CHALLENGES IN REQUIREMENT ENGINEERING FOR SOFTWARE DEVELOPMENT AND COTS COMPONENT SELECTION

SARATH. P
Department of Information Technology, Sri Ramakrishna College of Arts & Science, Coimbatore, India.
23107116@srcas.ac.in

NANDHINI. C
Department of Information Technology, Sri Ramakrishna College of Arts & Science, Coimbatore, India.
nandhini.c@srcas.ac.in

DR. SRIDHAR. M
Department of Computer Technology, Sri Ramakrishna College of Arts & Science, Coimbatore, India.
sridhar@srcas.ac.in

***Abstract*—The requirement engineering provides the spine for any software development process and plays a vital role in the success of the whole process. This whole software structure is supported by four significant pillars, namely, the requirement engineering process. In this context, functional and non-functional requirements are the bricks and cements that provide strength to the architecture. Successive layers of design, implementation, and testing form the construction of the overall software system. The base established during the requirement engineering phase of the software development cycle, therefore, has to be strong to hold the whole structure of the software. Regarding this, requirement engineers are confronting a number of tasks for developing appropriate and effective solutions of software. This paper will review the challenges faced in requirement engineering, relating to the development of software applications and choices of appropriate COTS components. Such challenges include the interpretation of stakeholders' requirements, deriving workable products from incomplete or inconsistent process descriptions, verification and validation of requirements, large-scale data classification, modeling, and choice of COTS products with minimal modification to the original requirements. Along with pointing out these challenges, the paper also reviews critically the existing literature in order to explore the identification of the challenges by other researchers. More importantly, it proposes a model that identifies seven recurring challenges during the requirement engineering phase, which are grouped into certain problem areas. This is integrated with prior research in order to highlight the challenges which may not have been covered in earlier works. With these types of investment in engineering challenges of requirements, the engineers are able to provide a solid foundation for their software and thereby avoid failures in projects.**

Keywords---Requirement Engineering, Customer-off- the-shelf (COTS), Multi-site software development.

I. INTRODUCTION

Software requirements describe the services that a system provides and, hence, reflect a stakeholder's need. They originate from the real world in operations in the application domain. All activities regarding collection, analysis, specification, validation, and maintenance of requirements are collectively called Requirement Engineering, RE. The aim with RE is to reach end-user satisfaction considering optimal cost and time boundaries.

In this way, this requirement elicitation phase identifies problems that are happening in the existing system. However, most of these errors are not usually detected during the development process but come into view at the operational stage when the system fails to meet the expectations of all stakeholders [14]. Various researchers say that correcting such errors at the elicitation stage is far cheaper than in later stages of software development [14]. Hence, software applications' success is determined by the elicitation of requirements. When the requirement engineers consult different stakeholders to elicit requirements, they face various problems, which are later compiled into recognized challenges. This early problem identification will enable the engineers to take proper precautionary measures so that these projects do not go off track.

The challenge of managing and modelling unstructured elicited requirements from the operational domain is another problem. Requirements should be well-defined and well-structured based on some standards of specification templates. This will make it easier for the stakeholders and the maintenance team to understand the requirements much better. Well-defined requirements can also be easily validated by the stakeholders. Poorly defined requirements are ambiguous and could not be measured, which is one of the reasons for software failure.

System requirements describe exactly what the software is supposed to achieve. They are classified into functional requirements that deal with system functionalities and non-functional requirements that deal with software constraints. Both types depend on one another and are equally important to satisfy. However, decomposing, refining, and validating these requirements presents huge challenges to the requirement engineers.

Moreover, many software applications are built with reusable components to reduce development cost and time. With this in mind, the selection of COTS components will be a major challenge whereby the engineer has to match needs of stakeholders with the product being offered by COTS vendors [15]. This imposes more complexity in the requirements engineering process. Selection of COTS components normally relies on subjective judgment; hence, vendors take every advantage of this by releasing new versions of components, which places engineers back at the modifying desk to adjust the original requirements to what was available in the market. Secondly, the COTS vendors often provide minimal specifications relative to internal architecture and detailed description of their products, leaving the engineers with very limited information to verify if the integrated component would meet the user's requirements. Apart from this, some of the COTS components have never been tested in a real-world scenario .

Previous researches had selected one particular domain and focused on requirement engineering challenges associated with that particular domain. However, in this paper, challenges related to multiple domains have been presented and integrated in one document. The background study has been divided into four quadrants, namely, requirement engineering processes, system requirements, applications and products. Each quadrant is further divided into sub-sections, and every section has focused on specific problems and challenges. This paper presents a review of the related literature with its critical evaluation, whereby it has introduced a framework that indicates the challenges of requirement engineering that have not been covered by previous studies.

The following framework identifies seven relevant challenges. These are grouped in problem areas. The challenges include technological and economic crises, external events, requirement engineering process difficulties, organizational issues, stakeholder conflicts, and time constraints. These factors have been linked to the quadrants of the background study to present a broader view concerning the general challenges in RE.

The rest of this paper is organized as follows. Section 2 reviews related prior work in key areas. Section 3 discusses critically the challenges identified from previous studies. Section 4 discusses in detail the framework and its components. Finally, conclusions and possible future work are presented in Section 5, while Section 6 lists references.

II. BACKGROUND STUDY

Several researchers have identified a number of challenges on various domains of requirement engineering. Most of the previous studies have addressed issues within specific areas such as requirement elicitation and analysis challenges or selection of COTS components. However, in this section, challenges from various research works are integrated into one, facilitating an overview of the background research into four quadrants comprising the major challenges of the requirement engineering as highlighted in Figure 1 below. The background study has taken into consideration four major areas: the process of requirement engineering, system requirements, applications, and product. For each of these areas, further subcategories provide more detail. Requirement Engineering Process: Included in this quadrant are the critical stages of requirement elicitation, specification, and validation.

System Requirements: The second quadrant is related to functional and non-functional requirements, which are key issues in the definition of the systems' behavior and constraints. Applications: This category brackets enterprise application-related challenges and multi-site software development, where the emphasis is on complexity and coordination in very large projects. Products: The last quadrant deals with the problems associated with the usage of COTS components, more precisely the prebuilt products to ensure that they meet the particular requirements of the system to be developed.

This categorization enables a more focused investigation into the diversity of issues in the area of requirement engineering by relating many research areas together into one integrated view.

A. Requirement Engineering Process

The key activities of the Requirement Engineering Process are requirements elicitation, requirement specification, and requirement validation. These three activities form the foundation of requirement engineering and involve how the needs within the software systems are collected, documented, and checked.

1) Requirement Elicitation and Analysis

Requirements elicitation is intrinsically a difficult job, as stakeholder needs are often complex and variable in nature. According to Goldin and Finkelstein, the understanding of a stakeholder's needs and the handling of requirements unexpected growth over time is always very problematic. Often, the information extracted during the elicitation stage is inconsistent or conflicting in nature, as it is brought out from different sources with different perspectives. In the presence of these challenges, Abstraction-Based Requirement Management-or AbstRM- is among those methods that could reduce the problems with requirement elicitation. The AbstRM method makes this process easier by abstracting key concepts from raw requirements into higher-level categories that are easy to understand. The supporting tool, called AbstFinder [20], has identified critical abstractions, or "abstraction identifiers," which fall into such categories as agents, entities, actions, and goals. These identifiers are then mapped onto an abstraction network, allowing the elicitor to visualize and trace relationships between salient concepts.

However, efficiency aside, there are a few limitations: for example, AbstRM does need extensive manual processing in fine-tuning the abstraction identifiers besides the fact that it may misinterpret other linguistic features such as verbs and nouns. For example, a term such as "book a flight" might incorrectly be classified by the tool; it should be "book" as a verb and not as a noun. Therefore, human elicitors necessarily need to cross-check these terms to be sure about their accuracy ın [20]. Besides, the issue of the integration between various requirement management tools is also problematic, since most tools do not explicitly state their compatibility features and require deep analysis to make sure smooth integration is successfully achieved [21,22]. More specifically, the cost of the software development life cycle aggravates the intricacy in the requirements engineering process.

2) Requirement Specification

Fire smith gives a good account of the problems associated with requirement specification, particularly in the traditional waterfall development cycles that rely greatly on heavy manual documentation. Such specifications, because they are manual, are most likely to be incomplete or vague and hence often include major headaches related to configurations and requirement management. The specification documents themselves become costly and cumbersome for distribution across various stakeholders in manual systems as well. The paradigm shift from traditional document-centric requirement engineering to iterative requirement engineering has helped solve some of the aforementioned problems. Iterative methods nudge refinements and validations of the requirements throughout the development process in small steps. As outlined, this helps to locate and correct defects early as shown. This does take time and can be more laboured for projects with hundreds of requirements that will need to be constantly re-elaborated and verified.

In enhancing the development of specifications, researchers recommend the use of requirement models such as use cases that logically organize specifications, hence allowing clearer traceability. Specifications can be stored in object-oriented databases or extended relational databases from centralized repositories for faster access and verification. Requirement templates and modern requirements engineering tools also contribute to more accurate software requirement specifications with minimal likelihood of omission or errors. However, even with the advantages, large volumes of requirements are still a challenge to handle in these repositories. The requirement to revise terabytes of requirements manually whenever the stakeholders change their needs is an ongoing burden to the requirement engineers. The risk of introducing inconsistencies or errors during this process underlines some limitations of current specification methods.

3) Requirement Validation

The process of requirement engineering also includes requirement specification validation, which is assigned the task of ensuring correctness and completeness. According to Sequeda, ambiguous or incomplete requirements may lead to serious problems during software development. Thus, selecting appropriate verification and validation

techniques becomes of great importance for the requirement engineers, who have to be certain that system specifications meet the expectations of the stakeholders. To address these issues, Sequeda has developed a taxonomy of requirement specifications: executable and non-executable specifications. Executable specifications are written using declarative languages such as Java Modeling Language and can be verified through the development of prototypes, while there is little chance to validate the non-executable ones that are usually written in natural language. One possible solution to validate non- executable specifications is the ERM tool. This tool converts the requirement documents into XML format and makes use of XSLT for its validation. While these techniques reduce the pain associated with the validation process, they also require a very high amount of proficiency in domain knowledge as well as specialized tools like ERM and XSLT. Moreover, prototype-based development for the purpose of validation can only partly guarantee the realization of non-functional requirements. Hence, validation remains one of the most cumbersome and time-consuming stages of the whole requirement engineering process.

B. System Requirements

System requirements may be broadly categorized into two types: functional and non-functional requirements. Each category poses its peculiar problems in both elicitation and management.

1) Functional Requirements

Functional requirements identify what the software system must undertake in terms of actions or functions. Their elicitation and documentation are usually problematic because analysts may perceive different needs from the stakeholders due to the conflicting interpretation given to them. For instance, some functional requirements can be similar, even conflicting, it is vague what to expect from the system. Functional requirements have to be categorized and refined, according to researchers, to remove redundancy and hence ensure consistency. The functional requirements must be kept synchronized and consistent across the different teams that take part in software development. Early confirmation of functional requirements with the stakeholders is secondly required to avoid complications later in the project lifecycle.

Formal methods have thus been advanced to formalize and manage functional requirements. These formal methods are based on mathematical techniques such as set theory and logic. These methods provide short, but rigorous specifications that are verifiable according to rules predefined earlier. While Formal Methods reduce ambiguity and clarify precision, they are significantly complex and costly to apply and also require specialized expertise 56. Additionally, manual tracing of the requirements becomes too complicated to manage with no supporting formal tools and mechanisms for assistance in the process..

2) Non-Functional Requirements

NFRs define the quality attributes of a system, such as its performance, reliability, and scalability. As stated by Thomas, "non-functional requirements are typically poorly specified by the stakeholders or require considerable refinement.". For example, NFRs of the form "the system shall be robust" are usually imprecise and hard to quantify. Consequently, they are often neglected in favor of functional requirements, which are more straightforward to define and validate. However, both functional and non-functional requirements are essential for the success of a software system [4]. Due to the increasing complexity of contemporary software systems, NFRs have gained more and more importance. NFRs must be precisely captured and validated for instance, system availability and response times - in order to achieve high-quality standards [1,3]. These challenges have been addressed by several researchers using approaches such as the LEL methodology, which supports the elicitation of non-functional requirements through the decomposition of domain-specific terms and their behavioral implications [5,6]. Non-functional requirements can also be validated using abstract interpretation-based static analysis based on source program analysis and selection of abstract domains [2].

Despite these advances, the elicitation and validation of non-functional requirements remain a challenging task. It will require knowledgeable stakeholders and sophisticated tools to overcome this bottleneck.

3) Product: Commercial Off-The-Shelf (COTS)

Selection of COTS products introduces added difficulties for the requirement engineers. As Alves explains, because COTS products have their own characteristics, it is difficult for them to meet the specific needs of any organization. This entails the fact that the organizations have to compromise on their requirements with respect to the features available within the COTS product, which

may lead to certain trade-offs influencing their complete business strategy.

Alves comes up with a goal-oriented approach; in this, the objectives of the system are identified first, followed by the assessment of the suitable COTS products against those goals. Selection is done when the features of the product match the objectives of the system. However, the paper recognizes limitations in this approach. It does not fully address the issue of the relationship between COTS features and the supporting technology, thus leading to various problems at the integration stage of the product. For instance, choosing a product that uses a client-server architecture for an organization with a distributed system could bring major incompatibility issues. Conversely, adopting COTS may also entail greater risks as an organization will have to adapt its business process to suit a particular COTS product.

In such cases, decision-makers can utilize fuzzy logics approaches in identifying the appropriate COTS components from various criteria such as functionality, performance, and reusability. Thus quite a non-crisp mode of decision-making is possible.

C. Applications

This section covers requirement engineering challenges for enterprise applications and multi-site software development.

1) Enterprise Applications

Enterprise applications are a unique type of application that have their own particular challenges in understanding the application domain, with specific business processes. Salim [7] explains that the classification of a huge amount of data and dealing with incomplete or scant information by stakeholders complicates the requirement engineering process in these large-scale systems. Requirements are usually poorly documented, which creates challenges for the validation and modifications that are taking place [8,9]. SMEs have issues related to resource and technical capability constraints arising specifically for SMEs. These stakeholders may not have experience or exposure to articulating business processes in a manner that requirement can be effectively captured, thereby resulting in massive communicational gaps.

Business-IT alignment is considered imperative within enterprise contexts; however, this may hardly be achieved. Poor communication exists between departments due to different priorities and different levels of understanding.

In this kind of environment, it is the requirement engineers' responsibility to ensure business objectives are properly represented in the technical specifications of an enterprise application. Techniques like Business Process Model and Notation and the Enterprise Architecture frameworks have been proposed that can alleviate these problems through systematic documentation of business process and mapping with the system functionality..

2) Multi-Site Software Development

With the development of distributed software, requirement engineering becomes increasingly complex, especially in geographically dispersed teams. According to Berenbach, multi-site development creates delays and miscommunication problems as teams from different regions might follow different practices or adhere to a different timeline. It is difficult to plan and synchronize the requirement activities across time zones and languages. This can result in incomplete or conflicting requirements.

In order to manage these challenges, cross-reviewing techniques may be used in which the requirements gathered by one team are reviewed and validated by another. This technique contributes to spotting inconsistencies early and has a set of consistent requirements across all teams. Additionally, assigning a requirements facilitator who is actually in charge of coordination and communication between teams may reduce some risks of distributed requirements engineering.

III. CHALLENGES IN REQUIREMENTS ENGINEERING

Requirements engineering is among the most crucial activities within the lifecycle of software development. Most of the defects in requirements go unnoticed during the development process but tend to appear when the system starts operating and the users are dissatisfied due to unmet expectations. The ill-defined requirements take lots of rework on redesigning, re-implementing, and re-testing hence increased cost and time spent. Thereby, these requirements engineers face many difficulties in rendering efficient and effective software systems. Most of such challenges need to be identified from an early stage for improving the success rates of the software projects.

Only a few works have addressed these particular challenges in the area of RE within various domains. These few research works proposed several models and made some recommendations to handle them. However,

most of them present partial views of RE, not holistic ones. In this respect, we present a more comprehensive RE challenge framework that is depicted in Figure 3. The model extends the challenges identified in Figure 2 and discussed in Section 3 by placing most of the issues that regularly arise at development time as well as during the selection of COTS components within a wider context.

Instead, the proposed framework integrates the RE challenges across various domains into one rather than limiting it to a particular area. In this way, RE challenges are more organized; hence, the engineers can easily identify and get ready for specific challenges. The model also utilizes previous studies to indicate challenges that had not been identified before. Seven components make up the major RE challenges:

- Technological challenges
- Economic constraints
- External events
- The requirements engineering process itself
- Organizational issues
- Stakeholder conflicts
- Time management

A. Technological Challenges

Outdated RE tools impede proper functionality, particularly in prototype or simulation developments. Introducing newer tools generally creates incompatibility problems, such as an inability to convert the current file formats. Besides, when trying to integrate the features of various RE tools, developers must perform deep structural and functional analysis, which is challenging and requires much time.

Other challenges involve the procurement of the COTS components. Selection of the COTS is most often subjective whereby the selected products may not meet the full needs of the customers. In addition, configurations of the COTS may influence their selection because upgraded versions of a product may lack some substantial features which were under evaluation. Incorrect selection of appropriate components leads to software which does not meet user requirements.

B. Budgetary Limitations

The IT market is characterized by rapidly changing technologies and budgetary constraints. Ill-defined requirements might result in higher maintenance costs. Economic uncertainty for either the customer or the developing organization may add to these woes. Unexpected expenses, budget overflows, and economic downtrends result in postponed projects or even cancellation. Besides, some economic factors such as depreciation, taxes, and changes in the stock exchange rate may hinder effective COTS product selection within the available budget.

C. External Events

Software development may be badly affected by the external factors if not foreseen. Accidental data loss, file corruption, viruses, or hardware failure may cause major setbacks. Natural disasters, fire, or even terrorist attacks might affect the regular process of RE and may result in huge losses in terms of finance and delays.

D. Requirements Engineering Process

The main objective of RE is to identify what is to be done and specify the associated constraints and bounds. But all this is easier said than done when dealing with complex or critical systems. The stakeholders may poorly express their needs and requirements remain vague and ambiguous. Additionally, various specifications of requirements give solutions and not focus on the problems; that may result in poor designs for software. Software requirements validation is very crucial for project success, but sometimes it produces flawed prototypes, which further delays the project and may affect the overall project timeline.

E. Organisational Issues

Software development involves the alignment between business strategies and IT solutions. Unfortunately, organizational departments face different perspectives that make this alignment challenging to realize. Further, organizations are continuously performing business process re-engineering to achieve efficiency, which dramatically changes software requirements. Dealing with these volatile and changing requirements is a major challenge for a requirement engineering professional.

F. Stakeholder Conflicts

The success of a software project is heavily dependent on the expertise and cooperation of the stakeholders. Poor technical skills and domain knowledge of the requirements engineers may result in poorly defined requirements that do not meet users' needs. Conflicts among stakeholders, ambiguity in roles, or the resignation

of important team members further complicate the RE process. Recruitment and training replacements is often not feasible within constraints of time and budget, further adding to the challenges of delivering successful software.

G. Time Management

Scheduling and estimation are considered important tasks in software development, but the RE activities have been most of the time underestimated regarding the required time and hence caused delays in attaining project milestones. Underestimating the importance of these activities due to available time, the engineers may use short cuts or even omit the main stages of the process, thereby poorly defining the requirements, which will result in software failure.

IV. CONCLUSION

The paper addresses some critical challenges in requirement engineering: stakeholder needs, incomplete process descriptions, verification, and validation of requirements. It focuses on the effective selection of Commercial Off-The-Shelf (COTS) products with minimal modification of requirements. We identified the problems related to the requirement engineering process, system requirements, applications, and product-related issues through a critical review of the literature. These were further broken down into specific problems, showing the facets of requirement engineering.

Various techniques proposed in past research were critically reviewed and compared for their effectiveness and applicability to different domains. We synthesize these techniques and provide a framework that encompasses the documented as well as the new challenges evident in the field. The framework recognizes seven major challenges: technological crises, economic crises, external events, process difficulties, organizational issues, stakeholder conflicts, and time management. All these challenges are further elaborated through their sub- issues to present a comprehensive idea of the challenges evident in the requirements engineering. It is an overall tool for requirement engineers, indicating common problems and providing suggestions where improvements should be made. Such challenges linked to the wider perspective of requirement engineering give a wide view into the field and point out continued research and adaptation to meet the emerging challenge.

What follows is that such multifaceted challenges have to be put right, as identified in this paper, for improving the success rate of software projects. Moreover, with the management of stakeholder expectations, comprehensive requirement validation, and strategic selection of COTS products, project objectives and qualitative software solutions would be achievable

REFERENCES

[1] G. Maria C. de and J. Brelaz de, “Improving the Separation of Non-Functional Concerns in Requirements Artifacts,” in Proceedings of the 12th IEEE International Conference on Requirements Engineering (RE 2004), Kyoto, Japan, 6-10 September 2004, 2004.

[2] A. Cortesi and F. Logozzo, “Abstract Interpretation- Based Verification of Non-functional Requirements,” Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 3454, pp. 54-59, 2005.

[3] IEEE, IEEE Recommended Practice for Software Requirement Specification, 1988.

[4] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, “Non-Functional Requirements in Software Engineering,” Springer Berlin / Heidelberg, pp. 363-379, 2009.

[5] L. M. Cysneiros and E. Yu, “Perspectives on Software Engineering,” in J. C. Sampaio do Prado Leite and J. H. Doorn, Eds., Kluwer Academic Publishers, pp. 114-138, 2004.

[6] L. Marcio and J. C. S. do Prado, “Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation,” in Anais do WER01 - Workshop em Engenharia de Requisitos, Buenos Aires, Argentina, November 22-23, 2001.

[7] J. Salim, “Requirement Engineering for Enterprise Application Development: Seven Challenges in Higher Education Environment,” World Academy of Science, Engineering and Technology, vol. 4, pp. 101, 2005.

[8] H. d. Vries, H. Verheul, and H. Willemse, “Stakeholder Identification in IT Standardization Processes,” in Standard Making: A Critical Research Frontier for Information Systems MISQ Special Issue Workshop, 2003.

[9] A. T. Bahill and S. J. Henderson, “Requirements Development, Verification and Validation Exhibited in Famous Failures,” Systems Engineering, Wiley Periodicals, Inc., vol. 8, no. 1, pp. 1-14, 2005.

[10] F. T. Sheldon and H. Y. Kim, "Validation of Guidance Control Software Requirements Specification for Reliability and Fault-Tolerance," in IEEE Annual Proceedings on Reliability and Maintainability Symposium, Washington, DC, USA, 2002.

[11] M. Oktay, A. B. Gülbağcı, and M. Sarıöz, "Architectural, Technological and Performance Issues in Enterprise Applications," World Academy of Science, Engineering and Technology, vol. 27, pp. 224-230, 2007.

[12] C. Alves, J. B. P. Filho, and J. Castro, "Analyzing the Tradeoffs Among Requirements, Architectures, and COTS Components,pp.23-26, 2001. [Online]. Available: http://wer.inf.pucrio.br/WERpapers/artigos/artigos_WER 01/alves.pdf [Accessed: Nov. 2008].

[13] C. Alves and A. Finkelstein, "Investigating Conflicts in COTS Decision-Making," International Journal of Software Engineering and Knowledge Engineering, vol. 13, no. 3, pp. 1-21, 2003.

[14] B. Thomas, "Meeting the Challenges of Requirement Engineering," News at Software Engineering Institute, 2009.[Online].Available: http://www.sei.cmu.edu/library/abstracts/news-at-sei/spotlightmar99pdf.cfm [Accessed: Jan. 2010].

[15] C. Alves and A. Finkelstein, "Challenges in COTS Decision-Making: A Goal Driven Requirements Engineering Perspective," in Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, 2002.

[16] C. J. Fidge and A. M. Lister, "The Challenges of Non- Functional Computing Requirements," pp. 6-7. [Online]. Available: http://sky.fit.qut.edu.au/~fidgec/Publications/fidge93c.pdf [Accessed: Nov. 2008].

[17] L. Goldin and A. Finkelstein, "Abstraction-Based Requirements Management," in Proceedings of the International Workshop on Role of Abstraction in Software Engineering, Shanghai, China, 2006.

[18] A. V. Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective," in Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, 2000.

[19] Endava, "White Paper on Requirements Gathering and Analysis," pp.710,2007.[Online].Available:http://www.endava.com/resources/Endava.com-WhitePaper-RequirementsGathering.pdf [Accessed: Oct. 2008].

[20] L. Goldin and D. Berry, "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirement Elicitation," in IEEE Requirements Engineering, Proceedings of the First International Conference, 1994.

[21] Volere[Online].Available: http://www.volere.co.uk/tools.htm [Accessed: Nov. 2008].

[22] Requirement Tools,[Online].Available: http://easyweb.easynet.co.uk/~iany/other/vendors.htm [Accessed: Nov. 2008].

[23] L. K. Meisenbacher, "The Challenges of Tool Integration for Requirements Engineering," in Proceedings of SREP'05, Paris, France, 2005.

[24] B. Berenbach, "Impact of Organizational Structure on Distributed Requirements Engineering Processes: Lessons Learned," in Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner, Shanghai, China, 2006.

[25] S. Timea, H. Andrea, and P. Barbara, "The Challenges of Distributed Software Engineering and Requirements Engineering: Results of an Online Survey," pp. 9-13. [Online].Available:http://www-swe.informatik.uniheidelberg.de/research/publications/TR_Distributed_RE_Version1.pdf [Accessed: Oct. 2008].