Linear models vs Agile models: Making the right model decision

Aline U. Hakizabera*, Koichi Yamada (Nagaoka University of Technology)

Abstract
The selection of the right development model should be one of the major steps to minimize the risk of failure in the software development project. However, most of the developers choose to use a model based on routine or popularity which can lead to problems in the development. This paper examines the different attributes of linear models and agile models and proposes a decision process to choose the right model based on the software projects characteristics.

Keywords: Linear models, Agile models, Decision making, Dempster Shafer theory, Software development project, Risk management

1. Introduction

With the increasing demand of customers and the high level of competition, the software industry has developed a variety of tools and methods to improve the quality of software development projects. However, different studies and surveys have revealed that a significant percentage of software development projects fail to be completed on time, to budget and to their initial requirements (Kweku, 2003; Standish group , 2005). One of the main challenges of the software industry is to choose from the very beginning the right tools and techniques for developing their software. The first step to reduce risk of failures would be to correctly select the right models and tools for software development projects from the early stage of the life cycle (Tailor, 2004).

The development of a software project involves a conceptual model that describes the phases in a software system from the initial conception to the completed application (Oxford, 2006). That conceptual model is generally referred as the Software Development Life Cycle (SDLC). A SDLC comprehends a series of phases that can vary between four and ten. In general, an SDLC have the following phases: planning, design, coding, testing and integration, installation and maintenance and disposal. The SDLC is tackled with different models that can be categorized in two big groups: linear models and agile models.

The linear models referred also as traditional models have been used as early as 1960s to methodologically solve problems related to the software developments (Lewis, 2008). Winston Royce (1970) formally introduced in 1970, a linear model later identified as the waterfall model. However, he recognized it to be a flawed method of software development because of its shortcomings. Among the weakness of the linear models, Taylor (2004) cited inflexibility in regards of the change of business needs, heavy documentation and low business value. Despite the problems identified, the linear approach is still widely used in the software industry (Davidson, 2008). Some of the positive points of the linear approach are that it keeps the software development under control; it is easier to predict the budget and it enables to review the progress at the end of each phase.

On the other hand, some developers advocate strongly the use of agile models for software development projects (B. Kent et al., 2001). The agile approach was formally introduced by Edmonds in his research paper in 1974 (Edmond, 1974). But, agile models (such as 'Scrum', 'Adaptive Software Development', 'Extreme Programming'…) started to evolve in the 1990s when developers decided to break away from rigid approaches and move towards more flexible development styles. However, some of the agile approach problems are that it lacks of up-front planning, lack of management control and lack of predictability.

"The best model" does not exist; each SDLC model has different advantages and disadvantages. Some developers select a model because of habit and others because of novelty. However, trying to accommodate a software development project to a model that doesn't fit the appropriate conditions leads to the path of failure.

In this paper, we examine the features of the different models of the SDLC used in software development projects. In addition, we examine the decision criteria to choose the right model for a designated software project. We propose the use of Dempster-Shafer theory to support the decision of the manager under incomplete information or uncertainty. Our work is limited to two approaches that we divided in linear and incremental models for the linear approach and iterative, adaptive and extreme models for the agile approach.

2. Software development models

The features of the linear and agile approach are described briefly through a series of advantages and disadvantages as shown in Table 1 and Table 2. A model should accommodate the characteristics of a software project to maximize the success of its implementation.

2.1 Linear approach

The linear approach requires a well-planned strategy from the early stage of the project. Two models can be extracted:

1) Linear model: It consists of a number of dependent phases that are executed in a sequential order with no feedback loops. It produces a solution only at the final phase. (Example: Waterfall model).

Software project characteristics:

- *Well-defined goals*
- *Well-defined requirements and solution*
- *Few scope change requests*
- *Routine or repetitive projects*

2) Incremental model: Same as the linear model except that each phase releases a partial solution or deliverable (Example: Feature-Driven Development model).

Software project characteristics:

- *Well-defined goals*
- *Well-defined requirements and solution*
- *Need to release deliverables against a more aggressive schedule*

Table 1: Features of Linear approach models

	Advantages	Disadvantages
Linear	-Project under control -Milestones are known and tracked -Resource requirements are known -Work distribution -Work well with inexperienced developers	- Inflexibility - Long period of development -No business value until late in the development - Heavy documentation - No focus on customer value
Incremental	-Early business value in the development life cycle -Better use of scare resources -Possible change requests adjustment -More focus on customer value than linear model	-Rigorous set of processes -Heavy documentation -Function and features increments interdependencies -Partition of functions and features can be problematic

2.2 Agile approach

In the agile approach, planning keep changing along the iteration until a satisfactory result is achieved. Three models can be extracted:

1) Iterative model: It consists of a number of phases that are repeated in groups with a feedback loop after each group is completed. At the discretion of the customer, the last phase in a group might release a partial or final solution (Example: SCRUM, Rational Unified Process model).

Software project characteristics:

- *Well-defined goals*
- *Not all features of the solution are known but most of the functions are known*
- *Learn -by-doing strategy.*

2) Adaptive model: progresses from iteration to iteration based on limited specification of the solution. Each iteration learns from the preceding ones and redirects the next iteration in an attempt to converge on an acceptable solution for the customer. (Example: Adaptive Software Development model, Adaptive Project Framework model).

Software project characteristics:

- *Well-defined goals*
- *Solution is only partially known, most of the functions and features are still to be defined*
- *Frequent change requests and just in time planning.*

3) Extreme model: Same as the adaptive model except that the specification of the solution is minimal and the goals of the software project are not clearly defined. (Example: Extreme programming, INSPIRE model).

Software project characteristics:

- *Goals are not known*
- *The solution is unclear*
- *Typical of Research & Development projects.*

Table 2: Features of Agile approach models

	Advantages	Disadvantages
Iterative	-Early and frequent review -Accommodation to changes between iterations -Adaptation to changing business conditions	- Need more customer involvement - Unclear final solution
Adaptiv	- No time wasted on non-value added work - Significant business value within the given time and cost constraints	- Need meaningful customer involvement - Difficulty to identify what will be delivered at the end of the project
Extreme	- Rapid feedback - Continuous review and testing - Maximum business value within the given time and cost constraints	- No focus on long-term goals at early stage -Need constant customer involvement -Lack of design documentation -Lack of quality plan - Uncertainty on the final project

Figure 1 shows the dynamic evolution of phases from linear (1) to extreme (4) through iterative (2) and adaptive (3) models. The more we move from a model to another the shortest are the period of iteration and the deepest can the changes be applied.

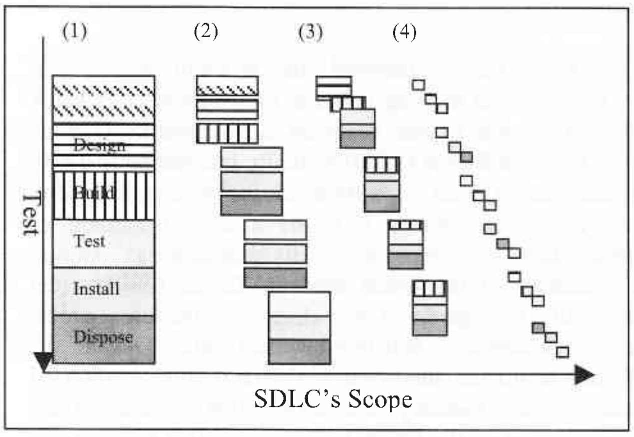

Figure 1. From Linear to Extreme model

3. Decision making on the right model

3.1 Decision process

A decision process involves a series of information collection, judgment and evaluation. Figure 2 illustrates simplified rules to support the decision process of the right model.

The project manager (PM) or the decision maker (DM) has to base his decision on the characteristics of the software project and the information on the whole project. When the goals are not well-defined, the project points directly to the extreme model because without goals, there are no directives and no knowledge on the requirements. The goals have to be defined after several initiatives. On the other hand, when the goals are well-defined, the DM has to evaluate if the solution or the requirements are complete enough. If yes, the DM has to choose the incremental model if there is a need of early release of deliverables otherwise he has to select the linear model. If no, the DM has to evaluate to what extent the requirements are defined. If most of the functions are available, the DM has to select the iterative model but if most the requirements are still have to be defined the adaptive model has to be selected.

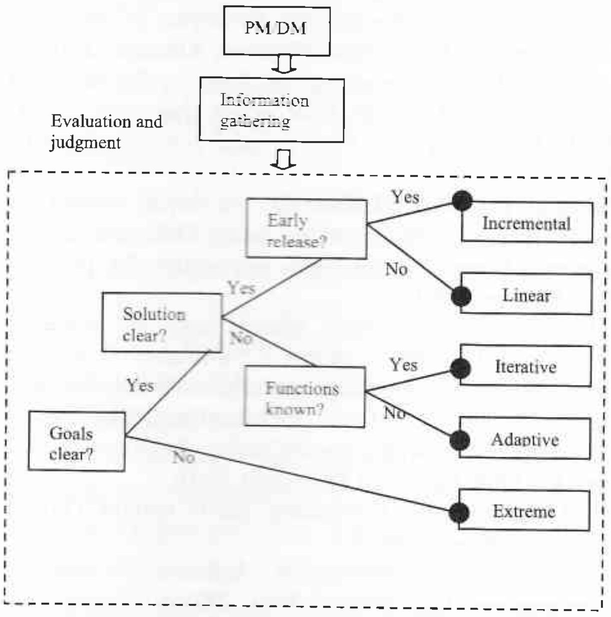

Figure 2. Decision process

3.2 Decision making with Dempster-Shafer theory

To select an appropriate model, the DM has to make a decision according to available data on the solution and goals of software projects. However, realistically on early stage of the software project, the DM does not have complete knowledge on the project and some information might not be available. The other problem is that the DM might face the difficulty to evaluate with precision the clarity of the goals and the solution.

Decision making with Dempster-Shafer theory of evidence (DST) can be used to overcome the uncertainty and or the lack of information on the software project at the moment of the decision. DST was first introduced by Dempster in the 1960s (Dempster, 1967). It was later expanded by Shafer in 1976 (Shafer, 1976). DST can be interpreted as a general extension of Bayesian theory that can manage incomplete date. Its unifying framework to represent uncertainty can also include cases of risk and ignorance. The basic probability assignment function (bpa or *m*), the Belief function (Bel), and the Plausibility function (Pl) are the three important functions of the DST.

For X, a given universal set that represents a set of all possible states where P (*X*) represents the power set of *X* and A is a set in the power set (*A*∈P (*X*)), *m* can be represented as follows:

$$m: P(X) \rightarrow [0,1] \qquad (1)$$

$$m(\emptyset) = 0 \qquad (2)$$

$$\sum_{A \in P(X)} m(A) = 1 \qquad (3)$$

The value of *m*(*A*) expresses the proportion of all relevant and available evidence that supports the claim that a particular element of *X* (the universal set) belongs to the set *A* but to no particular subset of *A* .

Based on the basic probability assignment, certain measures of confidence can be defined; precisely the upper and lower bound of a probability interval. This interval contains the exact probability set of interest and is bounded by two non-additive continuous measures called *Belief* (Bel) and *Plausibility* (Pl).

A *Belief* measure (lower bound) *for* a set *A* is defined as the sum of all the basic probability assignments of the proper subsets (*B*) of the set of interest (*A*) (*B* ⊆ *A*):

$$\mathrm{Bel}(A) = \sum_{B \mid B \subseteq A} m(B) \qquad (4)$$

A *Plausibility* measure (upper bound), is the sum of all the basic probability assignments of the sets (*B*) that intersect the set of interest (*A*) (*B* ∩ *A* ≠ ∅):

$$\mathrm{Pl}(A) = \sum_{B \mid B \cap A \neq \phi} m(B) \qquad (5)$$

As there are related to each other, *Plausibility* can be derived from *Belief* and vice-versa:

$$\mathrm{Pl}(A) = 1 - \mathrm{Bel}(\bar{A}) \qquad (6)$$

where $\bar{A}$ is the classical component of A.

The basic probability assignment m(A) can be obtained from a given *Belief* measure Bel (B) for all subsets B and A where |A–B| is the difference of the cardinalities of the two sets:

$$m(A) = \sum_{B \mid B \subseteq A} (-1)^{|A-B|} \mathrm{Bel}(B) \qquad (7)$$

Using Dempster's rule of combination, we can combine measures of evidence from different independent sources. For two basic probability assignments m_1 and m_2, the combination is calculated as follows (Shafer, 1976):

$$m_{12}(A)=\frac{\sum_{B\cap C=A} m_1(B)m_2(C)}{1-K} \qquad (8)$$

when $A \neq \varnothing$
and $m_{12}(\varnothing)=0$

where $K=\sum_{B\cap C=\phi} m_1(B)m_2(C)$

K is the measure of the amount of conflict between the sets of evidence.

DST has been proved to be useful to express uncertain judgments of decision makers. It would certainly support the decision of choosing the right SDLC model. According to decision making under uncertainty, the DM defines the alternatives from which he has to make a choice. The outcomes of the decision are the combination of the alternatives and the states of nature which are factors affecting the decision (Hanssen, 1994). The set of alternatives is defines as $A=\{a_i|i=1\ldots n\}$, the set of states of natures is defined as $S=\{s_j|\ j=1,\ldots m\}$, the outcome set $O=\{o_{ij}|\ o_{ij}=f(a_i, s_j)\}$ and the utility function $u_{ij}= u(o_{ij})$.

From the two approaches linear and agile, the set of SDLC model alternatives in the decision process is as follows: A={Linear, Incremental, Iterative, Adaptive, Extreme}.The goals and the solution which clarity are sometimes uncertain have a strong impact on the alternatives and can be chosen to characterize the states of nature. The goals domain can be defined as (clear, somewhat clear, not clear) and the solution domain can be defined as (fully clear, mostly clear, somewhat clear, not clear). The outcomes would be defined and then replaced by their utility values to allow evaluation. The basic probability assignments values are considered. As sometimes the DM does not know the complete probability distribution of the state of nature, the probalitities from bba values are approximated using OWA operator. Nusrat (2010) has demonstrated a method to convert decision making under uncertainty to decision making under with expected utility (GEU) and OWA to decision making under risk using DST and the Prospect theory. Furthermore, the Prospect theory is used to transform the decision making into a descriptive decision makinf model under uncertainty.

4. Conclusion

In this paper, we have discussed the decision criteria to choose the right SDLC model by taking into account the features of each model in either the linear approach or agile approach and the characteristics of the software project. Decision making using Dempster-Shaffer theory has shown to give a support to the decision maker in situation of uncertainty and incomplete data especially at early stages of the software development projects. In the continuing research, we are examining the best support that DST can provide in decision making of the SDLC models.

References

A.P Dempster, "Upper and Lower probabilities induced by multi-valued mapping", Annals of mathematics Statistics, vol. 38, pp 325-339, 1967.

Beck, Kent; et al. , "Manifesto for Agile Software Development". Agile Alliance available at:. http://agilemanifesto.org/., 2001. (accessed 05 September 2010).

E. A. Edmonds, "A Process for the Development of Software for Non-Technical Users as an Adaptive System". *General Systems* 19: 215–18, 1974.

E.M. Kweku, *Software Development Failures: Anatomy of Abandoned Projects*, The MIT Press, 2003.

E. Nusrat, "A Descriptive model of Decision Making under Uncertainty using Dempster-Shafer theory and Prospect Theory", Master thesis, Nagaoka, 2010.

G. Shafer, *Mathematical Theory of Evidence*, Princeton, NJ, 1976.

J.Taylor, *Managing Information Technology Projects: Applying Project Management Strategies to Software, Hardware, and Integration Initiatives*, Amacom, 2004.

L.Lewis, *SDLC 100 success secret, Software Development Life Cycle (SDLC) 100 Most asked Questions, SDLC Methodologies, Tools, Process and Business Models*, 2008.

M.Beynon, D.Cosker, D.Marshall, "An expert system for multi-criteria decision making using Dempster Shafer Theory", Expert Systems with Application 20, pp 357-367, Elsevier, 2001.

M. Davidson, Survey: Agile interest high, but waterfall still used by many, available at : http://searchsoftwarequality.techtarget.com/news/article/0,289142,sid92_gci1318992_mem1,00.html?ShortReg=1&mboxConv=searchSoftwareQuality_RegActivate_Submit& , 2008. (accessed 28 August 2010).

Oxford, *Dictionary of Computing, fourth edition*, Oxford University Press, 1996.

R.K.Wysocki, *Effective Software Project Management*, John Wiley & Sons, 2006

S.O. Hanssen, Decision Theory- *A Brief Introduction, Royal Institute of Technology*, Stockholm, 1994.

Standish Group, *The Standish Group Report: Chaos*, 1995.

W. W. Royce, "Managing the Development of Large Software Systems", Proceedings, IEEE WESCON, pg 1-9, 1970.