SOFTWARE REQUIREMENTS ENGINEERING: STATE OF THE ART AND RESEARCH TRENDS

VASSILIS C. GEROGIANNIS

Business Administration Department, Technological Education Institute of Thessaly, Larissa, Greece
E-mail: gerogian@teilar.gr

Abstract- The success of software projects highly depends on the level that their outcomes (i.e., the developed software products/services) satisfy the needs of all stakeholders and software requirements encompass these needs. In this paper, a state of the art survey is presented in the area of Requirements Engineering (RE). The paper presents the problems solved through the RE processes, summarizes the current research and industry status in the field and presents a classification scheme of various RE methodologies. The paper concludes by discussing the current research/industry trends in the field of RE.

Keywords- Software Projects, Requirements Engineering, State of the Art.

I. INTRODUCTION

The success of a software project highly depends on the degree that the project outcome (i.e., the developed software system/service) satisfies the needs of its users and software requirements encompass these needs [1]. Requirements Engineering (RE) is the process that emphasizes on determining/analyzing requirements and involves [2, 3]: (i) discovering the purpose for which a software system has to be developed, (ii) identifying system stakeholders and their needs, (iii) understanding the environmental context in which the system will be deployed and used, (iv) negotiating with project stakeholders, (v) prioritizing requirements artifacts, (vi) modeling and analyzing requirements, (vii) documenting requirements in a form that is suitable for communication, implementation and testing, (viii) validating that the documented requirements are in line with the identified stakeholders' needs, (ix) verifying that requirements are adherent to quality attributes, and x) managing requirements as they evolve and change. In this paper, a brief state of the art survey is presented in the area of RE. The paper presents the problems solved through the RE process, current research and industry status, a classification of various RE methodologies and tools as well as current RE practices. The paper concludes by discussing the current research/industry trends in the field of RE.

II. STATE OF THE ART IN REQUIREMENTS ENGINEERING

2.1. Problems Solved with Requirements Engineering
The "requirements mess" problem [4] is a continuous challenge in Software Engineering (SE) research and practice. Although there are various perceptions on defining the success/failure of a software project [5, 6], most academic and industrial research reports demonstrate (see, for example, [7, 8, 9, 10]) that among the leading sources for a software system success or failure there are factors related to inadequate consideration of the RE process. Examples of these factors are: (i) misunderstanding/lack of user input & involvement, (ii) incomplete/rapidly changing and highly volatile requirements, and (iii) extremely complex relationships between software systems of today and their technological/organizational environments in the context of which software systems operate. By following a conventional classification scheme for the various tasks comprising a RE process [3, 11], we can realize some conventional problems addressed by RE: (i) Requirements elicitation activities enable understanding and identification of organizational/stakeholder objectives and needs. (ii) Requirements analysis and specification tasks transform the elicited requirements in terms of a representational notation by using either informal or formal models. (iii) Requirements validation tries to ensure that requirements reflect stakeholders' needs, while verification examines the degree that requirements are error-free and conform to quality criteria, such as consistency, feasibility, traceability and lack of ambiguity. (iv) Requirements management focuses on identifying requirements' changes over time and across different software product families, establishing and documenting traceability links, determining maturity/stability of elicited requirements, performing cost-benefit analysis for requirements' changes, enhancing maintainability, prioritizing often a large number of requirements, which may be distributed geographically, functionally and organizationally.

2.2. Status of Research in Requirements Engineering
RE engineering research and practice are active for more than 40 years, since practitioners in early 1970's noticed the benefits of adopting a systematic consideration of requirements in a software development project. Despite the significant progress in the relevant body of research (improved software tools, modeling methods and process technologies for

RE), there are always difficulties in applying efficiently/effectively a RE process.
The economics of RE is always an important success factor, since the cost of resolving requirements errors and handling requirements volatility is growing exceedingly, as software life-cycle evolves [12]. The main source for RE difficulties arises from the distinctive "nature" of RE compared to other SE processes and activities [3]. RE focuses on the (large) problem space area, by defining the problem that is to be solved by a software system, while other SE processes emphasize on the (more limited) solution space area, by defining, implementing and testing the behavior of a software system.

RE starts with analyzing an ambiguous problem description and defining the system environmental conditions, and must, finally, result in a concrete representation of requirements (i.e., solution oriented requirements) that is amenable to implementation. However, the conditions in the environment of a software system can be various, complex and unconstrained and thus, the complexity of requirements tasks is getting higher. The intrinsic limitation of human information processing and obstacles in interaction/mutual understanding between designers and multiple stakeholders also make RE an inherently difficult process. RE should emphasize on reconciling the relationships between non-technical staff (customers/users) and technical staff (analysts/designers/developers/testers), gaining involvement from all stakeholders and providing results (requirements artifacts) which are understandable and useful to all. All the above inherent characteristics of the RE process develop a constant interest in continuous re-think, re-align and re-evaluate RE practices to cope with practical challenges and emergent demands of today's software intensive systems.

2.3. Status of Requirements Engineering Adoption and Requirements Engineering Maturity
A number of RE methodologies, techniques and tools have gained a wide adoption in practice [13]. Representative examples are:
(i) requirements elicitation techniques like focus groups, stakeholder interviews, exploratory prototyping and design thinking, (ii) business process and system modeling notations, structured requirements and object-oriented requirements documentation by using the notations of the Unified Modelling Language - UML (especially use case diagrams) and utilizing CASE tools, (iii) traceability analysis techniques for examining requirements evolvements throughout the software development life-cycle, (iv) RE process quality assessment and improvement methodologies by adopting quality frameworks such as CMMI, (v) RE for systems based on service-oriented architectures, (vi) outsourcing of RE activities etc.

Yet, research and practice in RE cannot be considered as mature, since new challenges appear as new needs from software intensive systems are emerging. The interest in RE advances is growing up, mainly due to the evolving complexity, heterogeneity and uncertainty of the technological/organizational environments where the software is developed or utilized [14].
Requirements of contemporary software systems are shaped by the rapid change in technological capabilities, new application demands in rapidly evolving environments and the need of developing systems for maximum value creation. The success of RE process (followed in a RE project) is also more dependent on complex organizational issues such as the politics of resource allocation and the legitimacy of human decision making [15]. Principles of information hiding and data abstraction are no longer enough to handle the "requirements mess" problem. New principles, multidisciplinary research strategies and key issues are required to be investigated and adopted for handling the new shift of the field's focus, that is, the transition from engineering isolated systems and components to generating and adapting IT-supported ecologies of systems [14].

In a strict sense, there are no competing alternatives to RE. Even in iterative and agile software projects which do not emphasize on formal documentation (e.g., in agile software development processes), solutions evolve through feature-driven prioritization and development and communication/collaboration between self-organizing/cross-functional teams of stakeholders [16]. RE is a multidisciplinary domain that merges many technological approaches to suggest the theoretical foundation as well as practical methods for eliciting, modeling, analyzing and managing requirements [1]. The background of RE draws not only upon software/systems engineering, but also upon theoretical/practical computer science, logic, cognitive psychology, anthropology, sociology, linguistics and philosophy (epistemology, phenomenology, ontology).

III. CLASSIFICATION OF REQUIREMENTS ENGINEERING TECHNIQUES

Recent state-of-the-art reviews in RE are performed by classifying the activities of a RE process into the following phases, which are executed not sequentially but rather iteratively and progressively [3, 11]:

(i) requirements elicitation and discovery, (ii) requirements modeling, analysis and specification, (iii) requirements validation and verification, and (iv) requirements management. Table 1 (mainly adopted from [3] and enhanced with findings from [1] and [11]), these phases are considered to classify various RE methodologies and techniques that have been proposed in the relevant literature.

Table 1: Requirements Engineering Methodologies & Techniques

RE Phase/Activities	Methodologies / Techniques
Discovery & Elicitation	• Techniques for stakeholders identification, • Metaphorical techniques (Metaphors, Personas), • Techniques for inventing requirements (questionnaires, surveys, brainstorming, interviews, focus groups, consensus-building workshops, e.g., RAD/JAD workshops), • Direct observation, • Cognitive techniques (e.g., Protocol analysis, Laddering, Card sorting, Repertory grids), • Contextual Techniques (e.g., Ethnography, Ethnomethodology, Conversation analysis), • Exploratory prototyping, • Models animation & simulation, • Invariants generation, • Storyboards, • Exploratory models (Use cases & Scenarios, e.g., CREWS), • Enterprise models, • Policy & Goal models, • Goal-based methods (e.g., KAOS and i*), • Agents, • Elicitation of security requirements (Anti-models, Trust assumptions, Threat models), • Elicitation of non-functional requirements.
Modeling	• Formal models, • Natural language representations, • RE Reference models, • Viewpoints, • Patterns, • Modeling facilitators, • Formalization heuristics, • Models merging/synthesis/composition, • Enterprise modeling (business goal, rule & workflow models), • Behavioral & functional models, • Structured analysis & design, • Object-oriented models, • Data models (ERD, Class/Object diagrams), • Domain models & descriptions, • Property languages, • Notation semantics, • Goal-based models, • Agent models, • Anti-models, • Threat models, • Non-functional requirements models
Analysis	• Negotiation, • Requirements analysis with respect to Components Off The Self (COTS), • Components adaptation and reuse, • Conflict management & analysis, • Linguistic analysis, • Ontologies, • Checklists, • Inspections, • Obstacle analysis, • Risk management, • Impact analysis, • Causal order analysis, • Requirements selection techniques (from multiple viewpoints), • Requirements prioritization, • Variability analysis, • Automated reasoning (e.g., analogy & case-based reasoning), • Consistency checking.

Further to the above techniques, there are also valuable research and practice results in RE software process improvement (SPI) methods which can be classified into two broad categories: inductive (or problem-based) [17] and prescriptive (or model-based) [18, 19]. The former assess process improvement upon the strengths/weaknesses of existing processes and consider as knowledge sources the stakeholders' perceptions, while the latter, and more dominant in the SPI area, center process improvement upon best practices within industry.

IV. CURRENT PRACTICES IN REQUIREMENTS ENGINEERING

The current status of practice in the practical use of RE techniques is rather limited, compared to the extensive relevant research work in the area. Field

surveys and empirical studies (e.g. [11]) report that there is a persistent divergence between research and practice, as practitioners "slowly" adopt requirements methodologies/techniques proposed by researchers often "ignore" practices and actual needs of requirement engineers/designers. In particular, a field survey, involving 39 RE professionals from Fortune 500 companies [11], demonstrates the wide acceptance of a rather limited subset of requirements practices, such as: (i) risk mitigation and control, (ii) use of semi-formal modelling techniques (e.g. use of UML, particularly UML use case diagrams, and supporting CASE tools), (iii) requirements elicitation, through focus group discussions and cross-disciplinary stakeholder teams, and (iv) requirements validation, through prototyping, mock-ups and system walkthroughs.

Traceability analysis and software process improvement (e.g., CMMI [18], ISO/IEC 15504 – SPICE [19] or "lightweight" process improvement models [17]), although not widely accepted, constitute directions of interest for many companies, especially for software Small and Medium Sized Enterprises (SW SMEs). Effective balancing the challenges of specificity, comparability and accuracy in a process assessment/improvement project, by considering also resources' constraints, is another practical problem, especially for SW SMEs interested in investing resources at an SPI project [20]. Furthermore, Service-Oriented-Architectures (SOA) and outsourcing of development are influencing requirements practices in many companies, while issues like the systematic treatment of non-functional requirements, use of formal methods and methodical handling of conflicts among stakeholders, are not among the commonly applied practices.

V. CONCLUSIONS: CURRENT RESEARCH AND INDUSTRY TRENDS

Designing and implementing an effective/efficient RE processes is always a challenging problem. Many new challenges in RE have arisen in the recent years, while the transfer of new practice problems to research has been less successful. Therefore, the R&D community needs to be proactive in identifying RE research problems and solutions which will leverage successful and practical results. Representative key research areas which have been reported in the relevant literature to be of interest in the future are the following [3, 11, 13]:

(i) Requirements analysis techniques for large-scale and ultra-large-scale systems. Systems scaling is no longer related only with system size parameters, but also with complexity, heterogeneity, uncertainty, environment variability, number of interacting devices and interfaces, number of distributed decision/computation making nodes, etc. In the close future we will require more systematic RE methodologies and rigorous techniques to cope with the complexity of large-scale systems (e.g., next generation command & control systems, future intelligent transportation systems, systems managing power grids, systems controlling telecom infrastructures, integrated health-care systems, systems which are integrated with legacy infrastructures etc.). Potential problems are the diversity of requirements and stakeholders' concerns, the need to integrate multiple disciplines (e.g., sensors communication, intelligent scheduling & control), new abstraction/refinement paradigms and conflict management techniques for large-scale systems.

(ii) Security RE for pervasive & mobile systems and flexible tolerance in critical applications. Since systems become more mobile, pervasive and ubiquitous, they are more potential to be targets of exceedingly emerging security threats/attacks. Thus, additional research efforts are required towards low-level security RE (e.g. identify/avoid vulnerabilities, protect information, specify counter measures to handle attacks and recover from attacks), but also towards structuring and reasoning on high-level security requirements (e.g., security requirements for organizational structures and security policies) and examining the intertwining between low-level and high-level security. The complexity of future critical systems (e.g., software systems for transportation vehicles, medical applications and command & control systems) increases the need to focus more on tolerance requirements of these systems in order to present more adaptable behavior to dynamically changed environments. Issues of specifying levels of acceptable behavior in fault tolerant systems, relaxed correctness requirements as well as diagnostic and recovery mechanisms are expected to be further investigated.

(iii) RE techniques for highly integrated systems. The trend of developing integrating applications rather than developing new ones is expected to be continued. There is a variety of new generation highly integrated systems in which computing is tightly interleaved with environmental monitoring and control. Examples are intelligent transportation systems, intelligent manufacturing control systems, energy consumption optimizers and smart wearable computing systems. Systems integration, particularly in business domains, is driven by user considerations, while there are additional demands for providing seamless user experience and transparency across different platforms. Research in requirements analysis for highly integrated systems is anticipated to focus on analyzing the interdependencies among physical environment, human behavior and software system.

(iv) RE techniques for self-managing systems. Self-management capabilities are essential for software systems which are aware of their context and continuously react and adapt to contextual changes. Examples are mobile devices which provide services according to the user location, control systems able to

recover dynamically from failures and intrusion detection systems which handle security breaches at run time. For this contemporary class of systems, the current research work on specifying and verifying their adaptive behaviour (e.g., [21]) is expected to gain a more growing interest. Another promising direction of research is investigating theories from other disciplines (e.g., biology) in order to define metaphors for discovering and analysing requirements for future adaptive, self-managing computer systems [22].

(v) Effective/efficient techniques for globally distributed RE. Current trend in software development is the increased distribution of requirements processes across functional, organizational and geographic boundaries. Distributed requirements enhance the needs for achieving parallelism in RE by distributing requirements efforts among multiple independent organizations, supporting outsourcing of RE tasks, exploiting time zones differences to improve the process performance, managing distributed teams of stakeholders and narrowing communication gaps among diversified stakeholders. In this context, we can expect the development of more advanced collaborative tools to support the RE process for distributed requirements. Some tool developments (e.g., [23]) indicate the importance for further R&D work in this area.

(vi) Requirements reuse for software product lines. Software product lines/families have received a great adoption mainly in the embedded systems industry. Due to their success, product families are characterized by a broadening of their scope and require different approaches than conventional integration-oriented approaches. Failing to identify the commonalities/variabilities and reusable requirements artefacts of a product line, results in unresponsiveness of the platform, difficulties to deal with cross-cutting features and architectural challenges due to over-engineering. We can expect advances in requirements reuse techniques for software product lines which will further improve/validate existing interesting solutions (e.g., multi-level feature trees [24] and hierarchical composition of product families [25]).

(vii) Research in more synergetic RE approaches. Surveys regarding the current practice of RE reveal that a rather small subset of the available RE techniques is utilized by requirements engineers/designers. There are possibilities for further research and empirical study towards more integrated methodologies which interrelate available methods/technologies into coherent, systematic and synergetic engineering processes. Such a research orientation will result in well-defined approaches which will certainly improve the productivity, the quality and the adoption of RE in industry settings. There has been some promising works in this research topic (e.g., [26]) which shows that further research is expected on how to integrate various RE methods/technologies and propose effective methodologies.

(viii) More "sustainability" oriented RE. The process of eliciting and describing software requirements is, nowadays, more tightly related to the goal of attaining sustainability in the global environment. According to the editorial note of the special issue on the IEEE Int. RE Conf. RE'08 [27] there are two critical challenges for RE: "First, when the target domain of software to be developed is directly/indirectly involved in the environmental issues, complex factors affecting the global environment should considered when analyzing and defining requirements. Second, only by fully understanding stakeholders' needs, and documenting them in concise/unambiguous ways, we can consistently deliver quality products designed to meet the complexities of the advanced information society. Failing in engineering high quality requirements will surely bring in an unsustainable information society".

ACKNOWLEDGMENTS

This research has been co-financed by the EU and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework. The work has been partially financed by the R&D program "SPRINT SMEs" which take place in the context of the "ARCHIMEDES III" National Research Programme.

REFERENCES

[1] Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: a roadmap. In Proc. of the IEEE Int. Conf. on Software Engineering, pp. 35–46.

[2] Zave, P. (1997). Classification of research efforts in requirements engineering. ACM Computing Surveys, 29(4), pp. 315-321.

[3] Cheng, B.H.C., & Atlee, J. M. (2007). Research directions in requirements engineering. In Proc. of the IEEE Conf. on Future of Software Engineering, pp. 285–303.

[4] Lindquist, C. (2005). Fixing the requirements mess. CIO Magazine, pp. 52-60.

[5] Glass, R. L. (2005). IT failure rates, 70% or 10-15%? IEEE Software, 22(3), pp. 110-112.

[6] Peterson, D. K., Kim, C., Kim, J. H., & Tamura, T. (2002). The perceptions of information systems designers from the US, Japan, and Korea on success and failure factors. Int. Journal of Information Management, 22(6), pp. 421-439.

[7] Kappelman, L. A., McKeeman, R., & Zhang, L. X. (2006). Early warning signs of IT project failure: The dominant dozen. Information Systems Management, 23(4), pp. 31-36.

[8] McManus, J., & Wood-Harper, T. (2007). Understanding the sources of information systems project failure. Management Services, 51(3), pp. 38-43.

[9] Emam, K. E., & Koru, A. G., (2008). A replicated survey of IT software project failures. IEEE Software, 25(5), pp. 84-90.

[10] Chaos Report, Standish Group, 1994.

[11] Hansen, S., Berente, N., & Lyytinen, K. (2009). Requirements in the 21st century: current practice and emerging trends. In Lyytinen, Loucopoulos, Mylopoulos, Robinson (eds.), Design RE: a Ten-Year Perspective. Lecture Notes in Business Information Processing, 14, Springer, pp. 44-86

[12] Capers, J. (2008). Applied software measurement. McGraw-Hill.
[13] Jarke, M., Loukopoulos, P., Lyytinen, K., Mylopoulos, J., & Robinson, W. (2009). High-impact requirements for software-intensive systems: a manifesto. Informatik-Spektrum, 32(4(2009), pp. 352-353.
[14] Jarke, M., Loukopoulos, P., Lyytinen, K., Mylopoulos, J., & Robinson, W. (2009). High-impact requirements for software-intensive systems: a manifesto. Informatik-Spektrum, 32(4), pp. 352-353.
[15] Bergman, M., King, J.L., & Lyytinen, K. (2002). Large-scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering. Requirements Engineering, 7(3), pp. 152-171.
[16] Cockburn, A. (2006). Agile software development: the cooperative game. 2nd Edition, Addison-Wesley.
[17] Pettersson, F., Ivarsson, M., Gorsheck, T., & Ohman, P. (2008). A practitioner's guide to lightweight software process assessment and improvement planning. Journal of Systems and Software, 21(6), pp. 972-995.
[18] CMMI (2002). CMMI for systems engineering/software engineering / integrated product and process development/supplier sourcing. CMU/SEI-2002-TR-011.
[19] El Emam, K., Drouin, J. N., & Melo, W. (1998). SPICE: the theory and practice of software process improvement and capability determination. IEEE Press.
[20] Napier, N. P., & Mathiassen, L. (2009). Combining perceptions and prescriptions in requirements engineering process assessment: an industrial case study. IEEE Transactions on Software Engineering, 35(5), pp. 593-606.
[21] Zhou, Z., Zhang, J., McKinley, P. K., & Cheng, B. (2006). TA-LTL: Specifying adaptation timing properties in autonomic systems. In Proc. of the IEEE Workshop on Engineering of Autonomic & Autonomous Systems, pp. 109-118.
[22] Goldsby, H. J., Knoester, D.B., Cheng, B. H.C., McKinley, P. K., & Ofria, C. A. (2007). Digitally evolving models for dynamically adaptive systems. In Proc. of the 2007 Int. Workshop on Software Engineering for Adaptive and Self-Managing Systems, p.13.
[23] Damian, D., & Moitra, D., (eds.) (2006). Global software development. IEEE Software, Special issue, 23(5), 2006.
[24] Reiser, M. O., & Weber, M. (2006). Managing highly complex product families with multi-level feature trees. In Proc. of the 14th IEEE Int. Requirements Engineering Conf., pp. 146-155.
[25] Bosch, J. (2006). The challenges of broadening the scope of software product families. Communications of the ACM, 49(12), pp. 41-44.
[26] Broy, M. (2006). The 'Grand Challenge' in informatics: engineering software-intensive systems. IEEE Computer, 39(10), pp. 72-80.
[27] Tamai, T. (2009). Introduction to the RE'08 special issue: requirements engineering for a sustainable world. Requirements Engineering, 14(4), pp. 229-287.

★★★