Toward a Comprehensive Framework for Software Process Modeling Evolution

Osama Eljabiri
New Jersey Institute of Technology
CIS Department
University Heights
Newark, NJ 07102
omae@home.com

Fadi P. Deek
New Jersey Institute of Technology
CIS Department
University Heights
Newark, NJ 07102
deek@njit.edu

Abstract

Software process modeling has undergone extensive changes in the last three decades, impacting process' structure, degree of control, degree of visualization, degree of automation and integration. These changes can be attributed to several factors. This paper studies two of these factors, the time dimension and the interdisciplinary impact, and assesses their effect on the evolution of process modeling. A literature survey for software process modeling was carried out which provided evidence of how the time dimension and the interdisciplinary impact triggered process evolution and changes in methodology, technology, experience and business requirements. Finally, the paper concludes with a theoretical framework to serve as an illustrative model for the effects of the time dimension and interdisciplinary impact on process modeling evolution. This framework can serve as to develop more advanced models for technological forecasting in software process modeling evolution.

Keywords

Software engineering, software process modeling, software process evolution, interdisciplinary impact, change in time, software development, software project management.

1. Introduction

Examining the software development life cycle literature reveals a wealth of approaches that have been introduced in the last three decades. Many vary by titles, rationales, structures, degree of mapping and visualizing the real world applications and the extent of how these models reflect strategic goals in organizations. Moreover, there are a variety of approaches to classifying these models and to providing criteria for applying them to diverse business requirements. This can be attributed to several factors including evolving experiences of software engineers, degree of problem complexity, organizational goals, availability of technology, human factors and cognitive styles in addressing problems. Although several studies have examined the software development process literature at different levels of detail and abstraction [1, 2, 3, 4, 5, 6, 7, 8], there is still a benefit to a comprehensive review of the current software process literature, with a focus on the evolution of software process models as a function of time. As already indicated, there were several factors contributing to the diversity of software process models. One of these factors is the interdisciplinary impact influencing the development of software process models. The combined effect of the time dimension and the interdisciplinary impact might not only explain the evolution of process models but also may be useful in perhaps foreseeing future developments of process modeling.

2. Literature Review

The evolution of process models started by the code and fix model [2], which fits the solution into the problem rather than drawing solutions from well-defined problems. Pressman [9] presented a comprehensive survey, though some approaches were not considered, and introduced the following process models: linear sequential (classical waterfall), prototyping model, RAD model, incremental model, spiral model, component assembly model, concurrent development model.

Somerville [3] placed the process models he addressed in four main categories: the waterfall approach, the evolutionary development, the formal transformation, and assembly from reusable components. Evolutionary development is based on stages that consist of increments where "the directions of evolution are determined by operational experience" [2].

Behforooz and Hudson introduced another useful classification [4]. They considered all process models virtually as versions of the waterfall model and introduced models that were overlooked by others such as the Department Of Defense (DOD) system development life cycle and the NASA model.

The waterfall model has played a significant role in process modeling evolution over the decades and has become the basis for most software acquisition standards [2].

Although the waterfall model has its drawbacks, it is still the super class of many process-modeling approaches in software engineering. The unified software development approach proposed by Jacobson et al. [5] addressed some of the problems with previous models using an object-oriented approach and UML standards. This model is use-case driven, architecture centric, iterative and incremental, and has new phases: Inception, elaboration, construction and transition. While the unified process model can be characterized in terms of its object-oriented methodology and iterative nature, a framework by Abdel-Hamid et al. [10] was introduced to address management considerations coupled with software economics aspects. This framework recognized the impact of the control of resources variable on the overall performance of process models and thus gained popularity. IBM Cleanroom is another team oriented approach to software engineering in which intellectual control of the work is ensured by ongoing review by a qualified small team and the use of the formal methods in all the process phases and statistical quality control of an incremental development process [11].

Process models with built in object-oriented techniques can be easily modified, extended and viewed at appropriate levels of abstraction. Their application areas include "development of an abstract theory of software process, formal methods of software process, definition and analysis, simulation of software process and the development of automated enactment support" [12].

Although object-oriented methodologies have proven to be advantageous in process modeling, SOFL (structured-object-oriented-formal language) [13] is an approach that shows how integration between structured and object-oriented methodologies can add more value to a process model. This approach also combines static and dynamic modeling. These integrations aimed to develop a process model that overcomes formal methods problems, which limited their use in the industry.

Introducing risk-driven process models was a significant breakthrough in process modeling after a large library of models based on document-driven or code-driven approaches as "the evolving risk driven approach provided a new framework for guiding the software process" [2]. This was referred to as the spiral model, which was to be adaptable to the full range of software project situations and flexible to accommodate a high dynamic range of technical alternatives and user objectives. However, the spiral model required further calibration to be fully usable in all situations [2]. In an effort to resolve model conflicts, Boehm [14] expanded the spiral model to another version named "win-win spiral model". In this version of spiral model Boehm used a stakeholder win-win approach to determine the objectives, constraints and alternatives for each cycle of the spiral.

The prototyping model can be used as a generic tool in the software development process. Not only it can be integrated with other process models, but also it can assist in developing the requirements analysis phase. Furthermore, it can be used as an experimental tool in assessing the efficiency of the entire development process. In this respect, prototyping can be utilized as a mechanism in monitoring software processes before investing a great deal of efforts and resources [15]. The spiral model can also be utilized as a process model generator [16]. Boehm et al. used the spiral model as a framework for eliciting or generating adequate process models by means of the decision table technique. Another example for combined effect of both interdisciplinary impact and change in methodology is the commercial off-the-shelf (COTS) approach, which gained more attention recently. COTS components can be a complete application, an application generator, a problem-oriented language, or a framework in which specific applications are addressed by parameter choices [17].

Web development life cycle, recently referred to as web engineering, is also gaining an increasing interest in software development [18]. In order to develop and maintain web sites in a cost-efficient way throughout their entire life cycle, sophisticated methods and tools have to be deployed [18].

The reengineering process model is an approach based on business metrics of cost, time, and risk reduction as a result of substantial change in existing processes, which would create breakthroughs in the final product. According to Somerville [3], software reengineering has three main phases: defining the existing system, understanding and transformation, and reengineering the system. While traditional models were supported by loosely coupled CASE tools that provide assistance independently each phase of the life cycle, more sophisticated architectures were lately introduced. These provide mechanisms to ensure proper tool integration and interface capable of monitoring and coordinate the activities and actions of software projects and team members [19]. The TAME process modeling approach represents a step toward integrating process modeling with product metrics along with the automation capabilities of CASE tools in a more comprehensive framework [1]. Integrating experimental data with CASE tools can also make the process model much more efficient by allowing data collection and knowledge base building throughout the development process. This approach has been introduced through the CAESE methodology where the tools and the experiment design combine to meet the software production goals as assessing software product metrics will be more efficient with the statistical analysis based on experiments accompanied by the high degree of CASE automation [20]. The flow of events represented by the cleanroom development increment life cycle based on formal techniques can be categorized in the same class [21].

Finally, the cognitive prospective and human factors in developing process models are also relevant since problem solving cannot be achieved efficiently without adopting adequate strategies that are based on correct understanding of humans and their real needs [22]. Behavioral approaches have enhanced software usability from a user-oriented prospective particularly in the area of user interface design, thus influencing process modeling as well [23].

3. Analysis

The software development life cycle offers a methodology that developers follow to achieve software solutions for real world problems. This methodology reflects the evolving of the software solutions through a timeframe of a development process. It also represents the technical, human and financial resources required to perform the software project activities. In other words, it is a problem-solving framework that works within limited time and limited resources.

As identified by Jaccheri et al., software processes are complex activities that affect critical parameters such as final product quality and costs [24]. Therefore, process control is significant to assure software product quality, as the duality of product and the process is an important element in software engineering [9]. The control capability is not only utilized for the purpose of preventive maintenance and corrective actions but also for quality assurance, quality improvement and forecasting. both in their structure and outcomes.

Methodology adopted is an important factor that should be considered. Object-oriented methodology has a different impact than the process-oriented methodology on software development life cycle modeling.

The degree of complexity in business problems is also an influential factor that should be considered. The change in the nature of business problems added more complexity to business processes, which resulted in changes in business requirements as a function of time.

The time dimension variable and its associated factors impact the degree of visualization across process models. Degree of visualization is also a measure of process modeling evolution.

4. Conclusion

In conclusion, the paper suggests a final conceptual framework. This framework indicates the effect of the time dimension and the interdesciplinary impact on the evolution of process modeling. It also indicates the effect of the time dimension on the interdesciplinary impact. This cross-relationship can be attributed to the change in experience and business requirements that triger the involvement of more deciplines in the assessment and development of more effecient process models.

Based on this framework, several implications could be extracted.

The first implication is the significant role of the four intervenning variables (i.e: change in experience, technology, methodology and business requirements) in transferring the effect of the time dimension on the evolution of process modeling. Another implication is that the dgree of automation, degree of visualization, degree of control, degree of integration and changes in structure could be used as measures of the extent in which process models evolve. The third implication is that the interdesciplinary impacts have had critical effects on process model evolution. This effect was coupled with the time dimension variable and trigered by its four intervenning variables. Cognitive phsycology played an important role in the context of behavioral and protpotyping models as more user involvement implies more human considerations. This can also be understood in the context of the customer economy as user satisfaction becames an issue in evaluating information systems. Software economics is another significant issue in this framework as it trigers the attention to risk considerations. Software economics encompasses several metrics of business performance that can also be addressesd in future studies. These business metrics that should be reflected in process modeling include cost reduction, profit maximazation, market share, competitive advantage, and the effect of project diversification in large organizations.

Other interdecilpinary components addressed in this paper include management and industrial engineering which are correlated. Management had a significant effect on process modeling structure as it allowed the incorporation of system dynamics subsequent to static modeling. Industrial engineering drew the attention to quality assurance standards applied to business processes which motivated software engineers to develop standards such as ISO9001 and the CMM model. The paper also discussed the impact of mathematics on the evolving of formal mothods and specification languages. In sum, this paper presents a suggested framework for a better understanding of the evolution of process modeling in terms of the time dimension and interdisciplinary impact. This framework can be used as an explanatory model of process modeling history and evolution, as well as for predictive purposes and technological forecasting.

References

[1] Victor R. Basili and H. Dieter Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments", IEEE Transactions on Software Engineering, vol. SE-14, no. 6, June 1988, pp. 752-772.

[2] Barry Boehm, "A Spiral Model of Software Development and Enhancement", IEEE Computer, vol. 21, no. 5, May 1988, pp. 61-72.

[3] Ian Somerville, Software Engineering, New York, NY, Addison-Wesley, ISBN 0-201-17568-1, 1995.

[4] Ali Behforooz,, Software Engineering Fundamentals, ISBN 0-19-510539-7, Oxford university press, New York, 1996.

[5] Ivar Jacobson, Grady Booch and James Rambaugh, "The Unified Software Development Process", ISBN: 0-201-57169-2, Addison Wesley, New York ,1998.

[6] Barry Boehm ,"Anchoring the Software Process", IEEE Software, July 1996 .

[7] Shari Lawrence Pfleeger, Software Engineering: Theory and Practice, Upper saddle River, NJ: Prentice Hall Corp, 1998.

[8] 1074-1995: IEEE Guide for Developing Software Life Cycle Processes.

[9] Roger Pressman, Software Engineering: A Practitioner's Approach, 4th Edition, New York, NY McGraw-Hill, ISBN 0070521824- 1438, 1996.

[10] Tarek. Abdel-Hamid and Stuart E. Madnick, "Lessons Learned From Modeling The Dynamics Of Software Development ", Communications of the ACM vol. 32, no. 12 Dec. 1989, pp. 14-26.

[11] Carmen J., Trammell, Leon H. Binder and Catherine E. Snyder, "The Automated Production Control Documentation System: A Case Study In Cleanroom Software Engineering", ACM Transactions on Software Engineering Methodology vol. 1, no. 1 , Jan. 1992, pp. 81 – 94.

[12] John D. Riley, "An Object-Oriented Approach To Software Process Modeling And Definition", Proceedings of the 1994 conference on TRI-Ada '94, 1994, pp. 16 – 22.

[13] Shaoying Liu, Offutt, A.J., Ho-Stuart, C., Sun, Y., Ohba, M., "SOFL: A Formal Engineering Methodology For Industrial Applications", IEEE Transactions on Software Engineering, vol. 24, no. 1, Jan. 1998, pp. 24 –45.

[14] Boehm, B. & Port, D., "Escaping The Software Tar Pit: Model Clashes And How To Avoid Them", Software Engineering Notes. 24(1), January 1999 ,pp. 36-48.

[15] Bradac, M., D. Perry, and L. Votta, "Prototyping a Process Monitoring Experiment", IEEE Transactions on Software Engineering, vol. 20, no .10, October 1994, pp. 774-784.

[16] Barry Boehm and Frank Belz, "Experiences With The Spiral Model As A Process Model Generator", Proceedings of the 5th international software process workshop on Experience with software process models, 1990, pp. 43 – 45.

[17] W. Morven Gentleman, "Effective use of COTS (commercial-off-the-shelf) software components in long-lived systems" (tutorial), ACM Proceedings of the 1997 international conference on Software engineering, 1997, pp. 635 – 636.

[18] Jung Reinhard and Robert Winter, "Case For WEB SITES Towards An Integration Of Traditional Case Concepts And Novel Development Tools", Institute for Information Management University of St. Gallen, http:\\iwi1.unsg.ch\research\webcase, 1998.

[19] Jayashree Ramanathan and Soumitra Sarkar, "Providing Customized Assistance for Software Lifecycle Approaches", IEEE Transactions on Software Engineering, vol. ~14, no. ~6, June 1988, pp. 749-757.

[20] Torli, K., Matsumoto, K., Nakakoji, K., Takada, Y., Takada, S., Shims, K., "Ginger2: An Environment For Computer-Aided Empirical Software Engineering ", IEEE Transactions on Software Engineering, vol. 25, no. 4, July/August 1999, pp. 474 –491.

[21] Carmen J. Trammell, Leon H. Binder and Catharine E. Snyder, "The Automated Production Control Documentation System: A Case Study In Cleanroom Software Engineering", ACM Transactions Software Engineering Methodology, vol. 1, no. 1 , Jan. 1992, pp. 81 – 94.

[22] Leveson N.G., "Intent Specifications: An Approach To Building Human-Centered Specifications", IEEE Transactions on Software Engineering, vol. 26, no. 1, Jan. 2000, pp. 15 –35.

[23] J. D. Chase, Robert S. Schulman, H. Rex Hartson and Deborah Hix, "Development And Evaluation Of A Taxonomical Model Of Behavioral Representation Techniques"; ACM, Conference proceedings on Human factors in computing systems: "celebrating interdependence", 1994, pp. 159 – 165.

[24] Maria Letizia Jaccheri, Gian Pietro Picco and Patricia Lago, "Eliciting Software Process Models With The E3 Language ", ACM Transactions on Software Engineering Methodology, vol. 7, no. 4, Oct. 1998, pp. 368 – 410.