3-2006

Agility measurement index: A metric for the crossroads of software development methodologies

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Agility Measurement Index – A Metric for the Crossroads of Software Development Methodologies

Subhajit Datta
Department of Computer Science and
School of Computational Science
Florida State University
Tallahassee, FL 32306, USA
datta@cs.fsu.edu

ABSTRACT
Software engineering's journey to maturity has been marked by the advent of different development methodologies. While each paradigm has its context and cognoscenti, project teams are often faced with the choice of one approach over another in the grind of delivering software on time and within budget. In this paper, we briefly review the three major techniques of addressing enterprise software development, namely the Waterfall, Unified and Extreme styles. The metric *Agility Measurement Index* is then proposed, which helps organizations choose the methodology that best suites a particular project.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *complexity measures, process metrics.*

General Terms
Algorithms, Management, Measurement, Design, Economics, Reliability, Human Factors, Standardization.

Keywords
Waterfall, Unified Process, Agile Methods, Extreme Programming, Metrics.

1.INTRODUCTION
"In the beginning there was the waterfall" [1]. This technique prescribed software be built in a succession of clearly defined and demarcated sets of activities covering requirement specification, analysis, design, implementation and testing [2]. The implicit assumption was everyone knew every relevant detail *a priori*; customers knew what system they wanted and what the system wanted from them, analysts knew what they heard from the customers was what the customers wanted to tell them, designers knew they could get the design right the first time, implementors knew all they had to do was to translate the design into code, and testers knew what to test. In the Waterfall model projects progressed in a linear unidirectional path, like the eternal truth of water flowing downhill. In spite of all the inadequacy ascribed to the Waterfall model later – often justifiably – its value lies in the first semblance of order it sought to introduce in the hitherto *free-form* and instinct driven pursuit of software development.

ACM SE'06, March, 10-12, 2006, Melbourne, Florida, USA

The Unified Software Development Process (aka Unified Process or UP) took the best idea of the Waterfall model and made it even better. Software Development Life Cycle (SDLC) was now a two dimensional [3] matrix of phases – Inception, Construction, Elaboration, Transition – and workflows Requirements, Analysis, Design, Implementation, Test. The Unified Process is use-case driven, architecture-centric, iterative, and incremental [4]. In essence, UP places great emphasis on understanding the scenarios of user interaction with the system, culturing an architectural framework that supports reusabilty and extensibility, and building software iteratively and incrementally. It recognizes that getting it right the first time is an absurd chimera for anything other than trivial systems, and seeks to absorb effects of changing user needs through awareness and coordination.

Extreme Programming (XP), almost eponymously, takes one more radical step in the building of enterprise software. It is one – perhaps the most promising – among a gamut of "agile" methods, that "...attempt to offer once again an answer to the eager business community asking for lighter weight along with faster and nimbler software development processes" [5]. It repositions the conventional software process sideways. "Rather than planning, analyzing, and designing for the far-flung future, XP programmers do all of these activities – a little at a time – throughout development" [1]. The XP major practices, called the "circle of life" [6] such as *Planning game*, *Small releases*, *Metaphor*, *Simple design, Tests*, *Refactoring*, *Pair programming*, *Continuous integration*, *Collective ownership*, *On-site customer*, *40-hour weeks*, *Open workspace*, *Just rules* etc. are unconventional and exciting perceptions of new ways of building software in-the-large, as hinted by their maverick names.

All of the above methodologies embody key insights of software engineering that have been learned through collective experience, often at the cost of individual heroics, or martyrdom. It is vacuous to dwell upon the superiority of one method over another; every approach has a specific scope and facility. A common problem of building software for customers is to decide which methodology to adopt for a particular project. This decision, necessitated by schedule and budget constraints has to be taken very early in the SDLC, and once taken, has to be adhered to. Thus the choice is of major consequence to the project's final outcome.

In this paper, we propose a metric, the *Agility Measurement Index (AMI)*, which can serve as a heuristic to decide which methodology is the best fit for a given project. The next section

highlights the theme of agility in the desiderata of different software development strategies. We then present the idea of the metric and follow up with its derivation. The usage scenarios of the metric are outlined subsequently. We conclude with a summary and directions of future work.

2.THE METHODOLOGY FRAY

The evolution of software development processes points to a natural progression as one methodology begets another. A key theme in the genesis of every new model is the need to better understand, evaluate and manage *change* even as software is designed and built. It is a fact of life that requirements – the principal driver of a software system – will undergo change [7]; customers will change their mind, their perception of the role of the software will change, the environment in which the software operates will change and so will the technology with which the software is built.

The most important aspect of a successful software process is its ability to coordinate and control the effects of such changes. The word *agility*, though applied only recently in the context of software development, reflects a lasting holy-grail of software development methodologies – the capacity of adapting to and delivering in spite of, change.

Waterfall, UP and XP all have their own ways of embedding agility into the process; each with concomitant advantages and drawbacks. Even the latest "agile" methods, designed to deliver from the quagmires of earlier approaches, raises concerns about their supposed dependence on "premium people" (perhaps evoking wraiths of Nietzche's supermen or Huxley's Alphas !) [8]. There is abounding consensus on an elusive "synthesis" between methods [9], [10] without concrete ways to realize it.

3.AND THE NEED FOR A WAY

As a development organization engages with customers to deliver a software project under predetermined cost and time constraints, it faces the dilemma of which methodology to follow. There are no ready answers, as the decision needs to take into account a wide swath of factors and their combinations; and even situations which can not be envisioned upfront.

We now derive the *Agility Measurement Index (AMI)*, which seeks to streamline the decisioning process.

3.1Agility Measurement Index – An Indicator Metric

Intuitively, let us describe *Agility Measurement Index (AMI)* as an indicator metric for determining whether a software development project is best suited to the Waterfall, UP or XP development methodologies. At the end of this section we will reach a formal definition of *AMI*.

Let us define the following as the *dimensions* of a software development project.

- **Duration (D)** – From project inception, how far ahead in time is the delivery deadline ?
- **Risk (R)** – What is the impact of the project deliverable in its usage scenario ? Is it mission critical, like a hospital patient monitoring system, moon rocket controller; or is it meant for relatively less *razor-edge* use ?
- **Novelty (N)** – Does the project involve a domain where the users have never used a software before or the developers are looking to use new and untested technology ?
- **Effort (E)** – How much effort, in person-hours, is the customer willing to support and the development organization prepared to spend over the project duration ?
- **Interaction (I)** – What is the level of regular interaction between the development team and the customer ? Daily meetings ? Weekly ? Monthly ? Or is the customer only interested in seeing the finished product ?

Each dimension is given an *Actual score(A)*, on a scale between a *Min score(N)* and a *Max score (X)*. Choice of the range between *N* and *X* is based on the degree of granularity needed for a particular dimension.

The *Agility Measurement Index (AMI)* is formally defined as,

$$AMI = \frac{(\sum Actual\ score\ for\ each\ dimension)}{(\sum Maximum\ possible\ score\ for\ each\ dimension)}$$

We define the *Specific Dimension(SD)* for each dimension as the ratio of *Actual score* and *Max score*.

Calculations for a hypothetical project is shown in Table 1.

Table 1. Sample calculation of Agility Measurement Index (AMI) and Specific Dimension(SD)

Dimension	N	X	A	SD = A/X
Duration (D)	1	3	1.5	0.5
Risk (R)	1	5	2.5	0.5
Novelty (N)	1	4	1	0.25
Effort (E)	1	6	5	0.83
Interaction (I)	1	10	7	0.7

AMI = (1.5 + 2.5 + 1 + 5 + 7) / (3 + 5 + 4 + 6 + 10)
= 17 / 28
= 0.61

4.INTERPRETING THE METRIC

As stated earlier, the *AMI* is an indicator metric. A low value of *AMI* signifies the project is of short duration, low risk, low novelty, limited effort and with minimal customer interaction. Readily, the Waterfall model suggests itself as a suitable approach. However, for higher values of the *AMI*, the choices between UP and XP are not that apparent. In such cases, we take recourse to the *Specific Dimension (SD)* as calculated in Table 1. Projects with high AMI and high *SD* for the dimensions Duration (D) and Risk(R) are likely candidates for an UP approach, whereas those with similar *AMI* and high *SD* for Novelty(N) and Interaction(I) are best tackled through XP. Certain paradoxical situations may arise due to arbitrary choices of the *Max score (X)*. For example, it is possible to have some very high values in some fields, but still a low value of *AMI*. The only guarantee against such cases is to appreciate that assignment of the scores in the *AMI* calculation is best done by experienced analysts and designers with a clear vision of the project's context – the *Max*

score (X) needs to be decided on the required granularity for the dimension.

It must be underscored, *AMI* is not merely a *number* to blindly commit a project to a methodology. The metric needs to be interpreted in the light of a project's background and future direction. An element of subjectivity is fundamental to calculating and analyzing *AMI* results and talent at this task is honed through experience.

5.CONCLUSION

In this paper, we reflected on the crossroads of different methodologies every software development enterprise finds itself in. To alleviate the situation, we have proposed the metric *Agility Measurement Index (AMI)* to gauge the level of adaptability to change required for a project's success, and help decide on a suitable process thereon. A sample calculation of the *Agility Measurement Index (AMI)* along with broad suggestions on interpreting the metric have also been given. For further development of this idea, we look to incorporate the *Agility Measurement Index (AMI)* within analysis and design artifacts. We believe the *Agility Measurement Index (AMI)* can be applied to notable effect in enterprise software development.

6.ACKNOWLEDGMENTS

I would like to thank Dr. R. van Engelen for assistance inpreparing this paper. This work is supported in part by the Department of Energy grant DEFG02-02ER25543.

7.REFERENCES

[1] Beck K. Embracing Change with Extreme Programming. *IEEE Computer*, October 1999.

[2] Tilley T. , Cole R. and Becker P. and Eklund P. A Survey of Formal Concept Analysis Support for Software Engineering Activities. *Proceedings of the First International Conference on Formal Concept Analysis – ICFCA'03*. February, 2003.

[3] Schach S. *Object-oriented & Classical Software Development, Sixth Edition*, McGraw-Hill International Edition, 2005.

[4] Jacobson I., Booch G., Rumbaugh J. T*he Unified Software Development Process*. Addison-Wesley, 1999.

[5] Abrahamsson P., Warsta J., Siponen M., Ronkainen J., New directions on agile methods: a comparative analysis. *Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon*, 2003.

[6] Newkirk J. Introduction to Agile Process and Extreme Programming. *ICSE'02*, May 2002.

[7] Fowler M.. The New Methodology. *http://www.martinfowler.com/articles/newMethodology.html.*

[8] DeMacro T. and Boehm B. The Agile Methods Fray. *IEEE Computer*, June 2002

[9] Boehm B. Get Ready for Agile Methods, with Care. *IEEE Computer*, January 2002.

[10] Beck K. and Boehm B. Agility through Discipline : A Debate. *IEEE Computer*, June 2003.