Projects Characteristics Determining Suitability of Software Development Process

Michel S. Soares[1] and Joseph Barjis[2]
[1]*Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil*
[2]*Section Systems Engineering, Delft University of Technology, Delft, The Netherlands*

Keywords: Software Development, Characteristics of Software Projects, Software Quality, Project Management.

Abstract: Software development is a complex engineering activity that faces many challenges. In the last three decades, software projects frequently ran over time, cost over budget, and delivered less functionalities than promised. These challenges are dominantly caused by technical and management failures. One critical challenge is that project characteristics, such as the technical and management environment, are not well-known in advance. The preposition of this paper is that knowing the characteristics of a software project will have a positive effect on three indicators that are most important for stakeholders: costs, duration, and functionalities. This preposition is discussed and validated in relation to three sources: review of the literature, analysis of lessons-learned documents of completed software projects in industry, and interviews with practitioners. This research resulted in a list of characteristics (non-exhaustive) that should be taken into account during the initial phases of a software project development, which should increase likelihood of success.

1 INTRODUCTION

The rate of failures in software projects is currently very high, as reported in many publications (Pich et al., 2002) (Lee and Xia, 2005) (Cicmil and Hodgson, 2006) (Lewis, 2010) (Clarke and OConnor, 2012). Both researchers and practitioners have attempted to identify not only the causes of project failure but also the factors that lead to success. As a result, standards have been developed and disseminated in order to report what has been observed (PMI, 2004) (Cicmil and Hodgson, 2006). Currently there is still only limited research evidence that links adherence to these project standards to have improved project performance (Thomas and Mullaly, 2007). Even delivered software projects normally face future issues such as difficulty of management and evolution (Boehm and Beck, 2010). Disappointments with the return of investment by stakeholders are common. Software is viewed by many chief executives as one of the major problem areas faced by large corporations (Glass, 1998) (Yourdon, 1999) (Ewusi-Mensah, 2003) (Ward, 2012).

The success of a software project depends not only on technical soundness, but also on the perception of every involved stakeholder. Often, this perception is based on three indicators: cost, time, and functionalities (requirements) (Lewis, 2010). In the literature, the influence of these three characteristics on the success of software projects is well-known. According to (Reifer, 2001) and (Dybå and Dingsøyr, 2008), the size of a project, considering budget and duration, is imperative for software development success. Furthermore, activities related to requirements, such as eliciting and managing requirements, are considered key success factors by many authors (Hofmann and Lehner, 2001) (Komi-Sirviö and Tihinen, 2003) (Damian et al., 2004) (Minor and Armarego, 2005) (Soares et al., 2011).

Basing the discussion primarily on these three core characteristics, this article explores additional characteristics that may influence the suitability of a software project. The research question is stated as follows: *What characteristics of software projects influence the suitability of software development*? The research methods we used in this paper are literature review, followed by studying lessons learned documents from completed software projects in industry, and interviews with practitioners. Therefore, additional characteristics are derived from the three initial ones. These characteristics are independent of industry domain and software development methodology. The research described in this article was performed at a large financial organization that not only develops

S. Soares M. and Barjis J..
Projects Characteristics Determining Suitability of Software Development Process.
DOI: 10.5220/0004419501180125
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 118-125
ISBN: 978-989-8565-60-0

software internally, but also hire a number of vendors and consulting companies to develop their software projects.

The remainder of this paper is structured as follows. In Section 2, we discuss software development challenges from both technical and managerial angles. In Section 3, we discuss the main characteristics of software projects found respectively after performing an extensive literature review, searching lessons learned documents, and interviewing with practitioners. The main lessons learned are discussed in Section 4. Finally, we conclude the paper and present proposals for future work in Section 5.

2 SOFTWARE DEVELOPMENT CHALLENGES

The study of the complexity of software development is taken into account through two perspectives in this section: the technical and the managerial. It is not our intention to exhaust all technical and managerial challenges, but to point out the most common ones based on literature review.

2.1 The Technical Challenge

The development of software has always been challenging (Wirth, 2008). Currently, a great objective in modern society is to develop successful software respecting constraints such as costs and deadlines, and being able to maintain and evolve these systems. Among the technical challenges involved to achieve this objective, we can mention the Software Engineering sub-activities' of Requirements Engineering, Software Architecture, and Software Process Improvement (Nuseibeh and Easterbrook, 2000) (Hofmann and Lehner, 2001) (Komi-Sirviö and Tihinen, 2003) (Kruchten et al., 2006) (Brown and McDermid, 2007).

It is common-knowledge in Software Engineering that correctly performing activities of the Requirements Engineering discipline (elicitation, documentation, analysis, management, and so on) is crucial for software development (Berry, 2004). When any of these activities are poorly performed, the software project is at risk of failure (Abran et al., 2004). According to a number of practitioners and researchers, Requirements Engineering is the most problematic phase of the development of a software system (Damian et al., 2004) (Minor and Armarego, 2005). Dealing with ever-changing requirements is considered the real problem of Software Engineering (Berry, 2004).

Requirements are often misunderstood, misinterpreted, and poorly documented (Berry, 2004) (Minor and Armarego, 2005) (Walia and Carver, 2009). A major issue is that requirements are often written at only one level of abstraction, or requirements at different levels of abstraction are mixed, which brings even more confusion to stakeholders (Soares and Cioquetta, 2012). Requirements specifications for large systems should be specified at a number of levels of abstraction (Berander and Svahnberg, 2009), as for instance, user requirements and systems requirements (Sommerville, 2010). Another issue is that the majority of modeling languages are tailored to document requirements at only one level of abstraction. For instance, high level user requirements are often modeled using UML Use Cases. However, Use Cases are unsuitable to model more specific system level requirements (Lilly, 1999). The same holds for structured natural language (Cooper and Ito, 2002), which is used for algorithm specification and implementation phases, but is considered difficult to understand by most stakeholders.

An additional important success factor within requirements is related to non-functional requirements. These requirements are normally associated with restrictions and quality properties, and can be so important to software that they may determine its success. For example, a financial system that is perceived as sufficiently fast most of the time and has a nice interface is condemned to fail if stakeholders and clients have doubts about its security.

The importance of software architecture in the software life-cycle is widely recognized in theory and practice by many authors. A proven concept to build software and to avoid problems and failure causes, such as poor communication among stakeholders and sloppy and immature development practices, is to have a well-defined software architecture (Bass et al., 2003) (Kruchten et al., 2006). According to (Booch, 2007), having an architecture allows the development of systems that are better and more resilient to change when compared to systems developed without a clear architectural definition. Software architecture is also considered one of the most significant technical factors in ensuring project success (Brown and McDermid, 2007). The software architecture affects the performance, robustness, and maintainability of a software system (Bosch, 2000).

Another common problem in software development is the poor relationship between the requirements engineering and the software architecture teams (Hall et al., 2002) (Medvidovic et al., 2003). It is of utmost importance that both groups work together (de Boer and van Vliet, 2009). The main rea-

son is that the software architecture must be considered when engineering new requirements to be implemented in a software product (Miller et al., 2010). The requirements document will have a great influence on the architectural style to be chosen and the decisions that tailor the software architecture. On the other hand, each architectural style is more or less capable of addressing a different number of non-functional requirements. Therefore, requirements engineering activities must be closely related to the design of the software architecture.

The theme *software process improvement and capability determination* is controversial. It can be considered to have a positive factor for software productivity and quality (Pino et al., 2008). However, it is also considered too expensive to implement, and only useful depending on human and sociological factors (Baddoo and Hall, 2002). The evaluation of the software development process is often given by levels of maturity such as the ones of CMMI (Chrissis et al., 2003),

2.2 The Project Management Challenge

There is a debate by researchers and practitioners about what is considered to be failure or success of a software project (Thomas and Fernandez, 2008). Some authors have considered the cancelation of a project even before its start (Emam and Koru, 2008), safety problems of the delivered software (Leveson, 2004) (Dulac and Leveson, 2009), or insufficient business and goals results (Sauser et al., 2009) to be symptoms of failure.

However, there is more or less consensus (Linberg, 1999) (Zhang et al., 2003) (Lewis, 2010) that a software project is considered failed when at least one of the following factors are present:

1. The project cost is higher than the planned budget.
2. The project duration is greater than planned.
3. Not all required functionalities are delivered.

Our consideration of software project failures during the research took into account these three factors only.

The total cost and duration of a software project are inherently dependent on the project size. The difficulty here is to measure software. A number of techniques were proposed in past years, with varying degrees of complexity. Among the most well-known techniques, we can mention LOC (Lines of Code), Function Point Analysis, estimation based on Use Cases, and the COCOMO model (Pressman, 2010). However, the estimation of the size of a software project still suffers from high inaccuracy (Molkken and Jrgensen, 2003) (Jorgensen and Shepperd, 2007) (Magazinius et al., 2012).

3 PROJECT CHARACTERISTICS

The characteristics presented in this section were found respectively after performing an extensive literature review, searching lessons learned documents of completed software projects in industry, and interviewing with practitioners.

3.1 Literature Review

From the literature review, the following characteristics were found.

3.1.1 Risk Clearness

Managing risks in software projects has long been recognized as crucial (Brooks, 1987) (Boehm and Ross, 1989) (Han and Huang, 2007) (Bannerman, 2008). The degree of risks differs significantly according to previous knowledge of the domain. Therefore, for projects based on previous executed projects, risks are notably reduced. However, for projects that largely differ from previous ones, risks have a significant impact.

3.1.2 Requirements Maturity

Uncertainties and lack of maturity in requirements documents have been recognized as major issues in software development (Jiang et al., 2002) (Hickey and Davis, 2004). The characteristic “requirements maturity” takes into account how stable requirements are after have been elicited and documented, and the degree of changes expected.

The more mature the project is for each stakeholder, the more stable the requirements are. Stakeholders usually have an idea of what he or she wants, although no clear requirements are stated. This may create high levels of uncertainty for the entire project. Furthermore, when requirements are not mature, changes can easily occur while developing the actual software, resulting in over budgeted and over time projects.

3.1.3 Client’s Commitment

This characteristic takes into account how involved the client wants to be with the project and its progress. The degree can vary from simply stating requirements

to actually be involved in all phases of project development. This commitment is part of agile methodologies (Eckstein and Baumeister, 2004) such as XP (Beck and Andres, 2004). The importance of high involvement of clients is to avoid surprises in the end. The client continuously evaluate the progress of the project by means of prototypes and associated documents on a daily basis. On the other hand, this approach is more likely to increase the level of changes during project development.

3.1.4 Team Formation

Software projects can differ greatly with the amount of people working on the project. This does include developers, managers, clients, and personnel from the vendors.

Choosing the correct team size and scheduling each person to tasks and activities of a project are problematic concerns for project managers (Brooks, 1995). Not to mention knowing in detail the abilities of each team member, and where each one can make the personal best contribution to the project. Therefore, choosing the team and assignning responsibilities are important characteristics of a software project which influences the suitability of a software project.

3.2 Lessons Learned Documents

Some departments within the company where this research was performed evaluate finished projects. This activity is well-known as post-mortem (Birk et al., 2002), and is part of some software development methodologies, as for instance, the Personal Software Process methodology (Humphrey, 1996).

The purpose os these documents at the company are to state what went right, what went average, and what went wrong and should be improved. These documents were used to help finding the characteristics of projects described as follows.

3.2.1 Scope Clearness

In case the scope is not clear, team members can get lost in the project, and time and budget overruns are imminent. In many lessons learned documents, the clearness of scope was mentioned as an issue to be improved. Almost always unclear scope leads to frequent changes in requirements, which reflects on higher costs and longer project duration.

3.2.2 Department Influence

Within large organizations such as the one where this research was performed, multiple departments might play a role in large projects. Departments frequently offer services to each other. One issue here is that some departments may have stronger political position in the company than others. It is imperative to know how each department influences the project, and the role of each department in relation to the others.

3.3 Interviews with Practitioners

The interviews were performed with personnel throughout the organization. The target were managers and senior software designers involved with making important decisions about the software project. In addition, personnel from the vendor side were interviewed in order to get information from a different viewpoint. The general purpose of all interviews was to get in-depth information. By discussing tacit knowledge, opinions and practice related issues are uncovered. The two initial questions, which derived further discussion, were: “What characteristics are the most important ones for project success?”, and “How the characteristic affects a software project?”. The issues presented here are hardly described elsewhere. In total, 14 employees were interviewed, including software architects and managers.

3.3.1 Environmental Stability

This item relates to the environment to develop software, including facilities such as software tools, development infrastructure, methods, techniques and suppliers. If the environment to develop software is not stable, then frequent changes and additional risks can be expected.

3.3.2 Applications’s Familiarity

Projects developed at the refereed organization can differ in difficulty and complexity. One major factor for this is whether the application requested is already familiar to the development team. If the application is completely new, then additional efforts are expected from developers in order to understand the domain and the application. In case of introducing a new software platform or tool, the difficulty is also increased.

3.3.3 Stakeholders Relationship

This characteristic becomes more important with the size of the team and the project. The relationship between team members and other involved stakeholders, such as vendors and members from other departments, is of fundamental importance. It is not unusual in a large company that departments provide services

to other departments and groups. Therefore, cooperation is performed not only internally between team members, but also between teams and other groups of the organization. As a matter of fact, not only the relationship between team members should be considered, but also between groups and departments.

4 DISCUSSION

In this section, we try to relate back to our initial discussion. In particular, we draw our discussion on technical and managerial perspectives of software development challenges and reason in line with the core software development challenges as well (cost, duration, functionality) as the additional characteristics we have identified. For each one of the three factors we classify the list of characteristics found as follows.

4.1 Project Budget

Characteristics that are connected to the factor Project Budget are risk clearness, scope clearness, requirements maturity, client's commitment, environmental stability, department influence and team formation. If risks and scope are not clear, the project budget will rather represent unrealistic estimates. A significant number of over budget and over time projects are caused by uncertain boundaries. The stability of the development environment is of high importance for the project budget. If the infrastructure and techniques used are unreliable, then the budget becomes unreliable as well. In addition, with high level of client's commitment, it is more likely to increase the level of changes during project development, and budget can be overrun. Furthermore, when requirements are not mature, changes can easily occur while developing the actual software, resulting in over budgeted and over time projects. As some departments may have different political position in the company, the budget of a project and its priority will depend on how strong the department is inside the organization. Finally, choosing the correct team size and scheduling each person to tasks and activities of a project according to individual abilities will influence the final costs of the project.

4.2 Project Duration

The characteristics that influence the factor Project Budget are often related to the factor Project Duration. In addition, another characteristic was found that only influences Project Duration. This is the familiarity the project group has with the application to be developed. Certain projects are adaptations of already existing software. However, some projects concern entirely new innovative software applications. If this is the case, additional time is needed for the project group to get familiar with the domain.

4.3 Functionalities

The final factor mentioned is related to Requirements/Functionalities. Three characteristics were found that have influence on or are influenced by this factor. The first characteristic is requirements maturity. For many software projects, the stability of requirements is of the utmost importance. Changes at the end or during the project could cause enormous problems. The second characteristic is clients' commitment. For elicitation and collection of explicit and unambiguous requirements, clients need to be committed to the project. The final characteristic is the stakeholders relationship since requirements management is dependent on their wishes.

5 CONCLUSIONS

The research that led to this paper was performed at a large financial institution that develops software internally and also hire consulting companies and vendors to develop software. As the rate of failures in software projects is currently very high, in this paper we wanted to search for characteristics of a project that would have a positive influence on software development. The successful completion of a project depends both on technical factors, such as well-defined and well-described requirements, and on managerial factors, such as feasible estimations. We focused this research mainly on the project management point of view. The non-exhaustive list of characteristics described in this paper should be considered as success factors for software development projects.

Two main lessons were found in this research. The first one is that investigating projects after their completion is very useful. The two main characteristics found after performing this activity, Scope Clearness and Department Influence, are hardly described elsewhere. In addition, knowing what went wrong will help managers in making future decisions. The second one is that the point of view of practitioners should be taken into account, as they do not necessarily have the same opinions as the researchers. Performing interviews with developers and managers that face real problems everyday is useful to understand less conventional success factors that are often

neglected in mainstream theoretical frameworks and discussions.

Future work will focus on improving the validity of the proposed concept by following the same approach in another application domain. Particularly, we are interested in evaluating software projects applied to critical infrastructure systems, such as electricity, water and road traffic networks.

ACKNOWLEDGEMENTS

The authors would like to thank CNPq (www.cnpq.br) for the financial support.

REFERENCES

(2004). *A Guide to the Project Management Body of Knowledge (PMBOK Guides)*. Project Management Institute.

Abran, A., Bourque, P., Dupuis, R., Moore, J. W., and Tripp, L. L., editors (2004). *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA, 2004 version edition.

Baddoo, N. and Hall, T. (2002). Motivators of software process improvement: an analysis of practitioners' views. *Journal of Systems and Software*, 62(2):85–96.

Bannerman, P. L. (2008). Risk and risk management in software projects: A reassessment. *Journal of Systems and Software*, 81(12):2118 – 2133.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. Addison-Wesley Professional, Reading, MA, USA.

Beck, K. and Andres, C. (2004). *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional.

Berander, P. and Svahnberg, M. (2009). Evaluating Two Ways of Calculating Priorities in Requirements Hierarchies - An Experiment on Hierarchical Cumulative Voting. *Journal of Systems and Software*, 82(5):836 – 850.

Berry, D. M. (2004). The Inevitable Pain of Software Development: Why There Is No Silver Bullet. In *Radical Innovations of Software and Systems Engineering in the Future*, Lecture Notes in Computer Science, pages 50–74.

Birk, A., Dingsoyr, T., and Stalhane, T. (2002). Postmortem: never leave a project without it. *Software, IEEE*, 19(3):43–45.

Boehm, B. and Beck, K. (2010). Perspectives. *IEEE Software*, 27:26–29.

Boehm, B. W. and Ross, R. (1989). Theory-w software project management principles and examples. *IEEE Trans. Softw. Eng.*, 15:902–916.

Booch, G. (2007). The Economics of Architecture-First. *IEEE Software,*, 24(5):18–20.

Bosch, J. (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. ACM Press/Addison-Wesley Professional, New York, NY, USA.

Brooks, F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *Computer,*, 20(4):10–19.

Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley Professional, Boston, USA.

Brown, A. W. and McDermid, J. A. (2007). The Art and Science of Software Architecture. *International Journal of Cooperative Information Systems,*, 16(3/4):439–466.

Chrissis, M. B., Konrad, M., and Shrum, S. (2003). *CMMI Guidlines for Process Integration and Product Improvement*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Cicmil, S. and Hodgson, D. (2006). New Possibilities for Project Management Theory: a Critical Engagement. *Project Management Journal*, 37(3):111–122.

Clarke, P. and OConnor, R. V. (2012). The Situational Factors that Affect the Software Development Process: Towards a Comprehensive Reference Framework. *Information and Software Technology*, 54(5):433–447.

Cooper, K. and Ito, M. (2002). Formalizing a Structured Natural Language Requirements Specification Notation. In *Proceedings of the International Council on Systems Engineering Symposium*, volume CDROM index 1.6.2, pages 1–8, Las Vegas, Nevada, USA.

Damian, D., Zowghi, D., Vaidyanathasamy, L., and Pal, Y. (2004). An Industrial Case Study of Immediate Benefits of Requirements Engineering Process Improvement at the Australian Center for Unisys Software. *Empirical Software Engineering,*, 9(1-2):45–75.

de Boer, R. C. and van Vliet, H. (2009). Controversy Corner: On the Similarity Between Requirements and Architecture. *Journal of Systems and Software*, 82(3):544–550.

Dulac, N. and Leveson, N. (2009). Incorporating safety risk in early system architecture trade studies. *AIAA Journal of Spacecraft and Rockets*, 46(2):430–437.

Dybå, T. and Dingsøyr, T. (2008). Empirical Studies of Agile Software Development: A Systematic Review. *Inf. Softw. Technol.*, 50:833–859.

Eckstein, J. and Baumeister, H., editors (2004). *Extreme Programming and Agile Processes in Software Engineering*, volume 3092 of *Lecture Notes in Computer Science*.

Emam, K. E. and Koru, A. G. (2008). A replicated survey of it software project failures. *IEEE Software*, 25:84–90.

Ewusi-Mensah, K. (2003). *Software Development Failures*. MIT Press, Cambridge, MA, USA.

Glass, R. L. (1998). *Software Runaways: Lessons Learned from Massive Software Project Failures*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Hall, J. G., Jackson, M., Laney, R. C., Nuseibeh, B., and Rapanotti, L. (2002). Relating Software Requirements and Architectures using Problem Frames. In *Proceedings of IEEE International Requirements Engineering*

Conference, pages 137–144. IEEE Computer Society Press.

Han, W.-M. and Huang, S.-J. (2007). An empirical analysis of risk components and performance on software projects. *Journal of Systems and Software*, 80(1):42 – 50.

Hickey, A. M. and Davis, A. M. (2004). A unified model of requirements elicitation. *J. Manage. Inf. Syst.*, 20:65–84.

Hofmann, H. F. and Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software*, 18:58–66.

Humphrey, W. S. (1996). Using a defined and measured personal software process. *IEEE Software*, 13(3):77–88.

Jiang, J. J., Klein, G., and Discenza, R. (2002). Perception differences of software success: provider and user views of system metrics. *Journal of Systems and Software*, 63:17–27.

Jorgensen, M. and Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Trans. Softw. Eng.*, 33(1):33–53.

Komi-Sirviö, S. and Tihinen, M. (2003). Great Challenges and Opportunities of Distributed Software Development - An Industrial Survey. In *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'2003)*, pages 489–496.

Kruchten, P., Obbink, H., and Stafford, J. (2006). The Past, Present, and Future for Software Architecture. *IEEE Software*, 23(2):22–30.

Lee, G. and Xia, W. (2005). The Ability of Information Systems Development Project Teams to Respond to Business and Technology Changes: A Study of Flexibility Measures. *European Journal of Information Systems*, 14:75–92.

Leveson, N. G. (2004). A systems-theoretic approach to safety in software-intensive systems. *IEEE Trans. Dependable Secur. Comput.*, 1:66–86.

Lewis, J. (2010). *Project Planning, Scheduling, and Control: The Ultimate Hands-On Guide to Bringing Projects in On Time and On Budget.* Mc-Graw-Hill, New York, NY, USA, 5 edition.

Lilly, S. (1999). Use Case Pitfalls: Top 10 Problems from Real Projects Using Use Cases. In *TOOLS '99: Proceedings of the Technology of Object-Oriented Languages and Systems*, pages 174–184, Washington, DC, USA. IEEE Computer Society.

Linberg, K. R. (1999). Software developer perceptions about software project failure: a case study. *Journal of Systems and Software*, 49(2-3):177 – 192.

Magazinius, A., Brjesson, S., and Feldt, R. (2012). Investigating Intentional Distortions in Software Cost Estimation An Exploratory Study. *Journal of Systems and Software*, 85(8):1770 – 1781.

Medvidovic, N., Grünbacher, P., Egyed, A., and Boehm, B. W. (2003). Bridging Models Across the Software Lifecycle. *Journal of Systems and Software*, 68:199–215.

Miller, J. A., Ferrari, R., and Madhavji, N. H. (2010). An Exploratory Study of Architectural Effects on Requirements Decisions. *Journal of Systems and Software*, 83(12):2441 – 2455. TAIC PART 2009 - Testing: Academic and Industrial Conference - Practice And Research Techniques.

Minor, O. and Armarego, J. (2005). Requirements Engineering: a Close Look at Industry Needs and Model Curricula. *Australian Journal of Information Systems,*, 13(1):192–208.

Molkken, K. and Jrgensen, M. (2003). A review of surveys on software effort estimation. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, ISESE '03, pages 223–231, Washington, DC, USA. IEEE Computer Society.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements Engineering: a Roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 35–46, New York, NY, USA. ACM.

Pich, M. T., Loch, C. H., and Meyer, A. D. (2002). On Uncertainty, Ambiguity, and Complexity in Project Management. *Management Science*, 48:1008–1023.

Pino, F. J., García, F., and Piattini, M. (2008). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Control*, 16:237–261.

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA.

Reifer, D. J. (2001). Manager - Software Management's Seven Deadly Sins. *IEEE Software*, 18(2).

Sauser, B. J., Reilly, R. R., and Shenhar, A. J. (2009). Why projects fail? how contingency theory can provide new insights - a comparative analysis of nasa's mars climate orbiter loss. *International Journal of Project Management*, 27(7):665 – 679.

Soares, M. S. and Cioquetta, D. S. (2012). Analysis of Techniques for Documenting User Requirements. In *12th International Conference on Computational Science and Its Applications - ICCSA*, volume 4, pages 16–28.

Soares, M. S., Vrancken, J. L. M., and Verbraeck, A. (2011). User requirements modeling and analysis of software-intensive systems. *Journal of Systems and Software*, 84(2):328–339.

Sommerville, I. (2010). *Software Engineering*. Addison Wesley, Essex, UK, 9 edition.

Thomas, G. and Fernandez, W. (2008). Success in IT projects: A matter of definition? *International Journal of Project Management*, 26(7):733–742.

Thomas, J. and Mullaly, M. (2007). Understanding the value of project management: first steps on an international investigation in search of value. *Project Management Journal*, 38(3):74–89.

Walia, G. S. and Carver, J. C. (2009). A Systematic Literature Review to Identify and Classify Software Requirement Errors. *Information and Software Technology*, 51(7):1087 – 1109. Special Section: Software Engineering for Secure Systems - Software Engineering for Secure Systems.

Ward, J. M. (2012). Information Systems Strategy: Quo Vadis? *The Journal of Strategic Information Systems*, 21(2):165–171.

Wirth, N. (2008). A Brief History of Software Engineering. *IEEE Annals of the History of Computing,*, 30(3):32–39.

Yourdon, E. (1999). *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Zhang, L., Lee, M. K. O., Zhang, Z., and Banerjee, P. (2003). Critical success factors of enterprise resource planning systems implementation success in china. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 8 - Volume 8*, HICSS '03, pages 236–242, Washington, DC, USA. IEEE Computer Society.