9)/46= 4

e_{12}= (16) (9)/46 =3.1 e_{22} = (9) (9)/46= 1.8 e_{32} = (20) (9)/46= 4

e_{13}= (16) (9)/46 =3.1 e_{23} = (9) (9)/46= 1.8 e_{33} = (20) (9)/46= 4

e_{14}= (16) (9)/46 =3.1 e_{24} = (9) (9)/46= 1.8 e_{34} = (20) (9)/46= 4

e_{15}= (16) (9)/46 =3.1 e_{25} = (9) (9)/46= 1.8 e_{35} = (20) (9)/46= 4

We now use these to calculate our test statistic, which gives us:

$X^2 = (5\text{-}3.1)^2/3.1 + (3\text{-}3.1)^2/3.1 + \ldots\ldots\ldots (0\text{-}1.8)^2/1.8 + \ldots\ldots\ldots (6\text{-}4)^2/4 = 23.2$

Since $X^2 = 25.5 > 20.9$

We reject Ho; Process models are not applicable in the context of FPPD.

Table 4: Process model and FPPD characteristics matrix

Characteristics of Process model & FPPD	*Process Models*				
	Waterfall/V model	*Incremental*	*Iterative/spiral*	*Evolutionary*	*Agile/ XP*
Rigid planning	H	L	M	L	L
Late delivery	H	H	H	L	L
Unrealistic schedule	L	L	M	H	H
Changing requirement	L	H	H	L	H
Requirement volatile	L	L	M	H	H
Risk avoidance	L	H	H	L	H
Obsolete technology	H	M	M	L	L
Fixed allocated budget	H	M	L	L	L
Time overrun	H	M	M	L	L
Application domain	M, H	M, H	M, H	L, M, H	L
Size/Complexity	L, M, H	M, H	M, H	L, M, H	L
Uncertain requirement	L	L	L	H	H
Urgency	L	L	M	H	H
Maintainability	L	H	H	L	H
User involvement	L	M	H	M	H
Progress visibility	L	M	M	M	H

4. Findings

Data Grouping: Class Diagram Taxonomy

In this step, a final framework of process models classification is concluded based on the previous steps. This framework is diagrammed in terms of the amount of attributes in the matrix (High, Medium and Low) of Process model and FPPD characteristics matrix.

Table 5: Summary of the process model and attributes matrix

Attribute	*Process model*				
	Waterfall	*Incremental*	*Iterative*	*Evolutionary*	*Agile*
High	7	6	7	6	9
Medium	2	7	9	4	0
Low	10	5	2	10	8
Total	19	18	18	20	17

By rating the attributes H, M, L with 1, 2, 3 respectively, we obtain the following matrix

Table 6: Summary of the rated matrix

Attribute	*Process model*				
	Water fall	*Incremental*	Iterative	Evolutionary	Agile
High	7	6	7	6	9
Medium	4	14	18	8	0
Low	30	15	6	30	24
Total	41	35	31	44	33

Value of each process model can be calculated by the formula:

$V = \Sigma_i (i = 1)^n [(Hr1 + Mr2 + Lr3)]$)

where r1, r2 and r3 are rates given to the attributes H, M and L, respectively, V is value of each model and H, M, L are high, medium and low, respectively.

To this effect:

Iterative model = 31

Agile model = 33

Incremental model = 35

Waterfall model = 41

Evolutionary model = 44

Grouping the software models according to their capability and strength can contribute much in capturing important features of different models so that can tackle software development difficulties and may smooth the project development process.

According to the evaluation, iterative model can be suited better to a project with the situation of non flexible software procurement directive. But using process models in combinations might have good effects if integrated efficiently.

Although the iterative approach as spiral model was initiated independently and focused on risk management, it is wise if incorporated with several other process models. To that effect it is attempted to group the most suitable model with the iterative model according to the study with the alternative consecutive ones.

Iterative + Agile

Iterative + Incremental

Iterative + Waterfall

Iterative + Evolutionary

The scale and the optimal characteristics presented here are based on knowledge of each development approach from the literature and experience. The intent of the proposed framework is not to serve as a silver bullet but to show up by combining appropriate models which certainly have different capabilities can tackle different development problems at hand and one should not take the presented alternatives as given, but rather as a starting point for this kind of quantitative assessment and for future research.

The principles of different approaches like the linear or phased, iterative, and agile approaches with their embedded process models are thoroughly discussed earlier and the critical factors which are the main features of each approach to the context of the research are also illustrated in the comparison matrix above. These features basically are true to the embedded process models.

All these different software development models have their own advantages and disadvantages. Nevertheless, in the contemporary commercial software development world, the fusion of all these methodologies is incorporated.

To have a final understanding and agree with the proposed solution, it is proper to validate and is attempted to gather opinions of the software companies in this regard.

5. Conclusion and Future Work

The objective of this paper is to find a way for choosing appropriate software process model(s) that can go with the current local procurement directive. The consequence of inappropriate choice of process model(s) for a given software project is grievous - low quality software product delivered out of time and budget.

The work presented in this paper shows that some features are common to multiple process models. These common content could form the core of a

hybrid model, to which would be added the feature unique to specific models. This would retain the feature elements considered important by all models, and augment this with additional features to incorporate the strengths of multiple models which are currently incompatible.

The survey reveals that the FPPD, the tender document, and tendering process have to be revised so that negative impacts of these particular contributors on the software development process will diminish. The suitability of agile methodologies which have flexible feature goes with the non flexible software procurement directive seems to be controversial and needs future work. The software tender process which is believed to affect the software development process and the means of payment in the contracts where there is requirement change also needs investigation and is left to the future work.

References

[1] Ian Somerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[2] Federal Public Procurement Directives, Ministry of Finance.

[3] Kal Toth, “Which is the Right Software Process for your Problem?”, Portland State University.

[4] Osama Eljabiri and Fadi P. Deek, “Tailoring the Software Process Model to Project Requirements”, College of Computing Sciences, New Jersey Institute of Technology, Newark.

[5] Tesfaye Biru, “Reflective Steps: A Collaborative Learning Oriented Approach to Software Development and Process Improvement”, Oct. 2008.