December 2001

Drivers for Software Process Modeling Evolution

Osama Eljabiri
New Jersey Institute of Technology

Fadi Deek
New Jersey Institute of Technology

DRIVERS FOR SOFTWARE PROCESS MODELING EVOLUTION

Osama Eljabiri
CIS Department
New Jersey Institute of Technology
omae@home.com

Fadi P. Deek
CIS Department
New Jersey Institute of Technology
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 types of these factors and their growth over the time dimension, and assesses their effect on the evolution of process modeling. An initial representation to address software process evolution was presented, then a literature survey for software process modeling was carried out which provided evidence of how the time dimension has affected the growth of problem and solution related factors that triggered process evolution. Finally, the paper concludes with a theoretical framework to serve as an illustrative model for the effects of the time dimension, problem-related factors and solution related factors 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 impacts, software development, software project management, systems analysis and design

Introduction

Changes in demands of the software business and the software-driven business have dramatically impacted the way software is developed. While the primary goal of software engineering was to deliver software products on time within budget, and to meet customer expectations, software is developed today to thrill customers and to exceed their expectations. Achieving desired software quality used to be a tradeoff between speed of development, performance, reliability and cost. Today, software quality has to be aligned with rapid development and competitive costs. Fortunately, today's software engineer is more capable than ever before. Thanks to the accumulative experiences and empirical studies that lead to profound determination of best practices. This occurs in conjunction with the exponential rate of growth in technology, acquiring more processing power and more bandwidth. Furthermore, software engineering is today more exposed to plenty of interrelated disciplines providing more tools to analyze business problems and more instruments to quantify, specify, visualize and optimize the software development process.

Examining the software development literature reveals a wealth of approaches that have been introduced in the last three decades. Many vary by rationales, structures, automation, degree of control, degree of visualizing the real world situations and the extent in how these models reflect strategic goals in organizations This can be attributed to several factors including evolving experiences, problem nature, degree of problem complexity, organizational goals, availability of technology and human-related factors.

Although several studies have examined the software development process literature at different levels of detail and abstraction (Basili and Rombach, 1988; Humphrey and Kellner, 1989; Bandinelli et al., 1995; Sutton, 1988; Curtis et al., 1992;Boehm, 1988; Jacobson et al., 1998; Boehm, 1996), 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. However, most of these factors are related to the growth of goals and capabilities in the software industry over the timeline. Growth in goals drives problem-related factors that reflect stakeholders' requirements. Growth in capabilities triggers solution-related factors which projects emerging cutting-edge technologies, evolving methodologies and increasing interdisciplinary impacts. The combined effect of the time dimension, problem related factors and solution-related factors might not only explain the evolution of process models but also can help in foreseeing future developments of process modeling. *Figure* 1. shows an initial representation of this conceptual framework in terms of the time dimension as the initial independent variable that is anticipated to continuously trigger problem and solution related factors to generate more

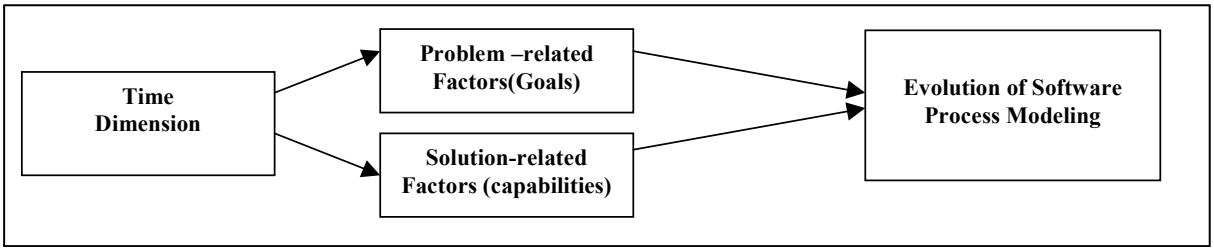

Figure 1. An Initial Representation of the Three-Dimensional Effect on Process Modeling Evolution

evolution in software process modeling. This paper provides a comprehensive review of software process modeling literature to assess this framework and further examines the effects articulated in this initial representation.

Method

This study assumes that goals (problem-related) factors and capabilities (solution-related) factors are the most influential factors in driving software process modeling evolution as a function of the time dimension. To examine and expand this initial framework shown in Fig 1, a literature survey is carried out. This survey cites ACM, IEEE, and other software engineering based sources that focus on software process modeling and project management to give supportive evidence and provide accurate measures of the independent and dependent variables addressed in the initial framework. Sources are grouped into general historical, problem-related and solution-related categories respectively to obtain relevant conceptual data. Conceptual data is then analyzed in terms of establishing a theoretical foundation of the considered variables and their measures. Interrelations and impacts among these variables are further explored. Consequently, a final expanded framework is concluded to reflect the study findings.

Literature Review

Evolution of Software Process Models

The software industry moves ambitiously to explore new techniques and methodologies for managing the ever-increasing complexity of software projects. In the past, business requirements were simpler in nature and stakeholders were not considerably involved in the development process. A large application was a few thousands lines of code. Additionally, due to mass production strategies in business, customers were of a much lesser impact on software developers. Software development was solution-centered where business problems are not carefully defined and the code and fix model (Boehm, 1988) was the practical trend in software industry. Unfortunately, this approach was not capable to handle dramatic changes in the business world, which resulted in several software crises in the 60's including visible and invisible backlogs, uncontrolled costs, unpredicted errors, and project run a ways. These problems triggered software developers to rethink their applications in a way that places the sources code as the final result of a well-defined process that starts with capturing problem dimensions and user requirements before any technical issues are involved. This process aims to enhance customer-developer communications, manage and control software projects effectively, and provide a foundation for building tools that will support software production (Liu et al., 1989).

Consequently, a large array of software development models evolved in the last three decades. *Table* 1 describes the chronological evolution of software process models over the last three decades based on the cited literature in this paper. Although this table does not include every model addressed in the literature, it represents the major streams in software process modeling since 1970.

Problem Related Literature

Software development is the art of solving problems by means of computer information systems. Thus, the evolution of business problems, degree of problem complexity and the extent in which customers are involved in problem definition drive the efforts in finding appropriate solutions so organizational goals can be sufficiently achieved. The waterfall model, proposed by Royce (1970), has played a significant role in process modeling evolution over the decades, as it has become the basis for most software acquisition standards (Boehm, 1988). The waterfall was another improved version of the earliest process model named nine-phase stage-wise model. As a response for change in project complexity and business demands for risk management, the V-shaped model was developed as an extension of the waterfall with the incorporation of validation and verification procedures (Madhavji, 1991).

Table 1. Chronological Evolution of Software Process Models Over the Last Three Decades

Process Model	Year of Development	Developer
Waterfall model	1970	Royce
Iterative enhancement model	1975	Basili and Turner
Transformational model	1981	Balzer
Evolutionary development model	1982	
Operational Specification Model	1984	Zave
Cleanroom model	1987	IBM
Spiral model	1988	Boehm
The TAME (Improvement-oriented) Model	1988	Basili and Rombach
Dynamic (Project Management-Oriented) Model	1989, 1991	Abdel-Hamid and Madnick
Rapid Application Development (RAD)	1991	Martin
V-shaped Model	1992	German Ministry of Defense
Capability Maturity Model (CMM)	1993	Curtis et al.
Concurrent Development Model	1994	Davis and Sitaram
Unified Software Development Model	1998	Jacobson et al.
Commercial-Of-The-Shelf (COTS)	1998	

Prototyping was the second most influential technique in software process modeling as it was adopted –implicitly or explicitly- in almost every process model after the waterfall. Although there is no unique definition for software or information systems prototype (Alavi, 1984; Lichter et al, 1989), it is based on viewing software development as an evolutionary process (Lichter et al, 1989) as well as recognizing the increasing importance of software economics and customer requirements.

The famous spiral model, proposed by Boehm (1988), also exhibits a heavy reliance on prototyping (Yamamichi et al, 1996) and software engineering economics (Boehm, 1984), as it is mainly a risk-driven process model (Boehm, 1988). Boehm integrated previous process models (waterfall, evolutionary, incremental, transform) into his spiral model based on project-customized needs in an effort to maximize benefits and reduce uncertainty. In an effort to resolve model clashes and conflicts, Boehm (Boehm and Port 1999) expanded spiral model to another version named "win-win spiral model" which is more customer-driven and user-centred.

With prototyping and spiral models software development moved from solving problems sequentially to tackling them iteratively. The iterative approach is a strong reflection of incorporating the business value and customers impact in software development. Furthermore, the iterative approach is the strategic framework for the unified process model suggested by the pioneers of the object-oriented unified modeling languages (UML) at Rational Rose. The unified software development approach, proposed by Jacobson and others (Jacobson et al, 1998) is an architecture-centric approach that seeks a strategic balance among the different goals of software process models.

As a response to management-related problems in software industry, Abdel-Hamid and Madnick (Abdel-Hamid and Madnick, 1989) introduced a dynamic model (1988-1991) to address management considerations coupled with software economics aspects.

Solution Related Literature

Inspired by the data mining capabilities and the advancements in mathematical approaches to requirements specifications, in 1987 IBM proposed another process modeling framework named the Cleanroom process model. Cleanroom is a team-driven approach to software engineering in which intellectual control of the work is ensured through continuous reviewing by a qualified small team and the use of the formal methods in all the process phases in conjunction with statistical quality control of an incremental development process (Trammell et al., 1992).

As a response to the cutting edge CASE tool technologies, the Commercial of the shelf (COTS) approach in process modeling was proposed in 1998 and has gained more attention over the time. 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 (Gentleman, 1997).

Moreover, Internet capabilities have had a significant impact on software process modeling in the last few years. Web development methodologies, recently referred to as web engineering, gained an increasing interest in software development and were reflected in new web-driven process models (Reinhard and Winter, 1998).

Furthermore, the business process-reengineering (BPR) trend in synchronizing business processes and software processes is being reflected in the reengineering process models. The TAME process modeling approach also represents another step toward integrating process modeling with product metrics in conjunction with automation capabilities of CASE tools in a unified framework (Basili and Rombach, 1988).

Integrating good practices has influenced software process models towards continuous improvement in an evolutionary cycle. This can be seen with models that focus on quality assurance in the software process such as the Capability Maturity Model (CMM) (1993), the Bootstrap Model, the Spice Model and other process improvement models (Somerville et al., 1999). More recently, attempts were also directed to integrate the CMM model with ISO9000 standards for quality assurance in software development.

Finally, the cognitive prospective and human factors in developing process models are also reflected in process modeling literature since problem solving cannot be achieved efficiently without adopting adequate strategies that are based on understanding of humans and their real needs (Leveson, 2000). 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 (Chase et al., 1994).

Analysis

The software development process offers a methodology that developers follow to achieve effective software solutions for real world problems. This methodology views the development process as a problem-solving framework that works in the context of limited time and resources as shown in *Figure* 2. This diagram is an abstract –super class - model extracted from the previous literature survey of software process models .It serves as a generic architecture of most software process models addressed. This super class model recognizes problem definition as the foundation of any process modeling approach that attempts to provide effective solutions. It also realizes the role of capabilities in terms of available resources. Problem definition reflects and impacts organizational goals as well as capabilities reflect and impacts human, technical and financial resources needed for efficient solutions. The time dimension brings more demands and changes that require continuous adaptability with added-complexity. However, time also drives more capabilities that assist in providing better problem analysis and better resources.

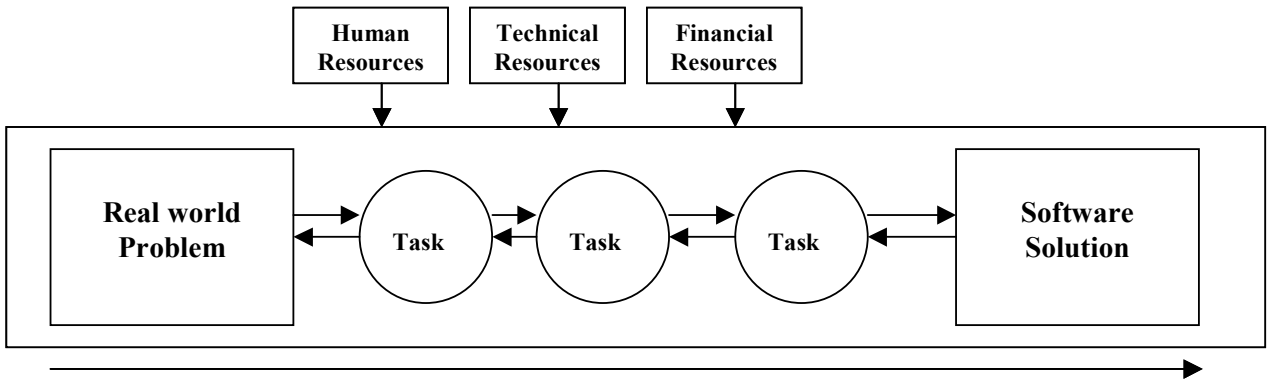

Figure 2. The Software Development Process as a Framework of Problem Solving in the Context of Time and Resources

The literature survey showed how time triggers changes in software process models in terms of the five aspects shown in Table 2.

Table 2. Measures of Evolution in Software Process Modeling

Aspect	Illustrations from Literature
Structure	The shift from sequential architectures in waterfall model to iterative architectures in Spiral and prototyping models
Degree of Control	Formal specifications – as in IBM Cleanroom- enables more validation and verification for user requirements as opposed to the early traditional methods
Integration	Rational unified process and win-win spiral models offer comprehensive frameworks in which several process models were utilized
Automation	COTS, Cleanroom, the TAME, and dynamic models utilize the evolutions of CASE tools and simulation technologies to automate the development process
Visualization	The move from structured modeling paradigms to object-oriented modeling and use-case driven paradigms enables more visualization especially with the availability of advanced modeling packages and simulation software

The main purpose of this paper was to explain the variation of the dependent variable of software process modeling evolution by means of problem-related (goals) and solution -related (capabilities) factors that evolve over the time. Table 2 explains how process-modeling evolution was a result of dramatic changes in process structure, control, integration, automation and visualization. These aspects are considered as the conceptual measures of the dependent variable.

On the one hand, the literature demonstrated a wealth of examples on how these measures indicate significant changes by the presence of more capabilities. These capabilities encompass technology, experience, methodology and interdisciplinary impacts as the main components of the solution-related factor. As shown in Table 2, technological capabilities seems to be the most influential component on process modeling in terms of their impact on process automation, visualization and degree of control. Furthermore, with the absence of fourth generation techniques and languages, the early process models were manual and more sequential in structure. This can be attributed to the usage of process technology as an enabler to support rapid application development that is needed for iterative approaches with more focus on risk minimization and user satisfaction. Time is mathematically related to advancements in process technology according to Moor and Metcalf's laws.

Time also improves experiences in process modeling development. Moving from the traditional waterfall to the V-shaped model and shifting from the convention spiral to the win-win spiral model across decades are examples of the effect of change in experience on process modeling structure and definition. This capability measure is also a function of problem-related factors where increase in problem complexity and business requirements significantly influence experiences and alter the way problems were originally tackled.

Moreover, the literature showed that the type of methodology adopted had considerable effects on process modeling evolution. For instance, object-oriented methodology offered tremendous support to the architecture-centric approach in rational unified process models in terms of structure, automation and visualization as opposed to process-oriented methodologies. Although these two methodologies have some conceptual similarities in the earlier phases of process model, they became more detailed as implementation-related factors are considered or techniques and representational constructs are utilized (Agarwal et al., 1999). However, the SOFL (liu et al., 1998) model presents an integrated approach that adopts structured oriented methodologies in the requirements phases and object oriented methodologies in the design and implemenation phases. The methodology adopted can be also quality assurance driven. This is associated with evaluation of software systems. Gradual improvement approaches such as TQM view problems from a different prospective as opposed to dramatic changing approaches such as BPR. Considering gradual improvement, SEI - CMM approach, SEI-CMM approach, Kaizen approach, QIP, and BUTD approach have been introduced (Bandinelli et al., 1995) with significant impact on structuring and automating the development process.

The software industry started with heavy reliance on technical sciences with very little or no attention to human disciplines. It was discovered thereafter that this was one of the biggest mistakes in software business strategies that contributed to the increasing number of project failures. Human resources were recognized by the incorporation of staff management in dynamic modeling as well as the cognitive-related variables in behavioral models. These incorporations are attributed to the interdisciplinary impact of management and cognitive psychology on strategic problem solving in software industry. Moreover, economical considerations were more identifiable in process modeling history by the incorporation of risk management components in the prototyping, spiral and other iterative models.

This is attributed to the contribution of software economics discipline to the software development process as it addresses crucial issues such as the assessment of project feasibility, cost estimation, risk assessment, productivity, planning and control. Integrating economics with process modeling provides an evaluation framework that takes into consideration both the technical and the

human aspects of a situation where obtaining the best possible information processing service is restricted to limited resources (Boehm, 1984). Industrial engineering, math, and artificial intelligence are other important examples of interdisciplinary impacts on process modeling evolution. Industrial engineering drew the attention to quality assurance standards applied to business processes which motivated software engineers to develop standard models such as ISO9001 and CMM.

On the other hand, the literature provides many evidences on how process-modeling evolution reflected the strong impact of problem- related factors, which ultimately changed business goals and strategies as well. The degree of complexity in business problems is one essential measure in this regard. 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. Moreover, the higher the level of management is in an organization, the more its business problems become unstructured. Also, the larger the project is the more likely risk will be increasingly involved. Working with small systems is a different experience than working with large ones, as modularization will not generate reliability without tailored approaches and techniques (DeRemer and Kron, 1976). It has also become apparent that customers increasing impact on business has made a tremendous impact on software modeling in terms of structure and visualization. Iterative structures increase user involvement significantly. Also, with more visualization, customer-developer communication becomes more effective. Therefore, incorporating customer considerations had significantly impacted the evolution of process modeling.

Conclusion

In conclusion, the paper suggests a final schematic representation of drivers for software process modeling evolution, as shown in *Figure* 3. This representation serves as a more representative theoretical framework in terms of relationships and measures as opposed to the initial framework presented in the introductory section. Based on this framework, the literature survey and the analysis sections, several implications can be made:

1. It becomes clear that time is a critical predictor for advancement in software process modeling. Most of the influential drivers in explaining evolution in process modeling were functions of time. However, time alone is not able to explain all the variation in the dependent variable unless combined with other independent variables. Time can be viewed as necessary requirement for problem and solution related drivers. Therefore, it is considered a trigger as well as a constraint.

2. Though problem-related factors were essential to trigger substantial changes in the evolution of software process models, the availability of resources and capabilities (solution-related drivers) had more impact on this evolution. This can be attributed to the crucial role of capabilities in impacting problem –related factors in the first place. This implies that problem and solution related factors are not mutually exclusive as one is dependent on the other.

3. According to the literature, the degree of automation, degree of visualization, degree of control, degree of integration and the extent in which changes in process structure take place can be used as measures of the evolution of software process modeling. However, this cannot guarantee the content validity of these measures, as further studies should be carried out to replicate these findings or add more to measures to them provide. These measures are also helpful in indicating a better framework of the major goals of software process modeling as opposed to the ones addressed in the traditional literature.

Interdisciplinary impacts have had critical effects on process modeling evolution. These effects were coupled with the time dimension variable. Cognitive psychology plays an important role in behavioral and iterative models as a reflection of the increasing user involvement in today's customer economy. Software economics is another significant component in this framework as it triggers the attention to risk considerations and the business value of software systems. Further studies can expand the human dimension to provide a more comprehensive framework. Other interdisciplinary components addressed in this paper include management and industrial engineering, which are correlated.

Although theoretical foundations were provided in support of the analysis and conclusions in this paper, statistical studies can add a significant value to the inferences drawn for generalization purposes . More variables could be also explored and integrated and variable measures might be further assessed and modified. This framework can be used as an explanatory model of process modeling history and evolution, as well as for predictive purposes in terms of technological forecasting.

Acknowledgments

This work is supported, in part, by New Jersey Information-Technology Opportunities for the Workforce, Education and Research (NJ I-TOWER), a New Jersey Institute of Technology (NJIT) project funded by the New Jersey Commission on Higher Education "High Technology Workforce Excellence Grant" initiative, award # 01-801020-02.

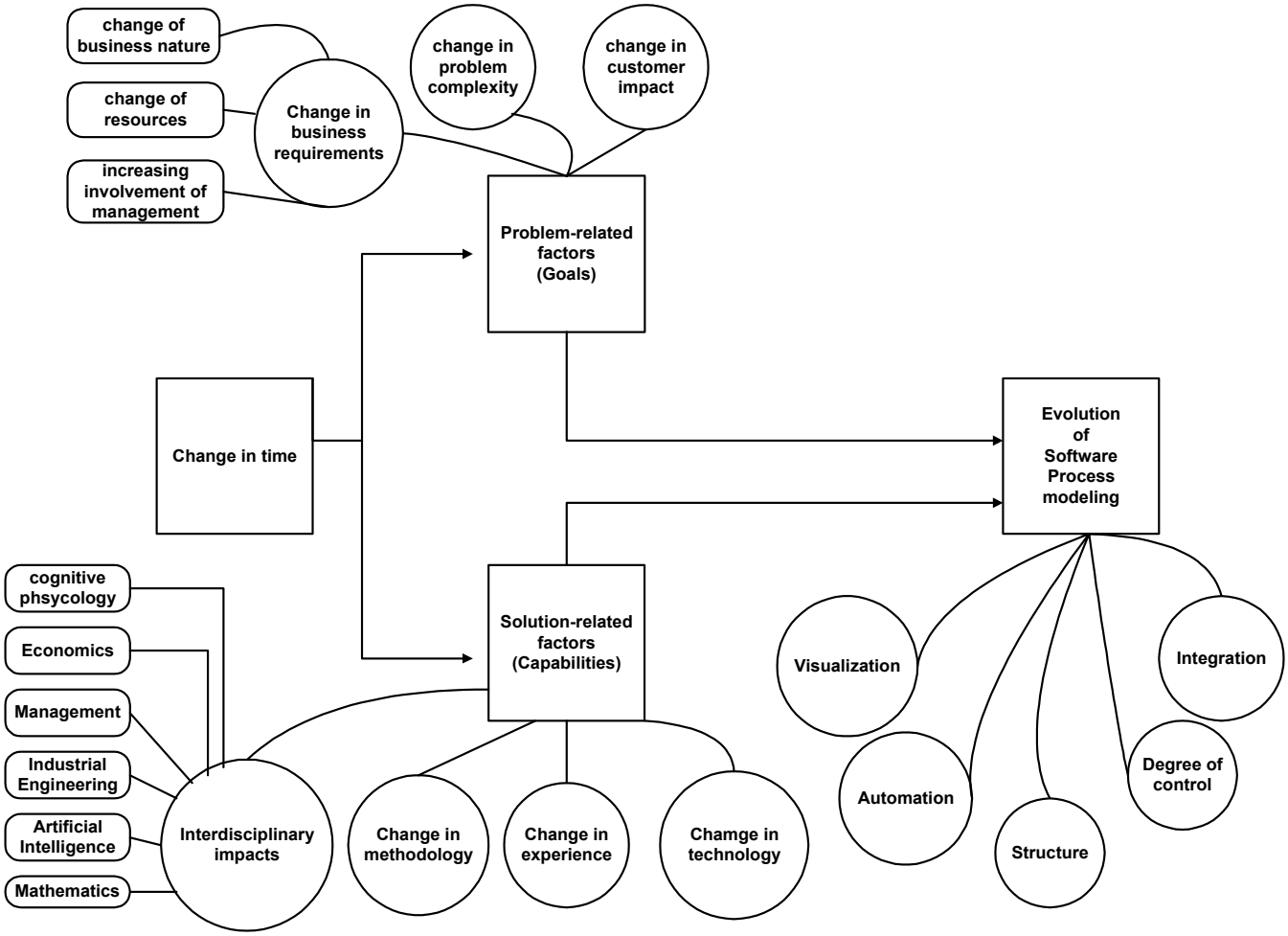

Figure 3. The Final Theoretical Framework of the Drivers for Software Process Modeling Evolution

References

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

Agarwal, R., De, P., and Sinha, A.P. "Comprehending Object And Process Models: An Empirical Study,” IEEE Transactions on Software Engineering, vol. 25, no. 4, July/August 1999, pp. 541 –556.

Alavi, M. "An Assessment of the Prototyping Approach to Information Systems Development," CACM, 27(6), June 1984,pp. 556-563. Basili, Victor R., and Rombach , H. Dieter. "The TAME Project: Towards Improvement-Oriented Software Environments," IEEE Transactions on Software Engineering, vol. SE-14, no. 6, June 1988, PP 752-772.

Bandinelli, S., Fuggetta, A. Lavazza, L., Loi, M. and Picco, G.P. “Modeling And Improving An Industrial Software Process ,” IEEE Transactions on Software Engineering, , vol. 21, no. 5, May 1995, pp. 440 -454.

Boehm, B. “Software Engineering Economics,” IEEE Transactions on Software Engineering,, vol. 10, no. 1, Jan. 1984, pp. 4-21.

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

Boehm, B., & Port, D. “Escaping The Software Tar Pit: Model Clashes And How To Avoid Them,” Software Engineering Notes. 24(1), Jan. 1999,pp. 36-48.

Chase, J. D., Schulman, Robert S., Hartson, H. Rex and Hix, Deborah. ”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.

Curtis, Bill ., Kellner, Marc I., and Over ,Jim. “Process Modeling,” Communications of the ACM vol. 32, no. 9 , Sep. 1992, pp. 75 – 90.

DeRemer, Frank. and Kron, Hans H. "Programming-In-The-Large Versus Programming-In-The-Small ," IEEE Transactions on Software Engineering , vol. ~SE-2, no. ~2, June 1976, pp. 80-86.

Gentleman, W. Morven. "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.

Humphrey, Watts S., and Kellner, Marc I. "Software Process Modeling: Principles Of Entity Process Models," Proceedings of the 11th international conference on Software engineering,, 1989, pp. 331 – 342.

Jacobson, Ivar., Booch ,Grady. and Rambaugh, James. The Unified Software Development Process , Reading , Massachusetts: Addison Wesley , 1998.

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.

Lichter, Horst, Matthias Schneider-Hufschmidt and Heinz Zullighoven "Prototyping in Industrial Software Projects," IEEE Transactions on Software Engineering, Vol 20 No. 11, 1989, pp. 25-832.

Liu, L. and Horowitz, E. "A Formal Model For Software Project Management". IEEE Transactions On Software Engineering. Vol. 15. N0. 10, October 1989, pp. 1280-1293.

Madhavji, Nazim. "The process cycle", Software Engineering Journal, IEE & The British Computer Society, September 1991, v. 6, no. 5, pp. 234-242.

Reinhard, Jung., and Winter, Robert. "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.

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

Somerville, I., Sawyer, P. and Viller, S. "Managing Process Inconsistency Using Viewpoints," IEEE Transactions on Software Engineering, vol. 25, no. 6, Nov.-Dec. 1999, pp. 784 –799.

Sutton, W. L. "Advanced Models Of The Software Process," Proceedings of the 4th international software process workshop on Representing and enacting the software process, 1988, pp. 156 – 158.

Trammell, Carmen J., Binder, Leon H. 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.

Yamamichi, N., Ozeki, T., Yokochi, K. and Tanaka, T. , "The evaluation of new software developing process based on a spiral modeling," Global, Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity, Volume: 3, 1996, pp. 2007 -2012.