IBIMA Publishing
Journal of Software & Systems Development
http://www.ibimapublishing.com/journals/JSSD/jssd.html
Vol. 2017 (2017), Article ID 171474, 11 pages
DOI: 10.5171/2017.171474

Research Article

Bridging the Gap between RFP and SDLC: How to Meet the Challenge with a Software Development and Implementation Plan (S-DIP)

Harvey Hyman

Library of Congress, National Library Service, Washington, DC

Received date: 10 May 2016; Accepted date: 28 July 2016; Published date: 27 January 2017

Academic Editor: Wan Nurhayati Wan Ab. Rahman

Abstract

This paper presents a discussion on addressing the gap that exists between the drafting of a Request for Proposal (RFP) and the successful delivery of a system that meets the expectations of stakeholders and the needs of end-users. The Software Development and Implementation Plan (S-DIP) framework is proposed as a solution to the problem by applying the theory stages of the Software Development Lifecycle (SDLC) to the RFP documentation. The framework supports a regimented vetting process to provide a better translation of stakeholder's goals and end-user needs to the RFP document and improve alignment between RFP with SDLC, project management execution, and stakeholder expectations. A study of the S-DIP framework across five organizations has found it to provide better insight into the end-users' needs for the proposed system, better clarity in terms of priorities and consequences in managers' decision making, and improvement in both quality and satisfaction in the final delivery of the system, especially on the enterprise level.

Keywords: Request for Proposal (RFP), Project Management, Software Development, Software Development Life Cycle (SDLC), Enterprise Systems, Business Process Reengineering (BPR).

Introduction

Over the past 20 years, numerous academic papers as well as industry consulting groups have consistently reported on the staggering failure rate for software development projects, especially enterprise level projects (DeMarco and Lister, 2013; Nelson, 2005; McConnell, 2001). Likewise, many of these same studies and industry reports have suggested a variety of possible explanations for this phenomenon of project failure. These explanations range from failure in the planning stage to incompetence in the execution stage (Nelson, 2005; McConnell, 2001). What the vast majority of the reported failures have in common is the recurring misalignment between system requirements and the final software product.

In fact, the subject of software project failure has become an almost ubiquitous "baked-in" expectation, accounting for predictions in cost overruns, late deliveries, missed milestones, and under-performance in features, functions and usability. Frequent industry status updates such as the Standish Group's Chaos Report, often site the economic costs associated with the latest round of software system failures to meet the end-user needs and stakeholder organizational goals.

This paper takes a fresh look at the software development process and explores how a particular aspect of the process might shed light on this recurring problem – the Request for Proposal (RFP).
The "RFP" as it is known by practitioners and managers alike, has grown from its apocryphal beginning to become the ubiquitous core document for system delivery and project management – it is the alpha and the omega, from which the entire project is sourced, from procurement to execution.

The Problem Restated as a Research Question

The problem, as documented in many reports and studies, is that quite often, RFPs do not produce the desired results for end-users or stakeholders (Saito et al., 2012). In fact, all too often, the RFP starts off as a vague and nebulous, yet aspirational document, representing the hopes and dreams for a new system, or the improvement in performance and responsiveness for a legacy system (Wilson, 1993; Lehman, 2005). Unfortunately, time and again, the contents of the RFP never fully translate end-user needs with system requirements (Porter-RothBud, 2004; Saito et al., 2012).

A common complaint documented in industry is the apparent gap between the RFP and the final system as delivered: RFPs frequently do not reflect the true, underlying goals of management; do not result in a clear development plan to accurately meet system requirements; and repeatedly lead to dissatisfied end-users – habitually necessitating scrapped work and rework (Boehm and Turner, 2003; Nelson, 2005; McConnell, 2001). The problem is restated as the research question addressed in this paper: How can we bridge the gap between RFP and SDLC?

Motivation

Perhaps the most interesting place to start a discussion about the subject of RFP is to acknowledge that it is an entirely industry created approach to system design and development (Finkelstein et al., 1996). The underlying assumption for its use is that it seeks to optimize the process for soliciting bids in the procurement of a service or product (Andrea, 2003; Davy, 2011). That being said, it is curious how there have been little to no significant studies or reviews exploring RFP as a conceptual topic, framework for analysis, or process oriented model. This might cause the reader to consider the conventional wisdom which typically suggests that where there is little study or review on a subject matter, there is usually nothing significant to explore.

But, how can we fail to critically review such a ubiquitous documentary procedure such as the RFP, especially in light of the fact that it is the gateway entry point for virtually all large scale industry and governmental procurements of systems? Likewise, with the importance and consequence in the vast effort of research struggling to explain the phenomenon of project failures, and the evolving paradigms in software engineering, software development and project management (Demarco and Lister, 2013; Nelson, 2005; McConnell, 2001; Hyman, 2013), isn't it time to examine the underlying assumptions and possible causes and effects that the RFP may play as a role: How is it that the RFP has sneaked up to become the de facto standard for the commencement of a system development project; and how is it that the RFP has gone almost completely un-scrutinized for so long?

The Origination and Evolution of RFP

Historical references for how the RFP came into existence are nebulous and obscure. Lacking a definitive introduction, it seems to have quietly creeped up like slow growing moss, to become the decisive procedural device for modern system and software acquisitions. A review of early articles from the 1960s and 1970s treat the RFP as having always existed, like the "steady state universe," incorporating it by reference but never truly establishing a "big bang" instantiation moment. However, as part of a preliminary discussion, these articles do provide significant insight about the original intent and purpose of the RFP, and are quite informative on its foundation in form, substance and use.

In these early references, RFP is existentially defined as: "very detailed, and [was] by itself, the performance and design requirements", and "at a level of detail that it was, by itself, the implementation concept and test plan" (Wolverton, 1974).

When we trace the roots of the RFP document and its associated process, we find that, ironically, it has evolved mainly from the domain of Cost Estimation. In fact, substantial work can be found on the subject matter and how the RFP is a definitive document in the evaluating and negotiating processes within the context of Cost Estimation (Wolverton, 1974; Saito et al., 2012; Andrea, 2003).

One of the earliest definitive references to RFP as a process, in and of itself, for software procurement is a 1979 article by Leland Coonce entitled "Use of Request for Proposals for Software Purchase Selection." Unfortunately, this obscure article, seemingly goes on to be cited by no one, and even the broadest web search queries run by this author have revealed no companion articles to offer to the reader.

However, the Coonce article does provide us with at least a frame of reference for RFP, operationally. In fact, in the intervening years from the 1980s to the 2000s, RFP goes largely unnoticed and quietly becomes an accepted underlying assumption for Cost Estimation and Project Management subject matter articles. It reemerges as an exploratory topic in the early 2000s, and even then, it is treated rather lightly, once again, as an untested assumption and foundation for narrative discussions on Project Management (Davy, 2011; Nelson, 2005; Hyman, 2013).

As an example of the modern treatment of the RFP we find a simple article in the Hudson Valley Business Journal, appropriately entitled "How to Create a Great RFP" (Ladke, 2013). The significance of its mention here is that, like many of the self-declared expert references and free advice sources on the subject of the RFP, little more than narrow opinion pieces and ad hoc "best practices" are offered to assist the curious practitioner (Davy, 2011). Rarely do we come across a significant and serious study offering a framework or model to enhance the use of the RFP as a vetting device for system and software procurement (Royce, 1970; Wilson, 1993; Saito et al., 2012; Lehman, 2005; Davy, 2011; Hyman, 2013).

"When done right, RFPs enable businesses and government agencies to fairly evaluate competing proposals while reviewing the broadest possible range of potential solutions. All too often, however, RFPs fail to meet these goals, because their creators simply don't understand how to create them effectively" (Porter-RothBud, 2002).

The Gap between RFP and Successful Delivery of a Software Project

Even the best executed project management will not rescue a poorly vetted system, or a system that does not fully reflect the goals of stakeholders or the needs of the end-users (Hyman, 2013). The RFP process, as currently implemented by industry does not align with the SDLC stages of *planning, analysis, design* and *implementation*.

The RFP process is broken down into four stages of *specification, proposal, evaluation,* and *implementation* (Andrea, 2003). Specification refers to the customer description of stakeholder requirements and end-user needs. This stage results in the release of the RFP document. The Proposal stage refers to the vendor response to the RFP. In the response, the vendor "assesses the requirements and delivers their response, which includes their proposed solution" (Andrea, 2003). The flaw that lies within this methodology is that, even assuming a vetting process in the Evaluation stage, too much of the analysis is left up to the customer in stage one, and too much of the design is left up to the vendor in stage two.

The gap lies in the reliance on the RFP to perform a function it was never intended to perform – act as a framework for system development. The SDLC has been a framework for system development since the 1980s (Royce, 1970; Boehm and Turner, 2003; Hyman, 2013). The RFP as a fully vetted supporting document produced during the SDLC process can lead to a well-designed system to be executed by project management leaders and development personnel.

However, this has not been the case; instead of a supported document, the RFP has evolved into a substitute procedural framework for the SDLC, leading to breakdowns in the development process (Hyman, 2013). Relying on the RFP as a framework in and of itself, rather than as a supporting document, will often result in the vendor proposal of stage two, being "a shot in the dark." There are two main reasons for this. First, the Specification stage of the RFP encompasses the first two stages of SDLC, planning and analysis; whereas the Proposal stage of the RFP encompasses the isolated SDLC stage of design, and nothing to establish continuity among these three stages with implementation. Second, the vendor is relying on the customer to provide a set of fully vetted system requirements. This is unrealistic, and in fact, this false assumption can be directly traced to numerous examples of failure in the planning stage of development, due to reliance upon wrongly deduced guidelines provided by the customer (McConnell, 2001). It is flawed thinking to expect the customer to be capable of providing complete and full system requirements vetting (Hyman, 2013). Many a failed project can trace its roots to this mistaken approach (Nelson, 2005).

The thesis of this essay is that the stages of RFP do not align with the stages of SDLC, and for good reason – the RFP was an evolving trend in documentation and never intended to be an alternative to substitute for the SDLC (Davy, 2011; Hyman, 2013). Therefore, how can we expect successful delivery of any project planned under one paradigm (RFP process) but executed under a completely different paradigm (SDLC process)?

It is certainly no mystery that we see so many project failures and negative end-user reports from the field, when we recognize the gap between the misalignment of planning and analysis, with design and implementation. The S-DIP presented in this paper is not the only attempt at addressing this gap. There have been other, albeit a few, attempts to address the gap between the "system and its environment" (Dardenne et al., 1993; Mylopoulos and Castro, 2000; Castro et al., 2011). One such recent example is the "Tropos Project" designed to model "early and late requirements" (Castro et al., 2011). The *Tropos* framework approach, as described by Castro et al., follows a "Requirements Driven Development Methodology." They apply *Tropos* to a case study of a "Media Shop" in an attempt to harmonize structured development techniques with programming paradigms.

Building on *Tropos* and other similar works targeting the gap between requirements and development, the framework of S-DIP is described in the next section along with a narrative summary of its implementation and roll out, in various development projects at several organizations.

The Framework of S-DIP

This section describes the proposed framework and model of S-DIP – designed to harmonize the RFP process with the SDLC model.

The S-DIP is a six stage process that has been developed based upon interviews of managers and developers at five different organizations by tracking the chain of events in their development processes, RFP processes, documentation produced during the processes, interactions with vendors before and after the release of the RFP documents, follow-up reports of "items falling through the cracks," tracing of scrapped work and rework, user acceptance testing reports, end-user satisfaction, and stakeholder interviews.

Based on the responses and documentation collected, the S-DIP has been designed to address the need for a supporting framework and model to guide an acquisition through a vetting process ending with a robust and detailed RFP document, and to provide a benchmark methodology for a thorough and detailed vendor selection and qualification procedure. There are four commonalities that were discovered to have occurred during the development processes at all of the organizations studied – these are identified as four basic principles that the S-DIP framework addresses.

The first principle that emerged from the study was that every organization conducted three categories of core activities in any system or software development project: Acquisition, Integration, and Implementation. The first thing we realized when we analyzed the data was that each of these categories really were life cycles themselves, and that many of the "failures" that were observed in the development process could be traced to conflating some or all of these activities. For example, we found the vetting process that takes place during the acquisition of a system is a life cycle in and of itself, given that every system begins with its procurement, whether internally sourced or externally vendor procured. It cannot be short circuited or rushed by combining it with another process or stage of development.

Likewise, integration is a category containing a collection of activities that should be treated as a life cycle of its own, with a focused placed on how the newly procured system will blend into the current way the organization performs its business activities. We found, that all too often, the activities associated with integration were among the most under-estimated by the procurement team.

The category of implementation is the third core set of activities. Questions that need to be thoroughly vetted in this stage are: How will this new system be installed? How will our transition plan impact the way we migrate our business activities to the new system? How will our maintenance plan and release schedule impact our work flow processes?

The second principle that emerged was to take a "clean slate" approach to system development. Meaning we start from scratch, by asking two simple questions: What do we want to accomplish with this system? What resources do we have to develop this system? The projects that began with a clean slate approach produced more robust and more detailed documentation.

The third principle is the application of an Input-Process-Output (IPO) model to every process and sub-process performed by the current or new system. This principle was a late addition to the study. It was proposed to several of the project teams during the study. The teams that applied the IPO model to their RFP process reported that fewer items "fell through the cracks" and they had a substantial reduction in scrapped work.

The fourth principle developed was to iterate and evaluate, at each and every stage of the S-DIP. By conducting frequent evaluations at each stage of the process, the study found that fewer "mistakes were baked in" to the final system design because they were discovered early enough in the process that they could be rooted out. We depict the above described activity categories in a Five-Stage Evaluation Model adapted from the original SDLC, displayed in Figure 1 below.

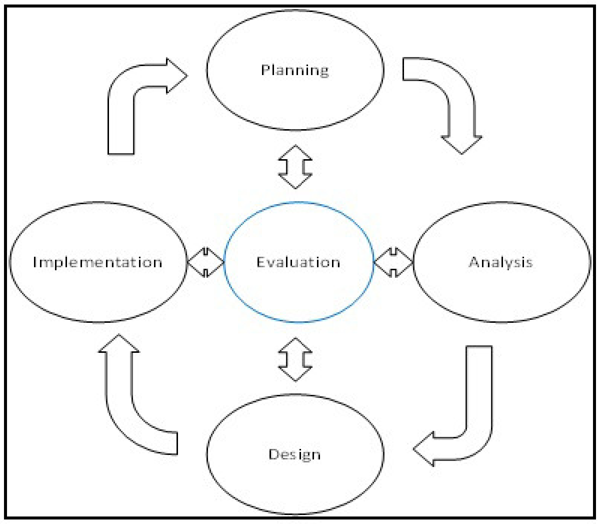

Figure 1: Five Stage Evaluation Model (Hyman, 2013)

The S-DIP itself is an adaptation of the stages of the SDLC and several of the "reviews" found and described in the MIL STD 1521b Technical Reviews and Audits for Systems, Equipments, and Computer Software (1986) and the IEEE 12207-2008 Software Life Cycle Processes (1995, 2008). When managers and developers were asked during the study about the MIL STD and IEEE processes, many reported that they were familiar with both of the processes, but that they rarely used them.

When asked why, the responses included: "too tedious", "not practical", "inconsistent with how we work here" – and no wonder, the MIL STD 1521b was released in 1986, and the IEEE 12207 has been largely unchanged since 1995 – how can we expect anyone to follow procedural guidelines that are 30 and 20 years old respectively? The general consensus among the practitioners about the SDLC was that "it is good in theory," but does not translate well to the focus of "project management and the PMBOK."

To address the concerns voiced in the practitioner interviews, the S-DIP adapts SDLC stages, and MIL STD and IEEE reviews, to the RFP process by categorizing them into six review phases, based on the three core activity categories discovered during the study: System Requirements Review, Preliminary Design Review, Critical Design Review, Test Readiness Review, System Specification Review, and Implementation Readiness Review. The phases are intended to act as final reviews, validating the correctness of the activities and verifying the build for that phase of documentation – with the ultimate goal being the production of a detailed supporting document for vetting and evaluating vendors in the RFP process. The next paragraphs provide brief descriptions of the six review phases of the S-DIP as depicted in Figure 2.

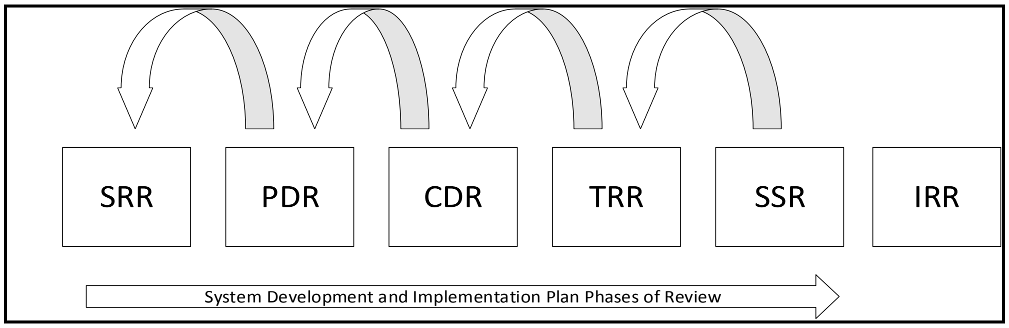

Figure 2: System Development and Implementation Plan (S-DIP) Framework Model (Hyman, 2013)

The System Requirements Review Phase of the S-DIP serves as a structured initial investigation into the system. Stakeholders are identified, requirements are derived from interviews and user stories, and use cases are proposed to represent system level tests to verify when requirements have been delivered. This phase is completed with a full review of all identified system requirements as validated. Validated means that each requirement has been documented and mapped by stakeholder, system module, use test case, and quality criteria. The key here is in being disciplined to not move forward in the RFP process until there is satisfaction that the requirements review as defined above has been met. Until it is met, the RFP document does not move forward.

Once system requirements have been reviewed, the RFP moves to the Preliminary Design Review Phase. During this phase a draft of the statement of work (SOW) is generated. The SOW is based on a work breakdown structure (WBS) that defines exactly what will be built. Every requirement must be broken down to its component features. Every feature must be broken down to its lowest, indivisible functions. The RFP document initiated in the Requirements Phase is further annotated with priorities and criticalities for each requirement. This is too important to leave to the vendor in the post RFP release.

The S-DIP puts the analysis and responsibility on the procuring organization, and does not abdicate it to the vendor. Risk identification, assessment, mitigation and contingency planning is established and considered as well. A completed Preliminary Design Phase should contain documentation supporting traceability for the SOW, WBS, and an updated requirements document with annotated priorities. If the evaluation review at the end of this phase results in a “no go” decision, then we revert to the previous Requirements Phase to refactor and revalidate that we in fact have correctly identified and analyzed the requirements.

The next phase is Critical Design Review. This phase assumes that the Preliminary Design Review has been verified. This phase is comprised of activities supporting design and build of visual display and visualization techniques. We found that organizations applying the S-DIP, would build small working prototypes in this phase for stakeholder and end-user feedback. At the very least, mockups of user interface screens are designed and validated by the end-user. The traceability documentation produced in this phase typically included screen shots from the end-user validated interface display screens. Managers reported that during this phase greater clarity is discovered about the details of the system and how it will operate. They also reported that, as a result, it was not uncommon, and it was actually viewed as a sign of good development and attention to detail, if the Preliminary and Critical Review Phases resulted in reversion to prior phases, even all the way back to Requirements Phase for refactoring and revalidation.

The Test Readiness Review assumes that the project has passed Critical Design Review. During the Test Readiness phase, test cases are verified for each development level: unit, integration and system. Test Readiness proved to be a significant milestone because it turns out to be the last feasible time in the project life cycle that the RFP documentation process can be reverted without significant time delay or delivery slippage. Documents produced and evaluated in this review are the Test Plan, Test Description and Test Report for evaluating the system's performance metrics.

System Specification Review is the next phase. The significance of this phase is that the RFP documentation has survived four iterated reviews and is now being evaluated for complete commitment of resources. During this phase the physical and logical specifications of the system are declared and described. In this phase hardware, software, operating system, communication links, system operators and administrative personnel are identified, allocated and budgeted. The system specification should be a natural consequence of the requirements documentation, scope of work, visualized designs and user interface screens, and test case verification. By now, the system has been thoroughly investigated, designed, analyzed and test verified by the procurement organization, and is ready to be released as a fully vetted RFP document that has organization-wide ownership and commitment.

These five phases describe how managers and developers can plan, analyze and design a fully vetted system to meet the stakeholder goals and end-user needs. The S-DIP incorporates a sixth stage entirely dedicated to producing implementation documentation, separate and apart from the documentation of the system itself. This is an original aspect of the S-DIP that focuses on robustness and detail currently lacking in the implementation stage of the RFP process. The underlying thinking here is that, even though the entire system itself (requirements and testing) has been fully vetted, without a complete and detailed approach for how it will be implemented, a project is still significantly exposed to risk of failure in execution. During our interviews of IT project managers and developers, we found that implementation is a particular subject matter area that is often routinely short changed or outright ignored during the RFP planning process. During our interviews, we were frequently told that implementation planning is often conflated into other segments of the RFP, or left out of the RFP documentation process completely.

S-DIP covers this additional aspect of RFP with the added phase of Implementation Readiness Review. The purpose of this phase is to develop the plans for transition and implementation of the new system. In this phase we find installation plans with timelines and team personnel identified, training plans for familiarizing end-users and stakeholders with the operation of the system, maintenance plans including upgrade and release schedules, and transition plans with timelines for shut down of the old system and cut over to the new system. User and administrator manuals are also developed during this phase.

Now, obviously, not all of these activities are completed during the RFP development. The goal here is for the procurement team to set aside place holders for these topics and begin initial discussions and thoughts in these areas, so that they are not completely abdicated to the vendor, post RFP release. Managers applying this phase of the S-DIP reported that they liked the fact that they were forced to carefully consider these topics and consequences during the RFP process and *before* the decision making is shifted to the consultant or vendor.

Implementing the Five Stage Evaluation Model

Managers reported that the application of the Five Stage Model enforced *evaluation* decisions at specified function points of the S-DIP and provided greater clarity at each step along the way in the RFP process. Developers reported that they particularly liked that *evaluation* was "built in, to each of the phases of development" rather than being a single phase, occurring at the end of the life cycle when little could be done about mistakes other than scrapping work and re-work.

Inclusion of the fifth stage of Evaluation as a central hub, interacting and influencing the traditional four SDLC stages marks a significant shift in the approach to software development. The new approach views evaluation as an intertwined element within the SDLC itself, rather than a later stage isolated gateway.

Applying S-DIP to Bridge the Gap

Since its introduction in 2013, the S-DIP framework has been initiated in five organizations, covering over one-hundred end-users and software managers. The responses from the field thus far have been overwhelmingly positive. There have been three main areas of feedback received. The first area has been specifically about requirements – teams that implemented the S-DIP reported that they did a much better job at developing requirements in two ways: completeness and prioritization. Teams reported that the concept of review and reversion forced them to slow down and reevaluate their requirements at several occasions along the process. Teams also reported that reversion allowed them to "take a second and sometimes a third look at their priorities in their choice of requirements."

The second area of feedback was about testing – teams that implemented the S-DIP reported that they had never thought about acceptance tests before. Teams reported that by "being forced to think through" how a specific requirement was going to be tested for successful delivery gave them better clarity about the requirement itself and an increased confidence in directing the development project.

The third area of feedback was about the use of IPO as a method for analysis. All teams reported that they had never used IPO before, but now they are using it in nearly every aspect of their lives, not just on their development projects. Teams reported that using IPO gave them rigorous and very detailed insight into "exactly what we wanted to get out of the process and what was going to be needed for the process to work."

Summary and Conclusion

This paper documents the system development framework S-DIP, Software Development and Implementation Plan, as a methodology to support a vetting process for stakeholder strategic goals and end-user needs. The S-DIP provides a procedural model to fully review and mature system requirements for a more robust and detailed RFP document – thereby closing the gap between the RFP as a document and the SDLC as a paradigm, for system and software design and development.

The S-DIP has been introduced in several organizations that have been undergoing system development projects for both: automation of manual systems and reengineering of already automated systems. Many of the project team managers whom have used the S-DIP in their development and production of an RFP have reported that it has significantly improved their vendor selection process and increased their internal confidence and locus of control in the management of the system development project itself.

In selected field interviews, managers have reported that they "developed better clarity of the what we were looking for" and "had a better focus going forward into the project for what we needed to achieve for the new system." Developers reported that they had a better understanding of what the customer wanted to accomplish and this led to designing better test plans for user acceptance testing.

Of course, there is still much more to learn about aligning organizational goals and end-user needs with system requirements, but like the *Tropos Project*, the *S-DIP Framework* presented in this paper is a methodology for practitioners to consider during their RFP development and documentation processes.

References

1. Andrea, J. (2003, June). An agile request for proposal (RFP) process. In Agile Development Conference, 2003. ADC 2003. Proceedings of the (pp. 152-161). IEEE.

2. Boehm, B., Turner, R. (2003). Balancing agility and discipline: A guide for the perplexed. Addison-Wesley Professional.

3. Castro, J., Kolp, M., Mylopoulos, J. (2001, January). A requirements-driven development methodology. In Advanced Information Systems Engineering (pp. 108-123). Springer Berlin Heidelberg.

4. Coonce, L. H. (1979). Use of Request for Proposals for Software Purchase Selection. CAUSE/EFFECT, 2(6), 32-34.

5. Dardenne, A., van Lamsweerde, A., Fickas, S., (1993). "Goal–directed Requirements Acquisition", Science of Computer Programming, 20, pp. 3-50.

6. Davy, D. (2011, October). Lessons from the past: What can be learned from ancient and modern rhetoric for a better RFP. In Professional Communication Conference (IPCC), 2011 IEEE International (pp. 1-15). IEEE.

7. DeMarco, T., Lister, T. (2013). Peopleware: productive projects and teams. Addison-Wesley.

8. Finkelstein, A., Spanoudakis, G., Ryan, M. (1996, March). Software package requirements and procurement. In Software Specification and Design, 1996., Proceedings of the 8th International Workshop on (pp. 141-145). IEEE.

9. Hyman, H. S. (2013). Systems Acquisitions, Integration, and Implementation: For Engineers and IT Professionals. Sentia Publishing, Texas, USA.

10.IEEE 12207-2008: Systems and Software Engineering – Software Life Cycle Processes, (1995, 2008)

11.Lehman, M. M. (2005, September). The role and impact of assumptions in software development, maintenance and evolution. In Software Evolvability, 2005. IEEE International Workshop on (pp. 3-14). IEEE.

12.McConnell, S. (2001). The nine deadly sins of project planning. IEEE Software, (5), 5-7.
MIL STD 1521b: Technical Reviews and Audits for Systems, Equipments, and Computer Software, (1986).

13.Mylopoulos, J., Castro, J., (2000). "Tropos: A Framework for Requirements-Driven Software
Development," Brinkkemper, J. and Solvberg, A. (eds.), Information Systems
Engineering: State of the Art and Research Themes, Springer-Verlag, June 2000, pp.261-273.

14.Natovich, J. (2003). Vendor related risks in IT development: A chronology of an outsourced project failure. Technology Analysis & Strategic Management, 15(4), 409-419.

15.Nelson, R. R. (2005). Project retrospectives: Evaluating project success, failure, and everything in between. MIS Quarterly Executive, 4(3), 361-372.

16.Porter-Roth, B. (2002). Request for proposal: A guide to effective RFP development (pp. 1-293). Boston: Addison-Wesley.
17.Saito, Y., Monden, A., Matsumoto, K. (2012, October). Evaluation of Non Functional Requirements in a Request for Proposal (RFP). Seventh International Conference on Software Process and Product Measurement. Joint Conference of the 22nd International Workshop (pp. 106-111). IEEE.

18.Wilson, D. G. (1993). Is honesty too much to ask for in RFPs? Mechanical Engineering, 115(11),