The Dilemma of Iterative Software Processes: A Software Abstraction Levels vs. Agility Perspective

Reuven Gallant[1] and Leah Goldin[2]

[1]Jerusalem College of Technology, P.O.B. 16031, 91160, Jerusalem, Israel

[2]Shenkar College of Engineering and Design, 52526, Ramat-Gan, Israel

Abstract. The present paper is the product of the authors overall research regarding the efficacy of the software development processes in general and iterative processes in particular. In the context of this research the authors propose taxonomy of process features, the first of which is "Presentational," the primary focus of this paper. The authors acknowledge the grain of truth in Robert Glass's iconoclastic dismissal of the process re-engineering endeavors. However, while the "silver bullet" may be illusory, the issues motivating process re-engineering genuine. There is an inherent contradiction between the ways individuals naturally work iteratively (i.e. in an agile manner) and the drive for "moving forward". In this paper the authors examine several process models, and make observations regarding the effect of abstraction levels and their symbols on the ability to think and act iteratively and effectively harness process in the service of development progress.

1 Introduction

In an insightful editorial, written over a decade ago, Robert Glass [1] irreverently suggests that the claims for improved effectiveness, profitability, etc. to be gained for each newly proposed iterative process models, smacks of hucksterism. Glass argues that software life cycles are but a specific instance of the "universal algorithm of problem-solving" manifest in all "Professions."

Insofar as the generic characteristics of the problem-solving algorithm are invariant, having stood the test of time, it is unlikely that it can be replaced, in the guise of a new software life cycle, by anything better. The universal problem-solving algorithm should be taught in a "discipline-independent core curriculum rather than in a profession-specific one." Once its truisms have been apprehended, practitioners will stop wasting their energy trying to circumvent the essential difficulties of software construction. Rather they will confront these difficulties head-on. So argues Glass.

1.1 Motivation

We concur with Glass's assessment with the hucksterism inherent in software life

Gallant R. and Goldin L..
The Dilemma of Iterative Software Processes: A Software Abstraction Levels vs. Agility Perspective.
DOI: 10.5220/0003701301000107
In *Proceedings of the 2nd International Workshop on Software Knowledge* (SKY-2011), pages 100-107
ISBN: 978-989-8425-82-9

cycle innovation. However, the need to address new life cycles is deeper than those reflected in the materialistic claims of each generation of innovators. Software life cycle innovation reflects a deep aspiration to mitigate the angst inherent in the confrontation of elements that are inherently contradictory. The nomenclature may vary-discipline vs. agility [2], security vs. risk taking, familiarity vs. innovation, order vs. chaos, orderly documentation vs. impromptu spontaneity, collectivity vs. individuality, each pair of terms evocative of an irresolvable tension.

The terminology varies, but all software practitioners (and other professionals) know the feeling. Each new software life cycle attempts to achieve a balance between the contradictory poles, a beneficial synthesis. But the truce is uneasy and unstable. The individual and the organization inevitably find themselves sliding toward one pole or the other, and resolves to fix things by adopting, or creating yet another software life cycle, perhaps skewed in the other direction, perhaps more "epistemologically correct" so that we will be better prepared the next time that the specific circumstances that broke the truce reoccur.

Don't worry-next time it will be other circumstances, as before they will overpower our resolve to withstand the contradiction.

1.2 Overview

The tacit claim that yet another software life cycle will cure the angst engendered by irrevocable contradiction is the ultimate hucksterism. However, the act of creating, selecting or modifying a new life cycle is a legitimate and even vital means of mitigating this angst. To do so effectively, one must be conscious of the features that contributed to or mitigated this angst in the past. In the paper we present taxonomy of such features, illustrating them with examples of iterative process models a variety of process models. The taxonomy includes: presentation, activities per iteration, progress, and governance issues like roles, planning and control. We suggest a separation between presentation and more substantive process characteristics: activities, progress and governance.

In our investigation of several process models, we shall show how the presentation format influences both the dilemma posed by the particular process model and the relevant taxonomy of other characteristics.

1.3 Outline

Once the iterative software process, motivation and overview have been presented above, the rest of the paper describes taxonomy for iterative process adoption that was chosen, and analyses different process models accordingly.

Section 2 gives presents the taxonomy for iterative process adoption. Section 3 describes iterative process models. Section 4 analyses the process taxonomy related to the software process models presented in Section 3. Finally Section 5 concludes.

2 Taxonomy for Iterative Process Adoption

We present herein taxonomy for iterative processes that includes features that contributed to or mitigated the angst. Since each generation of process innovation addresses the same issues, it is useful to have a taxonomy to guide us in differentiating the various models, assess the commonality and originality of each, and ultimately guide us in selecting and modifying a process in response to specific, cultural, technical and business imperatives. The taxonomy includes:

Presentation. The process description that can be textual, sequential flow, spiral, circle, multiple circles, or bar graph.

Iteration Activities. How many activities per iteration are there, can some activities be skipped or weighted differently in successive iterations.

Progress. What kind of goals are there for each iteration, were they realistic, too ambitious, sufficiently defined.

Governance. High level visibility for managerial decision making.

- **Planning.** Schedule, resources, budget, number of iterations and deliveries
- **Stakeholders and Roles.** Developer, manager, tester, user
- **Control.** Tracking and oversight status

Preliminary remarks about the selection of terms are in order. The concepts of iteration and process are very broad and it is not our purpose to arrive at a definitive and comprehensive feature set. Rather we focus on software abstraction levels, which, according to our experience in industry, are crucial to adoption by practitioners and managers alike.

The myriad details of what goes into a process definition, e.g., phases, entry and exit criteria, review, can be overwhelming to all but the most die-hard process designers. Effective, attractive Presentation that abstracts away some of the detail but captures the spirit and motivation of the process are crucial to understanding and buy-in. The Activities of each Iteration are of paramount importance to practitioners as this tells them how the process will affect their "lives" on a daily basis.

The enthusiasm that practitioners exhibit toward the idea of iteration will be tempered by managers who want and deserve to know how short-term iteration is supposed to contribute to long-term Progress and what Governance mechanisms are in place to check whether this contribution is actually happening.

3 Iterative Process Models and their Precursors

Iteration is attractive in that it reflects how thought and work naturally evolve. Thus it is no surprise that iterative process models in many disciplines. Insights about iterative process models may be culled from a broad spectrum of models, presented in the ensuing sub-sections.

Section 3.1 addresses the grand-daddy of all software development models, the Winstron Royce's Waterfall Model [3]. Royce's paper was pioneering in its use of

diagrams to portray process problems and solutions. The paper does not propose iteration beyond "do it twice" but correctly analyzes how incorrect sequences of activities can sabotage progress. The proposed solution relies very heavily on Governance, including enforcement of formal documentation and reviews.

Section 3.2: If the Waterfall Model is the grand-daddy of all software development models, the Software Capability Maturity Model (SW-CMM[SM]) [5] is the grand-daddy of all software process improvement models. The Presentation aspect is highly developed, as is assessment-based Governance. The iterative component is defined and eloquently Presented but not explicitly integrated.

Section 3.3 discusses MIL-STD-498 [6-8] which is most noteworthy in its support for a variety of tailored process models, including iteration. However, the Presentation relies on labeled, linear drawings that evoke waterfall images. As a military standard, there is a heavy emphasis on detailed specification of activities and governance.

Section 3.4: Boehm's Spiral Model of Software Development [9], is pioneering in its attempt to integrate Iteration Activities and Progress in a single Presentation symbol. Because of the reliance on a single symbol, much of Boehm's wisdom is relegated to textual narrative.

Section 3.5: The Rational Unified Process [10] has evolved greatly since its genesis. However, its original Presentation symbol reflects an aspiration to unify iteration activities, progress and governance in a single perspective.

3.1 The Waterfall Model

The best known and most stigmatized process model is the waterfall model. The name most often associated with this model is Winston Royce [3], and hence his name is often stigmatized along with the model publicized in his pioneering paper. In a poignant defense of his father's honor, Walker Royce[4], aptly notes that his father's paper was quite critical of this model, focusing particular attention challenges posed by iteration.

Royce's (heretofore references Royce are to Winston) criticisms were indeed apt, but the linearity of the waterfall paradigm limited his thinking. According to Royce, iteration is viable only between "successive steps". Royce notes the problem engendered by detection of problems only in a late testing phase. The problem is substantive. But locked into a sequential model, expression of any kind of multi-stage iteration was visually intractable, and thus rejected. The only way he could integrate iteration in a sequential model was to insert another sequential step, "preliminary design" in effect implicitly iterating on design. Locked into a sequential model Royce had no choice but to "enhance" the process in order to get it right the first time, via abundant documentation. For risky projects, a pilot version would be developed prior to the production version. In Royce's mind this is a version and not iteration, since, in his mind set, anything traversing many steps is not iteration.

3.2 SW-CMM[SM]

The Software Engineering Institute's (SEI), Software Capability Maturity Model, is

now defunct but is instructive in its use of sequential and iterative elements. Its sequential model, encompassing the 5 maturity levels, is depicted as a staircase. There is also an iterative model that is supposed to capture how one progresses from one level to the next is a. Interestingly, in the book defining the SW-CMM[5], only one net page of text is devoted to the iterative model, the rest to the sequential model. In practice, progress in the sequential model was assessment driven, via a hierarchical model. The assessment strategy was to gather evidence along the hierarchy and reach conclusions bottom-up. Iteration was not integrated in the hierarchy, and under assessment pressure was likely to be neglected at best. In fairness there are implicit iterative aspects of CMM that do receive graphic expression. The change of part of speech in the level names from past participles (levels 1-4) to present participles (level 5, optimizing) suggests that at a certain level of success, freed from imminent assessment pressure, we have the luxury of iterating. Similarly, in the earlier representations of the sequential model, each level had an icon of the same form, some kind of metric such as cost, whose magnitude and variability dropped as the progressive model was ascended. This suggested that the same kind of achievement was attempted, over and over again.

3.3 MIL-STD-498

MIL-STD-498 [6] attempted to bring flexibility to development environments that adhered too rigidly to waterfall models. Its Guidebooks [7, 8] cover many cases where iterative development is appropriate. However a picture is worth a thousand words, the imagery is of replicated waterfalls, not of agile iterations

3.4 Boehm's Spiral Model of Software Development

Spiral models implicitly combine iteration and sequence, since a spiral concatenates successive circles. Nonetheless, as we saw in the previous case, spiral representations have a weakness with regard to the sequential component. Thus Boehm [9] apologetically explains that the radial distance in his model (figure 1), represents cost but is not to scale. He differentiates successive iterations according to their goals and which activities they do or do not omit. However, these are explained by anecdotal examples, and the combination of detailed text and multiple rounds of the spiral, is literally dizzying. This is not a criticism; Boehm's "it depends" characterization is intellectually honest. However the lack of a stereotyped, albeit overly generalized schema, makes it difficult to focus on the model representation.

3.5 Rational Unified Process

In contrast to Boehm, the Rational Unified Process, [10] discards the spiral metaphor, in favor of what is in effect a matrix, of development activities along one axis, and iterations, grouped according to phases along the other. This model succeeds in representing graphically, much information and correlation among the information types (figure 2). Since it has discarded the spiral, it can represent activities that do not

occur sequentially, in particular giving voice to Governance activities that are not represented in other models. Furthermore, rather than specifying discretely and only anecdotally whether specific activities occur in a particular iteration, their relative weighting over time is depicted graphically.

4 Analysis and Discussion

The taxonomy proposed in section 2 is multi-level abstractions: Presentation, Iteration Activities, Progress, and Governance. Not withstanding the truism that "devil is in the details," practitioners and managers who have neither time nor patience for frontal education, can obtain a vision of how these details fit together, and then be motivated to learn by doing. This vision can be imparted by the Presentation symbols. While Presentation is only one dimension, it can impart a vision of the other dimensions.

Effective presentation, however, requires effective presentation crafting tools. Up until the late 90s, the people who thought about processes could not directly create the Presentation symbols, but relied on the line drawings and text of graphic artists. This was the case for Waterfall, Boehm Spiral, and MIL-STD-498 of which had to rely primarily on supplementary text to articulate Progress and Governance issues.

The maturation of desktop publishing tools allowed process designers to directly articulate their ideas and stimulated thought about how to address disconnects among the various process model dimensions. (SW-CMM, RUP). Web-based presentation has further empowered process designers, a subject worthy of a separate study which we are undertaking.

As for the process adoption taxonomy features, we chose to illustrate them via a “sun” format (Figure 3) taken from coaching, in which every “beam” represents your current status of “success”. This presentation is very useful is defining your success vision in terms of where you would like to be according to the beams parameters values.

Based on the authors overall research and industry practice regarding software process implementation, the “sun” presentation of the process adoption is a valuable visibility into the process as adopted by the organization culture. The values on the “beams” of Progress, Governess, Iterative Activities, and Presentation, are always relative to the organization specific content (i.e. product engineering content, organization structure, roles title). However, looking into the sun shape of each process model can reflect on the organization vision and ability to improve on each of the process adoption taxonomy features that we presented.

5 Conclusions

Developing software and following a process is not easy. Iconic representations can help provide focus and some peace of mind. Although, as Glassman argues, there may be little fundamental difference among processes, "reinvention" of process representations has cathartic value for those grappling with difficult issues. Having said that there are significant representational differences among the different models,

and we propose a set of questions that should be asked when contemplating a particular model and evaluating its cultural and effectiveness with a given culture. We propose a set of criteria for comparing various iterative process models

- Graphical Sophistication and Effectiveness of the Presentation
- Level of detail and flexibility of Iteration Activities
- Level of detail and effectiveness of Progress measures
- Effectiveness of iteration-progress integration
- Effectiveness of Presentation Symbols in Facilitating Understanding and Adoption of Iteration Activities, Integration of Iteration Activities, Progress and Governance.

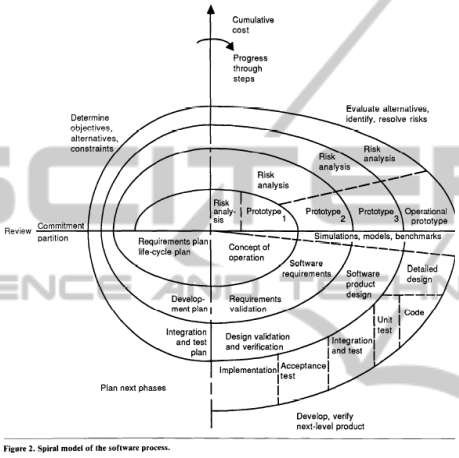

Figure 2. Spiral model of the software process.

Fig. 1. Boehm Spiral.

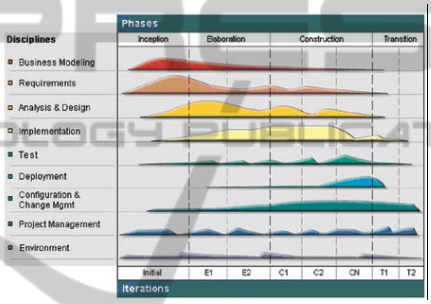

Fig. 2. Rational Unified Process.

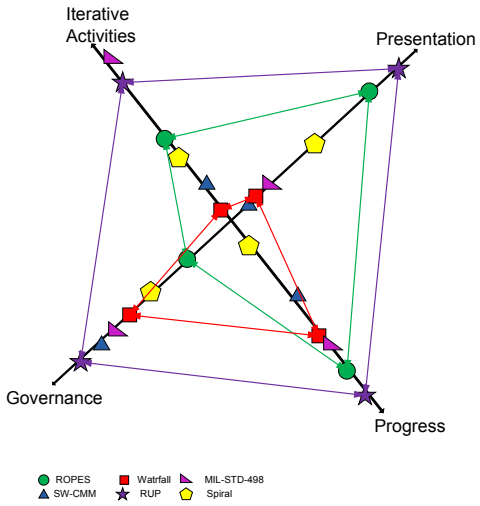

Fig. 3. Process Adoption Taxonomy Features.

References

1. Glass R.: Is there Anything "Time-Honored" In the Field of Software? Journal of Systems and Software, vol. 38, pp. 195-196 (1997).
2. Boehm B. and Turner R., Balancing Agility and Discipline, Addison-Wesley (2004)
3. Royce Winston, Managing the Development of Large Software Systems, IEEE WESCON Proceedings, pp. 328-338 (1970)
4. Royce Walker, Software Project Management: a Unified Framework, Addison Wesley, pp. 6-11 (1998)
5. Paulk Mark C. et al., The Capability Maturity Model: Guidelines for Improving the Software Process, Addison Wesley, p. 81-83 (1995)
6. MIL-STD-498, Software Development and Documentation (1994)
7. MIL-STD-498, Overview and Tailoring Guidebook (1996)
8. MIL-STD-498, Application and Reference Guidebook (1996).
9. Boehm B., A Spiral Model of Software Development and Enhancement, IEEE Computer, May (1988)
10. IBM Rational Unified Process Datasheet RAD10971-USEN-00, http://public.dhe.ibm.com/software/rational/web/datasheets/RUP_DS.pdf, (2007) last visited 20 Jan. 2011 (2008)