An Effective Taxonomy For Software Process Models

Completed Research Track

Abstract:

A vast body of knowledge exists in the software engineering literature regarding software process models. The *evolution of process modeling has resulted in a diversity of methods that tends to suit software project requirements from different perspectives. This paper surveys the field of software process models in an effort to build an effective taxonomy that categorizes process models in terms of their functionalities and interrelationships. This taxonomy is a step toward tailoring process models to specific project requirements. The research methodology is carried out through a 4-steps grouping process: data collection, data analysis, data representation and data grouping, which involved a literature survey of existing process models. Based on this survey, two frameworks are introduced to address similarities and differences among software process models. Consequently, a comparison table is presented and utilized to establish taxonomy of software process models in an object-oriented fashion. Finally, directions for future research work are suggested.*

Keywords

Software engineering, process modeling, process assessment, process diversity, requirements engineering, project management, tailorability, software development, process models evolution

Word Count: 4,996

Pages: 18 (excluding the title page)

1.0 -Introduction

The diversity of software process models has been at the center of a wide range of research efforts for some time [1], [2], [3], [4], [5], [6], [7]. Although all process models share the goal of providing effective solutions for real world problems, they adopt different approaches in tackling problems based on the methodology utilized, technology afforded, business requirements raised or interdisciplinary impacts involved. These approaches include establishing limited frameworks [8], [9], [10], [11], [12], introducing new solutions, and expanding existing solutions toward better adaptability to project specific needs. While some of these research efforts concentrate on studying this diversity in terms of building a comprehensive understanding, other studies focus on developing new approaches to elicit [13], replace [14], [15] or unify existing process models [10], [16]. However, only some of these studies are focused explicitly on providing a comprehensive, well-categorized taxonomy that enable software project managers to efficiently select and configure software process models to the specific characteristics of their projects.

2- Problem definition:

A vast body of knowledge exists in software engineering literature regarding software process models. Each model needs to be tailored to the business and project requirements and characteristics in terms of quality and productivity goals [1]. This is better achieved by placing every process model in its proper location in a comprehensive framework (taxonomy) of all process models. Building this larger picture will enhance our understanding of the specific capabilities of every process model and how it can be tailored and utilized to specific project and organizational needs. This paper is an attempt to build such a comprehensive taxonomy for software process models based on a 4-STEP grouping process. The basic research question is how to determine an appropriate way of classifying software process models effectively.

3- Research Methodology: A 4-step methodology is adopted in this paper to classify software process models. This methodology is as follows:

A-Data Collection: Extracting Process Models From Literature survey:

In this step, a literature survey of software process models is conducted to extract the major streams in process models.

B- Data Analysis: Establishing Frameworks For Similarities And Differences:

In this step, similarities, differences, relations and rationales among process models are explored and frameworks are presented to address how process models share common goals and how they also have different characteristics.

C-Data Representation: Building The Comparison Matrix For Software Process Models

In this step, a comparison matrix between explored process models is introduced to establish a foundation for the grouping process.

D- Data Grouping: Class Diagram Taxonomy

In this step, a final framework of process models classification is concluded based on the previous steps. This framework is diagrammed in terms of a static object class hierarchy diagram that also depicts associations and inheritance.

4- Data Collection: Extracting Process Models From Literature survey

The software process is a framework for problem solving. To solve real world problems, software engineers must incorporate a development strategy that encompasses the process, the methods and the tools, which is typically referred to as a process model [8]. Before the age of software engineering, the code and fix model was the primary approach adopted [2] where solutions are developed regardless of careful problem analysis or serious requirements determination. 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 [5]. The waterfall was another improved version of the earliest process model named nine-phase stage-wise model. The V-shaped model was an extension of the waterfall with the incorporation of validation and verification procedures, especially in one of its latest modified versions [6].

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 [17], [18], we can recognize three significant characteristics of it: it is temporary, it is fast and it is a visual representation of the proposed system. It is also based on an evolutionary view of software development [18].

The popular spiral model, proposed by Boehm (1988), also exhibits a heavy reliance on prototyping [19] and software engineering economics [20], as it is mainly a risk-driven process model [5]. 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 [15] expanded spiral model to another version named "win-win spiral model" which is more customer-driven and user-centred.

Pressman [8] presented a comprehensive survey for process models in which he introduced the linear sequential (classical waterfall), prototyping model, RAD model, incremental model, spiral model, component assembly model, and concurrent development model as the most frequently used models. Somerville [9] included the evolutionary development, the formal transformation, and assembly from reusable components in his categorization. Behforooz and Hudson added the Department Of Defense (DOD) system development life cycle and the NASA model. Both of these models were waterfall driven.

Both iterative and incremental (sometimes called phased development [21]) share the goal of reducing the cycle time in the development process .The incremental model is based on building parts of the system in each release until the final system is completed. However, the Ada process model extends this discipline into three dimensions: subsystem increments, build increments and component increments [22]. In the iterative model, the whole system is developed in the first release

but improved iteratively in each of the following releases until achieving the most optimized system [11].

Furthermore, the iterative approach is the strategic framework for the unified process model suggested by the pioneers of the object-oriented UML at Rational Rose. The unified software development approach, proposed by Jacobson et al. (1998) [23], addressed some of the problems with previous models using an object-oriented approach and UML standards.

Abdel-Hamid et al. [24] introduced a dynamic model (1988-1991) to address management considerations coupled with software economics aspects. In 1987, IBM proposed another process modeling framework named the Cleanroom process model. It 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 [25].

Additionally, process models based on object-oriented techniques were also integrated throughout the software process modeling evolution. 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" [26].

In 1998 the Commercial of the shelf (COTS) approach in process modeling was proposed 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 [27].

Moreover, the Internet has had a significant impact on software process modeling in the last few years. Web development life cycle, recently referred to as web engineering, is also gaining an increasing interest in software development [28].

Furthermore, the business process-reengineering (BPR) trend in synchronizing business processes and software processes is being reflected in the reengineering process model. The TAME

process modeling approach also represents another step toward integrating process modeling with product metrics along with the automation capabilities of CASE tools in a more unified framework [1].

Integrating good practices has influenced the 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 [29]. Several later attempts also integrated 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 [14]. 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 [30].

5-Data Analysis: Establishing Frameworks For Similarities And Differences:

Although several factors contribute to the formation or development of process models, all process models aim to achieve common goals and share general characteristics regardless of their degrees of success or accomplishment. These common goals and characteristics of process models can be summarized as follows:

1- **Significant Role of requirements engineering:** Process models in general attempt to provide a solution from a relatively well-defined problem. However, there are different levels and degrees of problem definition and specification.

2- **Influence of waterfall model**: Process models in general adopt four common stages of software development (i.e.: analysis-design-code-maintenance) implicitly or explicitly in a sort of sequence. This implies the influence of the waterfall model regardless of what degree of linearity is incorporated [14].

3-**Reliance on documentation:** Process models in general rely on documentation and artifacts as the main tool to assure quality, planning, monitoring and tracability. It is also likely that his reliance is negatively correlated with the degree of automation and usage of CASE tools based on the primary purpose of using these CASE tools.

4- **Stakeholders involvement:** Process models in general attempt provide sufficient control over the software development process in order to achieve a valid and verified software product that meets stakeholders' requirements and expectations. Stakeholders are the primary driver for the software engineering process.

5-The project management dimension: Process models in general are forms of managing project complexity in a more efficient manner. Projects are practical implementations of process models strategies. The final goal is to produce cost-effective software solution within budget and on time.

6-Financial Goals: Obviously, the most significant goal of process models is financial success in terms of profit maximization, cost reduction [20] or customer satisfaction. A traditional way to express financial goals in the software engineering literature is to address them in terms of meeting deadlines, within the budget, and utilizing resources in an efficient manner [31].

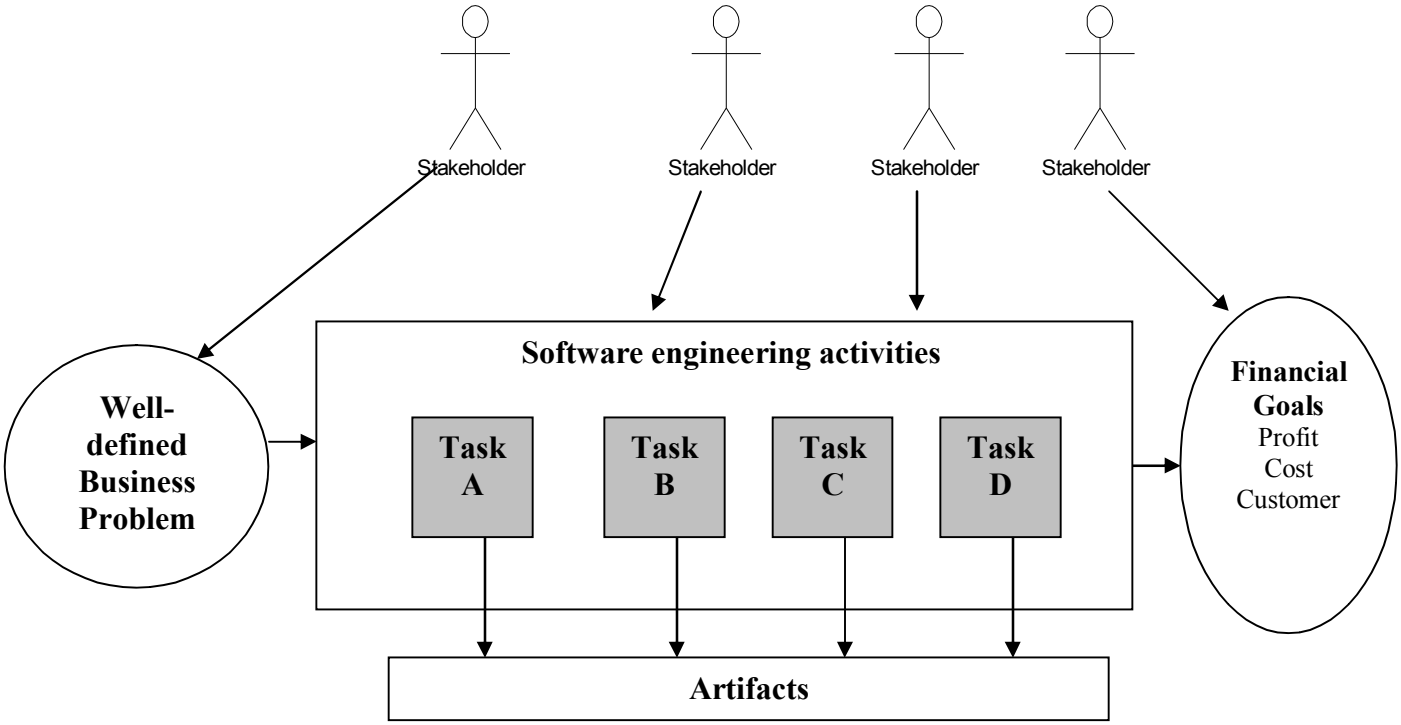


Fig.1. A framework of common characteristics among process models

Figure 1 presents a framework of common characteristics among process models. In this framework, a well-defined problem represents the significant role of requirements engineering input in the development process. Moreover, this element reflects the increasing influence of user involvement in all phases of the software development life cycle. Financial goals in Figure 1 represent the crucial outcome anticipated from the software development process, as there is no need for a software product that is not proven to be economically feasible. The third necessary element in this framework is stakeholders. Stakeholders could be direct or indirect users of the software product, people who influence the decision of determining system requirements, or developers and staff members involved in the development process The fourth element is the artifacts and deliverables. The fifth and final element is the tasks that should be followed to achieve a feasible solution from a well-engineered problem.

Despite these commonalities, process-modeling evolution is subject to accumulative refinements as a function of time and diversity in nature of projects and applications .It is also triggered by the changes in business environments, technological capabilities and evolution in software development

methodologies. Therefore, based on our survey above we can infer that the differences among software process models solutions can be attributed to one or more of the following influencing factors:

- **Time and previous work influence:** Time has played an important role in process modeling evolution. Taking earlier experiences into consideration, later models were based on careful evaluation of their formers. For example, the V-shaped was another version on the waterfall model but with different structure.
- **Technology:** Some models were functions of technology advancements over the years. For example, the rapid application development (RAD) approach is motivated by the introduction of CASE tools and 4GT techniques. Furthermore, the web engineering modeling frameworks are a reflection of the Internet age. According to Summerville, artificial intelligence utilizes the exploratory programming approach to emulate some human capabilities [9].
- **Interdisciplinary impacts**: Several models were the result of more interdisciplinary effects including psychological [14], managerial [24] and financial considerations as in [32],[5],[20].
- **Methodology and problem solving approach**: Several software process models are reflections of problem solvers' methodological approaches, such as sequential development as opposed to iterative development, and structured analysis and design versus object oriented analysis and design.
- **Problem Domain:** Application domain is another factor that triggers the evolution of software process models. DOD and NASA models are examples of domain-specific models.
- **Problem nature:** The nature of the problem addressed is primary driver in process modeling evolution. Problem nature encompasses three components in respect of business problems: size, structure and complexity.

- **Large and small:** While problems associated with large systems have triggered some approaches [33], other small-scale projects required solutions that are more scalable to suit their needs.
- **Problem structure**: The more complex are organizations the more ill structured are their problems [34] and the harder it becomes to take decisions at strategic levels.
- **Problem Added-Complexity:** Although problem structure and size are major drivers of problem complexity, there are other software-related and organizational-related elements that can add to complexity. For instance, there is a positive correlation between organization complexity and the impact of technical change [35].

- **Behavioral Considerations:** These considerations are the primary rationale for integrating system dynamics in process modeling. Process models that lack these considerations are more static in their structure [11], [30].
- **Critical Factors or drivers**: Process models tackle problems from different angles based on the major drivers of each process model. These drivers can be a major factor in categorizing process models and providing a profound understanding of their interrelationships. The major differences in process models are mapped in a schematic diagram as shown in Figure 2:

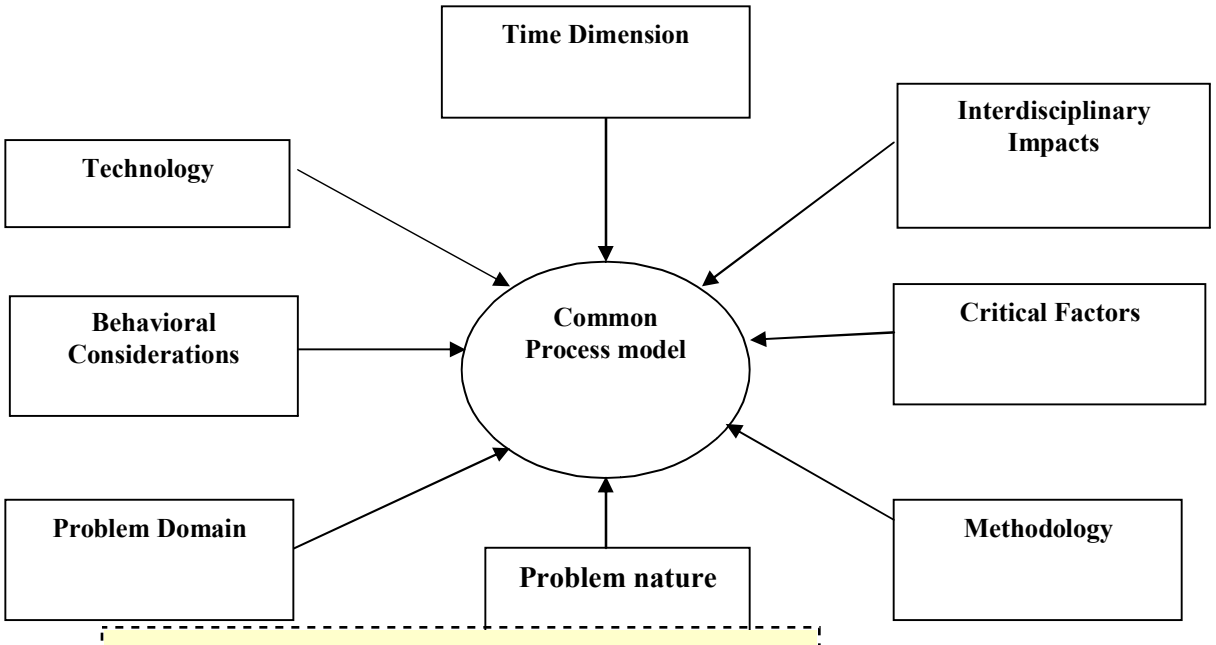


Fig 2. The context diagram of software process modeling

The above context diagram of software process modeling (Fig 2.) shows the eight most important factors impacting process modeling diversity as explored in the literature survey. The time dimension implies the evolution of software process models as a function of time while the interdisciplinary impact points out how several sciences and disciplines have influenced the development of software process models. Based on the literature survey, the time dimension has also triggered the change in technology, methodology and nature of business problems, which strongly impacted process models diversity as well. According to literature, behavioral considerations were the source of variation of several recent process models. Finally, many process models were focused on one or more critical factors as the main drivers for developing these models.

6- Data Representation: Building The Comparison Matrix For Software Process Models

Consequently, and based on the context diagram shown in Figure 2, a comparison table is developed as follows:

Process Model	Time Dimension (Evolution in goals)	Methodology	Technology	Critical Factors	Interdisciplinary Impacts	Behavioral Considerations	Problem nature	Application Domain
Waterfall	Solving stage-wise problems	Sequential and structured-oriented	Not- critical	Tasks	None	None	Large scale Projects	General
Prototyping Model	Overcoming late implementations in long cycles	Iterative	Can accelerate The process	User feedback	Psycho.	None	Small scale projects but can be integrated with other large-scale oriented models	General but more successful with artificial intelligence systems and user interface design
Evolutionary models	Overcoming sequential thinking	Iterative or incremental	Can accelerate The process	User feedback	Psychol.	None	Relatively small systems	General but more successful with artificial intelligence systems and user interface

								design
Incremental and iterative models	Overcoming sequential thinking	Iterative or incremental	Can accelerate The process	User feedback	None	None	Initial Shortage of resources and predicted technical risks	General
V-shaped model	Modified version of waterfall with more focus on quality assurance	Sequential	Not- critical	Tasks , where testing is related to analysis and design	None	None	Large scale Projects	General
Spiral model	Addressing risk assessment overlooked in previous models	Iterative with risk metrics	Recent automated tools are proposed for model generation	Risk Management	Economics	High user interaction specially in the win-win version	Mainly Large scale projects with high degree of uncertain	General
MIS-oriented model	Addressing time management and cost-benefit analysis more in depth (More business oriented than other models)	Sequential	Can be significantly optimized by CASE tools	Projects management.	MIS	None	Large and complex	Business information systems
4GT - based models	Function of available state-of-the-art Technologies	Automatic transformation And CASE tools	Totally dependent on software automation and process technology	Specification languages	AI	None	Used for both small and large systems but require more design considerations for large systems	Recently becoming able to address most software application categories
Rapid application development	High-speed adaptation of the waterfall model	Rapid Linear sequential development and Reuse	Can have great influence	Cycle Time reduction and reusable program components	None	None	Good for small systems but need sufficient human resources for large scalable systems	Some times not appropriate for high performance systems, high technical risks or when a system cannot be properly modularized
TAME	Improvement-oriented software development model	goal –question – metrics (GQM)	Initial prototypes	Feedback and measurements	None	High User involvement	More focus on tailorability for different Project requirements	General
CASE-tools based models Or automated development models	Supportive to several other models	Using waterfall with CASE tools support	Dependent on CASE tools	Software CASE tools	AI	None	None	General
Object-oriented process models	Overcoming structured-oriented problems	Object-oriented techniques and reusability	Can be extremely improved by CASE tools	Class objects components	None	None	Large and small systems	(More generic) Ability to work with cross-platform applications
Unified Software Development process	Capturing advantages and overcoming disadv	Object-oriented based on UML	Rationale rose ready – made	UML approach	Economic and management considerations	None	Large and small systems	General

	all previous models	modeling						
Component assembly model	Utilizing Software reuse advantages overcoming problems in structured paradigms	Object-oriented methodology and spiral model incorporation	Can be extremely improved by CASE tools	Class objects components	None	None	Large and small systems	(More generic) Ability to work with cross-platform applications
Assembly from reusable components model	A Japanese version of components assembly	Object-oriented from existing parts of the system		Existing system components	None	None	Large and small systems	General
Dynamic (management-oriented) model	Heavy focus on managerial considerations	System dynamics	Should be supported by software to capture links and quantitative descriptions due to high degree of complexity	Process Simulation	Management	Crucial specially with human resources	More adequate for Large systems	General
Behavioral models		System dynamics			Management			
Commercial-of-the-shelf "COTS"	Utilizing ready-made software solutions	Efficient Outsourcing and reusability to build cost-effective applications	Can be very effective	Ready-made reused applications	Economics	None	Might be difficult to manage change in complex environments which need high degree of flexibility and customization	Dependent on availability
Formal – based models	Focusing on accuracy and reducing ambiguity incompleteness and inconsistency for efficient verification	Mathematical Transformation	Highly dependent on Software automat.	Mathematical Specification	Math	Primarily, none	Complex systems with sufficient resources	Depending on level of staff training, available time and money, and types of customers
Cleanroom (IBM) model	Focusing on accuracy and reducing ambiguity incompleteness and inconsistency	Mathematical Transformation	Highly dependent on Software automation	Specification Language	Math	Primarily, none	Complex systems with sufficient resources	IBM but can be generalized
Concurrent development model	Capturing the richness of concurrency that exists across various project activities	Activity analysis with state identification		Activity status	Computer Engineering	None	Systems with concurrency and/or networking-architectures	General But more likely in client-server applications
Web-based (Web engineering) models	Response to internet requirements	More dependent on object – oriented modeling	CASE tools can be highly efficient when incorporated.	Web elements	None	None	Large and small web-systems	Web applications
Reengineering –based models	Dramatic changes over existing systems	Business Process-oriented utilizing reverse engineering techniques	IT is crucial	IT and human resources	Modern business	Can have significant influence	Complex and large systems	General but more likely with legacy systems with many problems

Process improvement models	Assessing and improving software product quality	Mainly CMM and ISO standards	Becoming strongly correlated with software automation	Customer satisfaction	Industrial engineering and marketing	Play important role	Large systems	General
Department of defense (DOD) model	A modified version of waterfall model	Sequential problem solving	None	Tasks with PDR Formal Reviews	None	None	Large systems	Department of defense
NASA model	Waterfall structure with slight difference in naming	Sequential problem solving	None	Tasks with function configuration audit	None	None	Large systems	NASA
Operational specification model	Another version of prototyping	Iterative	None	Early user involvement	None	High user involvement	Large and small	General
Resource and schedule driven model	Based on waterfall with very little formality and driven by schedule	Sequential problem solving	None	Tasks with certification testing incorporation	None	None	Large systems	General

6- Results: Data Grouping in Class Diagram Taxonomy

Based on the comparison table presented in the last section, we conclude that the following classes or grouping can capture the variety of process models in terms of shared characteristics, major differences and interrelationships amongst them. This is a primary taxonomy toward creating a comprehensive framework for software process modeling in general.

1. **Linear task-oriented models:** Sequential problem-solving approach applied generally on large-scale projects where activities decomposition is the core of this class. Long-term delivery is another characteristic for this class with the exception of rapid development models where models are maintain sequential approaches but designed to deliver software products more rapidly than the other subclasses. Members of this category include:

   Waterfall, V-shaped, MIS –Oriented, DOD, NASA, Concurrent, RAD and resource schedule models

2. **Reusable object-oriented components models:** Characterized by combinations of multiple process models and based on reusable components. These process models are also based on object-oriented methodologies. Members of this category include:

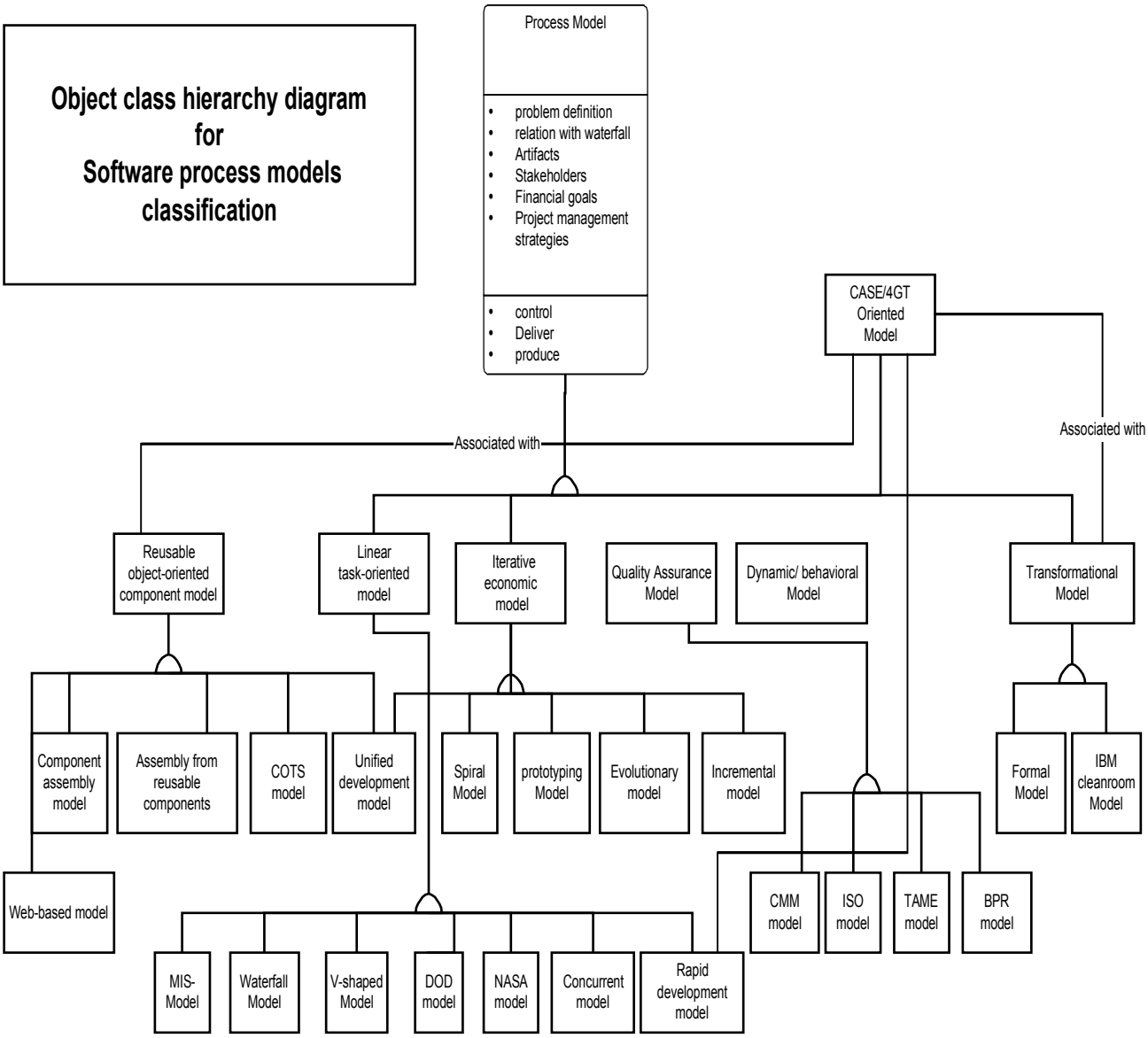
Component assembly, assembly from reusable components, COTS, unified development and web-based models. However, the unified development model (UPM) also exhibits multiple inheritance from iterative modeling as well.

3. **Quality assurance models :** Typically focusing on process improvement in terms of CMM or ISO standards . Many subclasses in this category are associated with CASE tools, software automation and IT advancements Members of this category include: ***capability maturity model (CMM), IS09000, TAME model and business process engineering (BPR) models***.
4. **Dynamic Models:** Behavioral and managerial considerations are crucial for this category. Heavy emphasis is on project control in terms of real world visualization and simulation. Therefore, Software automation can have a considerable effect on the efficiency of these models. Members of this category include: ***Abdel-Hamid's dynamic process model and behavioral models.***
5. **Iterative economic models :** Economic considerations in terms of risk management and users' early input are major factors in these models. Members of this category include: ***incremental model, prototyping model, evolutionary model, operational specification model, and spiral model in its different versions. The unified process model is part of this group in terms of its highly iterative manner.***
6. **Transformational models :** Math and specification languages for later software automation distinguish this category .However, it is still limited due to that lack of human resources and expensive implementation. Members of this category include: ***Formal model and IBM cleanroom model.***
7. **Fourth Generation Techniques Driven Models:** Although this category does not have independent members, it is associated with other members in other categories to enable other process models to generate more efficient results. Some of the other process models are totally dependent on highly automated techniques provided by 4-GT environment. Members

that belong to this category include: ***object-oriented models, RAD models and transformational models.***

Figure 3 demonstrates this classification in a **class hierarchy diagram** of process models based on Coad-Yourdan notations. This taxonomy is a step toward tailoring process models to specific project requirements. Therefore, it is part of the efforts towards creating a general or unified problem-solving framework in software engineering. This taxonomy also reflects the impact of technological advancements on the software development process.

Future extensions to this taxonomy would include additional process models that were not addressed in this study. Future studies may also consider an evaluation process to tailor super and sub-classes in this taxonomy to the variety of project requirements. Additionally, it is planned to examine whether this taxonomy can be generalized or replicated in the software engineering discipline. To have a reliable examination, the authors also intend to develop a relevant empirical study to reveal the statistical significance of this grouping process. A long term project would be to create a specialized CASE tool that can automate the utilization of this taxonomy to help in establishing a decision process in which project managers and decision makers can adequately tailor process models to their projects and organizational requirements.



References

[1] Victor R. Basili and H. Dieter Rombach. , "The TAME Project: Towards Improvement-Oriented Software Environments, " IEEE Transactions on Software Engineering, v. SE-14, n. 6, , June 1988, pp. 752-772.

[2] Watts S. Humphrey and Marc I. Kellner, "Software process modeling: principles of entity process models," Proceedings of the 11th international conference on Software engineering,1989, pp. 331 – 342.

[3] Sergio Bandinelli, Alfonso Fuggetta, Luigi Lavazza, Maurizio Loi, and Gian Pietro Picco, "Modeling and Improving an Industrial Software Process," IEEE Transactions on Software Engineering, vol .21,no.5, May 1995, pp. 440-454.

[4] Bill Curtis, Marc I. Kellner and Jim Over, "Process modeling," Communications of the ACM vol. 32, no. 9 , Sep. 1992 , pp. 75 – 90.

[5] Barry Boehm , "A Spiral Model of Software Development and Enhancement," IEEE Computer, vol.21, #5, May 1988 , pp. 61-72.

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

[7] H. Krasner, J. Terrel, A. Linehan, P. Arnold, and W.H. Ett. , "Lessons learned from a software process modeling system," Communications of the ACM, 35(9), 1992, pp. 91-100
[8] Roger Pressman, Software Engineering: A Practitioner's Approach, 4th Edition, New York, NY McGraw-Hill, ISBN 0070521824- 1438, 1996.
[9] Ian Somerville, Software Engineering, New York, NY, Addison-Wesley, ISBN 0-201-17568-1, 1995.
[10] Ali Behforooz,, Software Engineering Fundamentals, ISBN 0-19-510539-7, Oxford university press, New York, 1996.
[11] Shari Lawrence Pfleeger, Software Engineering: Theory and Practice, Upper saddle River, NJ: Prentice Hall Corp, 1998.
[12] Bruce I. Blum, "Taxonomy of Software Development Methods," Communications of the ACM 37(11), 1994, pp. 82-94
[13] Madhavji, N.H., Hoeltje, D., Hong, W. and Bruckhaus, T., "Elicit : A Method for Eliciting Process models, " Proceedings of the 3rd International Conference on Software Process, Reston, Virginia, 1994, pp. 111-122.
[14] Leveson, N.G., "Intent specifications: an approach to building human-centered specifications," IEEE Transactions on Software Engineering, , Volume: 26 Issue: 1, Jan 2000 , pp. 15 –35
[15] 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.
[16] Armitage, James W. and Marc I. Kellner. ,"A Conceptual Schema for Process Definitions and Models," Proceedings of the 3rd International Conference on the Software Process (Held at Reston, Virginia, USA, IEEE, October 1994, pp.153-165.
[17] Alavi, M., "An Assessment of the Prototyping Approach to Information Systems Development," CACM, 27(6), June 1984,pp. 556-563.
[18] 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.
[19] 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
[20] B. Boehm, "Software Engineering Economics," IEEE Transactions on Software Engineering, Vol. 10, No. 1, January 1984 , pp. 4-21.
[21] D. Graham, "Incremental Development and Delivery for large Software Systems " , Colloquium on Software Prototyping and Evolutionary Development, IEEE, November 1992.
[22] Royce W., "TRW's Ada Process model for Incremental Development of Large Software Systems, " TRW Technologies Series, TRW-TS-90-01, January 1990.
[23] Ivar Jacobson, Grady Booch and James Rambaugh, The Unified Software Development Process, ISBN: 0-201-57169-2, Addison Wesley, New York, 1998.
[24] 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
[25] 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.
[26] 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.
[27] 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.

[28] 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.
[29] Somerville, I.; Sawyer, P.; Viller, S., "Managing process inconsistency using viewpoints," IEEE Transactions on Software Engineering, Volume: 25 Issue: 6, Nov.-Dec. 1999, pp. 784 –799.
[30] 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.
[31] 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.
[32] Ropponen, J.; Lyytinen, K., "Components of software development risk: how to address them? A project manager survey," IEEE Transactions on Software Engineering, Volume: 26 Issue: 2, Feb2000, pp. 98 –112.
[33] Frank DeRemer and Hans H. Kron. "Programming-in-the-Large Versus Programming-in-the-Small," IEEE Transactions on Software Engineering, v. ~SE-2, n. ~2, June 1976, pp. 80-86.
[34] Mitroff, Ian and Murray Turoff, "Technological Forecasting and Assessment: Science and/or Mythology?" Journal of Technological Forecasting and Social Change 5, 1973, pp. 13-134.
[35] Keen, Peter G.W., " Information Systems and Organizational Change," Communications of the ACM, 24 (1), Jan 1981, pp. 24-33.