A Life-Cycle Taxonomy for Assessing Software Development Risks

Dr. Rashmi Jain, Sujoy Dey
Stevens Institute of Technology
Department of Systems Engineering and Engineering Management
Castle Point on Hudson
Hoboken, NJ 07030
USA
rjain1@stevens.edu, sdey1@stevens.edu

Abstract

Software development has become a necessary component of majority of the projects in this new world of Information Technology (IT). More or less all technology today has some form of software embedded in it. Therefore, it is imperative that we develop a good understanding of the risk involved in software development (what), their sources (why), and a process of identifying them (how).
Considerable research has been done in identifying risks in software development activity, factors impacting them, and some literature exists on how to measure them. However, existing literature has focused around creating a taxonomy of risks based on factors impacting them or sources of these risks. (Sherer, 1995) defined technical, organizational, and environmental as the three dimensions of software risk. (Boehm, 1988), (Clemens, 1991), and (Sherer, 1993) classified development, use and maintainability as components of software risk. (Chittister, 1993) proposed a framework grounded on a holistic concept introducing three perspectives or decompositions namely functional decomposition, source-based decomposition, and temporal decomposition.
This paper is based on the review of existing literature on software development risk assessment. It provides a more relevant and useful approach to identifying and assessing risks by taxonomising software development risks based on the Systems Engineering (SE) life-cycle concept. The SE life cycle concept describes the life-cycle of the software product as it passes through its phases of conception to operational implementation. Although there are numerous ways to define a system's life cycle from a SE point of view, the common stages or phases associated with a system that this paper will follow are requirements identification, detailed design, production and testing, operational implementation, evaluation and modification and operational deployment (Sage, 1992).
The authors believe that this approach will be more effective and result in savings. By identifying risks and associating them to a specific stage/phase of SE life-cycle may enable more efficient planning, management, and mitigation of risks through stage-containment.

Key Words: software development, risk taxonomy, risk assessment

1. Introduction

Software development lists a set of activities that results in software products. This may include new development, modification, re-use, re-engineering, maintenance, or any other activities that result in software products. The number of projects taken up each year is huge and the amount of resources put forth in these projects is staggering. Due to the importance of such projects, it is imperative that development is done on time, within budget, and with successful implementation. Although, significant progress has been made in software development methodologies, software project failures continue to exist.

The Chaos report from the Standish group shows 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns are just the tip of the proverbial iceberg. One has to also measure the loss in opportunity costs (The Standish Group, 1995). In the United States, more than $250 billion a year was spent on IT application development of approximately 175,000 projects. Although there has been little improvement in the project success compared to what it was in 1994, the number of failures is still very significant. In 1994, 16% of the projects met the criteria for success while in 2000 the success rate was 28%, which means the other 76% either failed or did not meet original goals. Tracking U.S. project outcomes showed that in 1994, 28,000 projects were successful, while in 2000, the number rose to 78,000—almost a threefold increase. Conversely, failed projects amounted to 54,000 in the 1994 study vs. 65,000 in the 2000 study. This was an 18% increase, while overall project growth exceeded 60%. Challenged projects grew at a rate of 62%, to equal 137,000 over the 1994 number of 93,000 (Johnson et al, 2001). Thus, it is not difficult to ascertain that there is still a lot to be done in order to increase the "successful" project rate.

2. Risk Factors in Software Development

Software development over the last decade has evolved to become an integral part of any technology implementation. More or less all technology today has some form of software embedded in it. However, software development is still not a perfect science. Several authors have written about the imperfections of software development as an activity. Majority of the imperfections are the result of decisions based on sometimes inadequate and incomplete information. The consequences of such decision making exposes software development projects to all kinds of risks. Identifying and understanding the source of such risks and their associated factors have been viewed as the first step to reduce risk (Boehm, 1991).

(Ropponen and Lyytinen, 2000) investigated and identified components of software development risk through a survey of project managers. These were: scheduling risks, system functionality risks, sub-contracting risks, requirements management risks, resource usage and performance risks, and personnel management risks.

(Murthi, 2002) discussed the importance of external risks as the category that most projects are likely to encounter. He proposes the following risk factors: requirements, technology, business, political, resources and skills, deployment and support, integration, schedule, maintenance and enhancement, design, and miscellaneous (the catch-all category).

(Keil et al, 1998) identified eleven risk factors as common to their survey panels of software project managers in different parts of the world. These were lack of top management commitment to the project, failure to gain user commitment, misunderstanding the requirements, lack of adequate user involvement, failure to manage end user expectations, changing scope/objections, lack of required knowledge/skills in the project personnel, lack of frozen requirements, introduction of new technology, insufficient/inappropriate staffing, and conflict between user departments.

(Addison and Vallabh, 2002) in an empirical study on risk factors in software projects analyzed the 'importance' and 'frequency of occurrence' of risks as perceived by project managers at different levels of experience. These risk factors were unclear or misunderstood scope or objectives, misunderstanding the requirements, failure to gain user involvement, lack of senior management commitment, developing the wrong software functions, unrealistic schedules and budgets, continuous requirement changes, inadequate knowledge or skills, lack of effective project management methodology, and gold plating, (additional capabilities make systems more attractive).

The revised 2001 list of Standish Group's success factors ("recipe for success") for application development projects, known as the "Chaos Ten" includes the following in order of importance. These are executive support, user involvement, experienced project manager, clear business objectives, minimized scope, standard software infrastructure, firm basic requirements, formal methodology, reliable estimates, and other criteria. Each factor was weighted according to its influence on project success.

A review of the risk factors included by the authors discussed above points out to some common sources of risks such as requirements, scheduling, functionality, user involvement, resources/skills, management support etc. (Ropponen and Lyytinen, 2000) mentioned two additional factors of resource usage and performance risks. Some additional sources of risk for software development included by (Murthi, 2002) are deployment and support, and integration, maintenance and enhancement.

2.1 Review of Traditional Risk Taxonomies

Table 1 summarizes the existing software development risk taxonomies. Boehm's model begins each cycle of the spiral by performing the next level functionalities in elaboration of the prospective system's objectives. These iterations build on these functionalities but may not end with usable software after each subproject in the iteration (Murthi, 2002). (Boehm and Bose, 1994) presented an extension of the spiral model; called the Next Generation Process Model (NGPM), which uses the Theory W (win-win) approach to converge on a system's next level objectives, constraints, and alternatives. This NGPM model addressed the need for concurrent analysis, risk resolution, definition and elaboration of both the software product and the software process in a collaborative manner.

(Sherer, 1995) identified the three dimensions of software risk. The first was the technical dimension resulting from uncertainty in the task and procedures. The second was the organizational dimension resulting from poor communication and organizational structure. Finally, the third dimension was environmental, which results from rapidly changing environments and problems with external relationships with software developers and/or users. (Chittiser, 1993) proposed a framework termed hierarchical holographic modeling where three perspectives or decompositions are introduced. They were 1) functional decomposition, which encompasses seven basic attributes associated with software development- requirement, product, process, people, management, environment, and system development; 2) source-based decomposition, which relates to the four sources of failure- hardware, software,

organizational, and human; and 3) temporal decomposition, which relates to the stages in the software development process.
Rowe's model provides a four-stage sequential lifecycle approach to risk assessment and management. His model is comprehensive and generic at the same time, and as a result its application is not restricted to any specific domain.

Table 1: Existing risk models

Model	Reference
Barry Boehm's Model	Boehm (1988)
Next Generation Process Model	Boehm and Bose (1994)
Software Risk Evaluation Model	Sista and Joseph (1994)
Three-dimensional Software Risk Model	Sherer (1995)
Hierarchical Holographic Modeling	Chittister (1993)
Rowe's Causative Events	Rowe (1997)
Sage's Model of Risk Management	Sage (1992)
Model of Individual behavior on Failure/Success	Walsh and Schneider (2002)
Iterative Model	Murthi (2002)

(Walsh and Scheider, 2002) proposed that the behavior of decision makers affected by risk propensity and motivation is critical to the outcome of a software project.
Murthi's model is compared to the spiral model. The spiral model has iterations built on functionalities but may not end with usable software after each subproject in the iteration. Murthi's model has a single iteration itself as a project resulting in software that users will actually use (Murthi, 2002).
An analysis of the sources of risks included in the above discussed models reveals four major categories of risks in some form or shape. These are human, organizational, technical, and environmental. These models are based on a one-time identification of risks as project risks rather than on an on-going evaluation of risks over the entire life-cycle of the project. Therefore, these models lack phase-specific identification and analysis of risks. An approach based on phase-specific identification and assessment of risks would be better to plan for, manage, and mitigate risks. This paper proposes such an approach.

3. Systems Engineering Life-Cycle Concept

All systems have a life-cycle defined as a series of steps, stages, or phases, beginning from its early conception to the phasing out or disposal at the end of its useful life. What is critical to the applicability of the systems engineering process is the thorough understanding of the system life-cycle. These life-cycle phases as identified and described below by (Sage, 1992) progress as follows:

3.1 The identification of requirements and specifications.

This phase identifies stakeholders' needs, activities, and objectives for a fully functional operational system. This phase should result in the identification and description of preliminary conceptual design considerations for the next phase.

3.2 Preliminary conceptual design.

This phase specifies the content and associated architecture and general algorithms for the system product in question. Development of a conceptualized prototype that is responsive to stakeholder requirements should result.

3.3 Logical design and system architecture specification.

This phase has detailed design and architectural specifications of the product.

3.4 Detailed design, production, and testing.
User/stakeholder confidence should be high so that the design results in a useful product. If this does not happen, phase three should be redone.

3.5 Operational implementation.
The product is ready for operational implementation. A general checklist for this phase is based on the success of the previous phases. The checklist is coding, testing, a recommended installation approach, sufficient documentation, and training and support readiness.

3.6 Evaluation and modification.
Evaluation of the detailed design and the resulting product, process, or system is achieved in this phase. There should be provisions for objective measures and critical evaluation measurements.

3.7 Operational deployment.
Final acceptance and operational deployment of the system occurs in this phase.

4. Identification and assessment of Risk

The goal for the first phase in the systems life cycle is a complete, unambiguous, and understandable documentation of requirements. The primary risk attributes for this phase are ambiguity, completeness, understandability, volatility, and traceability of requirements (Buttigieg, 2004). These attributes directly impact the potential success of the overall development project. For example, poorly written, incomplete and constantly changing requirements become a primary source of risk. Such ambiguous and inadequate requirements often lead to confusion, result in risks to the later phases of development, including "scope creep".
Buttigieg's description of risks during the design phases includes attributes like ambiguity, completeness, structure or architecture, and coupling. Design is critical to the future stages of development. Complex design needs highly experienced programmers and more attention to detail. Lack of standard design formats and metrics in this phase of the life cycle are often ignored or omitted from risk evaluation (Buttigieg, 2004). This can lead to a ripple effect of such risks throughout the later phases.
Evaluation, testing, and modification phases should have well defined guidelines to maintain quality and integrity of the product. Measurability is an important aspect which is often ignored in most evaluations. Any attribute that cannot be measured also cannot be evaluated.
Operational implementation and deployment phases have in them a major problem in change management and training. Changing existing work practices to fit the new developed system is a major difficulty. Similarly, it is tough to draw the line between changing and/or retaining the business processes to suit the system. There are a few organizations that maintain in-house experience to implement and maintain a large-scale integrated solution.
The approach stressed in our paper is the one of risk identification and assessment *within each phase or stage of software development.* This approach hypothesizes that risk factors are specific to a certain phase of the software development process. It hypothesizes that the same risk factors are not relevant for all the different phases and that even if they do their importance and impact on the success of the project differs in the different phases. For example, human factors such as skills of the team members may be a more important factor in the detailed design phase than in the operational deployment. Therefore, if we can identify the risk factors unique to each phase it will be easier to mitigate and manage these risks with minimal costs and disruption.

We base our approach on the concept and process of ‘stage containment’ as illustrated in Figure 1. Stage containment is the practice of striving to catch and resolve (stage specific) errors found in each stage of the development life-cycle of a product. This prevents problems being passed on to the next stage. This stage containment concept helps improve testing methodologies as well as build quality within each phase of the system development process.

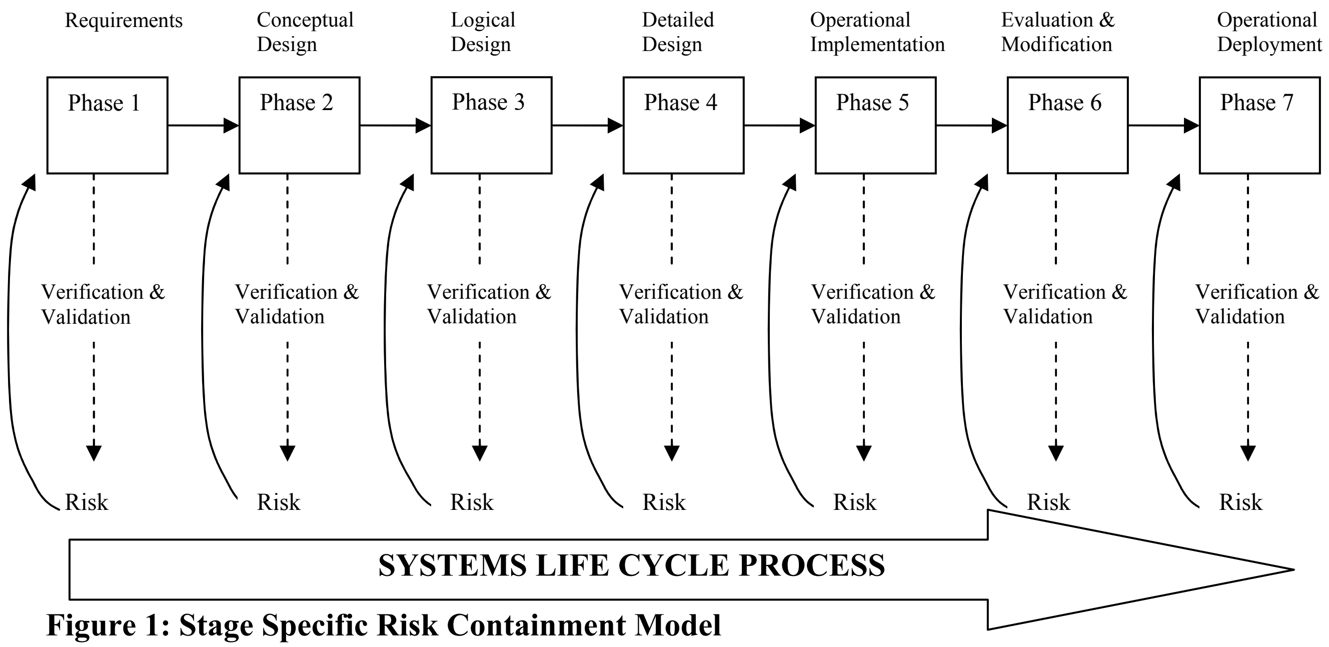

Figure 1: Stage Specific Risk Containment Model

The phases in the Systems Life Cycle process (Figure 1) as described by Sage are shown in the blocks as we travel from the left to the right (denoted by the arrow at the bottom). Complete verification and validation procedures are outlined within each life cycle phase. Each phase or stage has its own risk that directly or indirectly affects the life cycle. These risks get identified during (and as a result of) the continuous verification and validation procedures outlined for that particular phase. In this process of stage containment, each risk should be analyzed and resolved to the satisfaction of the stakeholders within that phase before continuing on to the next phase.

At this stage it is important to illustrate with simple and generic examples of errors, defects, and faults; the benefits of stage containing risks. We also discuss how these errors, faults, and defects impact the development effort and its success. For example, if we make a design **error** and it is found and corrected in the same design stage, then we have contained the error.

However, if we find the problem while in coding or testing stage, then the problem has become a **defect**. If the problem is not found until production, it becomes a **fault**. Errors, defects, and faults cause additional rework, additional cost, and additional time to complete a project. Stage containment helps reduce the likelihood that defects and faults will occur by catching and fixing them when they are just “errors” which can be fixed at very minimal impact to the project.

In order to implement “stage containment” for identifying and mitigating risks we need to identify clearly defined standards/criteria for each stage. These predefined standards must be met before exiting one stage and entering another. These standards are therefore the entry and exit criteria. These may include criteria covering skill set of the team members, traceability, documentation, testing methodology, quality, and functional, technical, and procedural requirements. Stage containment of software development risks will be deemed to have been achieved only when the exit criteria of the current stage and entry criteria of the next stage are fulfilled.

5. Expected benefits of this approach

The proposed life cycle taxonomy for assessing software development risks as illustrated in Figure 1 is expected to result in several benefits. As mentioned earlier, tremendous savings can be achieved in terms of time, effort, cost, with better quality, as a result of identifying and mitigating the risk factors in the stage/phase where they happen. Risk factors remaining unidentified in the phase where they occur may impact the later phases in terms of additional rework, additional cost, additional time to complete a project, and in some cases even untimely termination of projects. Whereas, observing and implementing stage containment process would help reduce the likelihood of carrying over risk factors and their impacts into phases subsequent to where they occur.

Moreover, by associating certain risks with a specific phase of software development will help in better planning and more efficient management of software development efforts. Development teams will understand and expect the exposure they have to different kinds of risks at every stage of development. This will help them mitigate these risks and also plan resources and time to manage such risks.

6. Recommendation and Conclusions

This paper is based on a review of existing literature and not on empirical research. The proposed taxonomy has not been implemented, researched, or validated. Informally, some software development projects may already be using this approach in some form or the other. However, no research has been conducted on the effectiveness and impact of the approach on the success of a software development projects. Further research can be done to ascertain if risk factors are unique to a specific phase of software development project or if some of them overlap over one or more phases. Certain measurable criteria will need to be defined or existing definitions of risk factors may be used. Surveys of software development projects will need to be done to validate the approach and ascertain if such an approach of assessing software development risk results in cost savings and other benefits.

References:

1. Addison, Tom, Vallabh, Seema, *Controlling Software Project Risks – An Empirical Study of Methods used by Experienced Project Managers*, Proceedings of SAICSIT, 2002.
2. Boehm Barry, Bose Prasanta, *A Collaborative Spiral Software Process Model Based on Theory W*; IEEE, 1994.
3. Boehm, B.W., *Software risk management: principles and practices, IEEE Software*, 1991, 8(1), 32-41.
4. Buttigieg, D. Anton, Risk Management in a Software Development Cycle, 2004.
5. CHAOS, The Standish Group Report, 1995.
6. Chittister, Clyde, *Risk Associated with Software Development: A Holistic Framework for Assessment and Management*; IEEE Transactions on Systems, Man, and Cybernetics, 23(3), May/June 1993.
7. Johnson, Jim, Boucher, D. Karen, Connors, Kyle, Robinson, James, *Collaborating on Project Success*, February-March 2001.
8. Keil, Mark, Cule, E. Paul, Lyytinen, Kalle, Schmidt, C. Roy, *A framework for identifying software project risks*, Communications of the ACM, v.41 n.11, p.76-83, Nov. 1998.
9. Murthi, Sanjay; *Preventive Risk Management for Software Projects*, IEEE, 2002.
10. Ropponen, Janne, Lyytinen, Kalle, *Components of Software Development Risk: How to Address Them? A Project Manager Survey*, IEEE Transactions on Software Engineering, 26(2), February 2000.
11. Sage, P. Andrew, *Systems Engineering*, 1992, John Wiley & Sons, Inc.
12. Sherer, A. Susan, *The Three Dimensions of Software Risk: Technical, Organizational, and Environmental*, Proceedings of the 28th Annual Hawaii International Conference on System Sciences, 1995.
13. Walsh, R. Kenneth, Scheider, Helmut, *The role of motivation and risk behavior in software development success*, Information Research, 2002, 7(3).

Biography

Dr. Rashmi Jain is an Associate Professor of Systems Engineering at Stevens Institute of Technology in New Jersey, USA. Dr. Jain has over 15 years of experience of working on socio-economic and information technology (IT) systems. Over the course of her career she has been involved in leading the implementation of large and complex systems engineering and integration projects.
Dr. Jain's research interests include network-centric systems, systems integration, and agile systems engineering. Dr. Jain is a Co-Chair of the INCOSE ABET Working Group and a member of the American Society for Engineering Management and Agile Alliance. She holds Ph.D. and MS degrees from Stevens Institute of Technology.

Sujoy Dey is a Research Assistant in the Systems Engineering and Engineering Management Department at Stevens Institute of Technology. Mr. Dey teaches courses in systems engineering and engineering management, namely, simulation and modeling, production and operations management, and engineering economic design.
Mr. Dey's research interests include systems modeling and simulation, and decision analysis under risk and uncertainty. He is a student member of the International Council of Systems Engineering (INCOSE), and the Project Management Institute. Mr. Dey holds an MS degree in Technology Management and is currently working on completing his PhD this summer in Systems Engineering.