Requirements Volatility in Multicultural Situational Contexts

Errikos Siakas[1], Harjinder Rahanu[2], Elli Georgiadou[2], Kerstin Siakas[3,4]

[1]*National Archaeological Museum, Athens, Greece, esiakas@culture.gr*

[2]*Middlesex University London, London, UK, h.rahanu@mdx.ac.uk, elli.georgiadou@mdx.ac.uk*

[3] *International Hellenic University, Thessaloniki, Greece, siaka@the.ihu.gr*

[4]*University of Vaasa, Finland, ext-ksia@uwasa.fi*

Abstract

Requirements' volatility refers to additions, deletions, and modifications of requirements during the system development life cycle. Different approaches in software development, including Agile and DevOps, have addressed requirements volatility by increased user participation throughout the whole development process.

In this paper we analyse requirements volatility from a situational context angle with the aim to increase understanding of the role of culture and cultural diversity in a multicultural requirements elicitation process. Research on situational context in Requirements Engineering (RE) is rather limited, despite the recognized importance of RE and requirements elicitation for improving the quality of the final system and software product.

This paper builds on an extensive literature review demonstrating the importance of raising awareness and understanding of the role of culture and cultural diversity for requirements volatility, as one of the most significant situational factors in the requirements elicitation process, with the aim to improve the whole systems development process as well as the resulting products and services.

The paper concludes with the presentation of the Requirements Cultural Volatility Framework which aims to reveal potential conflicts that may occur in requirements elicitation on a multiplicity of cultural dimensions, The framework proposes actions to be taken in order to address the conflicts and point out expected benefits on each dimension.

Keywords: Requirements Engineering, Requirements Management, Requirements Development, Requirements Elicitation, Requirements Volatility, Situational Context

1 Introduction

1.1 Global Software Development

In Global Software Development (GSD), also called Distributed Software Development, software development teams and stakeholders with an interest in the product or service under development are located in different parts of the world. GSD increasingly takes place mainly because organisations seek cost-effective alternatives outside the boundaries of countries or access to an international talent pool [53]. These decisions are motivated by various reasons with different benefits, such as potential costs reduction, access to certain professional skills, increased competitiveness, flexibility of hiring practices, risk mitigation related to labour and taxes, and possibility of expansion into new markets [9, 39]. GSD takes place between a mother organisation and its subsidiaries, partners in outsourcing arrangements and joint ventures (e.g., research and innovation projects) with partners from different countries and often from different organisations.

Despite many advantages, GSD has resulted in challenges for stakeholders which are not collocated projects, placed together in software development projects developed in the same location [19]. The characteristics exhibited by GSD show significant differences from traditional or collocated software development. Due to system development with team members and stakeholders being in different geographical locations, there are differences in cultures and time zones [25, 48] which impact communication and coordination processes (2, 35, 49). As a result, the frequency and quality of communication between system engineers within the development team diminishes and between members of the development teams and stakeholders, coordination becomes difficult, and trust declines [20, 49]. Additional challenges of GSD include [3]:

i. distinctive approaches to systems development;
ii. different processes to follow;
iii. disparate ways of doing things in diverse organisations;
iv. different technical capabilities of remotely distributed team members;
v. reduced visibility of the development work carried out at disparate sites.

1.2 Requirements Engineering

Requirements Engineering is widely studied in the field of system and software development to denote the systematic handling of requirements.

RE is an organized way to understand the system domain and context, to accurately capture, analyse, model, verify and validate the requirements/needs of the stakeholders [3]. After requirements definition the requirements are conveyed to the development team who aim to develop a product or service as described and modelled by the requirements engineers (49). The management and follow-up of changes/ enhancements in system and software requirements are also parts of RE.

1.3 Requirements Elicitation

Requirements elicitation is the first step, a multifaceted and iterative activity, in the RE process, and it is dedicated to uncovering, extracting, and surfacing the needs, requirements, expectations and preferences of users/customers, including their tacit knowledge [49]. The requirements elicitation phase aims to collect information and viewpoints regarding application domain, business requirements, user/customer requirements, constraints, security requirements, information requirements, standards, etc. [43]. It is fundamentally a concerted human activity regarding requirements determination with intensive and extensive communication between the requirements engineers and the various stakeholders, such as customers, end-users, domain experts, project owners, etc. [49]. Effective communication, collaboration, and negotiation with all relevant stakeholders, often having diverse knowledge domains is needed. This can be achieved through authentic commitment of all involved parties for accomplishing requirements development and prioritization.

The fundamental activities in requirements elicitation start with the identification of potential stakeholders and their needs, requirements and wishes. Stakeholders are also those who in the end will accept (or not) the system and judge its quality. Stakeholders, such as end-users and people affected by the system may be numerous, unknown or cannot be easily identified for participating in requirements elicitation activities. Current approaches try to deal with stakeholders outside organisational reach by online polls, questionnaires, or pilot studies (49). Four fundamental activities are involved in the requirements elicitation process [17, 18], namely communication, setting priorities, negotiation, and cooperation with stakeholders.

It is important that the requirements engineers fully understand the system domain. Traditional approaches to requirements elicitation do not consider factors such as distance, time zones, cultural diversity, communication, and language barriers. Effective communication and collaboration between requirements engineers and stakeholders are of the utmost importance. In GSD decreased communication, (informal communication is lacking), geographical distance, cultural diversity, differences in time zones and language barriers create difficulties for effective communication [10]. As a result, the ways by which requirements elicitation is usually being performed in collocated system and software development environments cannot be effectively utilized in a globally distributed environment [47].

1.4 Focus of this paper

As culture plays a major role in the way individuals communicate and collaborate it is also an important factor in the RE process, and has a critical impact on system and software quality and costs [6]. The culture of the individuals participating in the RE process may well influence both the RE process and the resulting product development. This work examines the RE and the requirements elicitation process from a cultural situational context viewpoint.

The focus of this paper is on the meaning and role of culture in Global Software Development (GSD) and the influence of culture on the requirements elicitation process. As a result of an extensive literature review and longitudinal experience from different software development approaches, such as Waterfall, Incremental, Iterative, , Agile and DevOps, we aim to bring the challenges in multicultural situational contexts into view. We aim to raise awareness and understanding of culture as a crucial factor for successful requirements elicitation in global settings, the quality of the whole system and software process, as well as of the resulting products and services, are likely to increase. It also aims to be a tool for requirements engineers to identify risks of volatility so that they can be minimized at an early stage.

The remainder of this paper is organized as follows. In the following section, we introduce prior work identifying how culture is viewed as a Situational Factor and based on two Situational Classifications we explain and justify

our approach to concentrating on the impact of culture on the RE process. The amplified challenges particularly relating to the culture of Software Engineering (SE) in global and multi-cultural environments, in which people with different national, organisational, team and professional cultures need to communicate and collaborate are discussed and the Cultural Impact on Systems and on RE are presented together with potential impact of Hofstede's each dimension [22]on RE. The paper continues with looking at trust, knowledge sharing and maturity through the lens of requirements elicitation and presents three case studies of potential volatility cases. Finally, we demonstrate a Requirements Cultural Volatility Framework that aims to provide a viewpoint regarding potential conflicts that may occur on a multiplicity of dimensions and interactions. The framework proposes actions to be taken in order to address the conflicts and show the expected benefits in personal, team, organisational and national cultures. Conclusions and Further Work complete the paper.

2 Requirements volatility

Requirements volatility is defined as "*the emergence of new requirements or modification or removal of existing requirements*" [15]. It is the most critical risk factor in systems and software development projects, causing problems such as higher defect density, low performance, project delays and cost overruns [11, 15, 61]. Requirements volatility, as a major source of risk, is often underestimated, particularly the impact of small changes to requirements in the later stages of the development lifecycle [13, 63]. Managing requirements volatility is paramount to success. Requirements are usually not fully known or understood in the beginning of a project. Organisational goals and objectives, policies, structures, and work roles of intended end users are likely to change during a system's development. New requirements and alterations to requirements can appear during any development phase because customer/user needs may mature due to increased knowledge brought on by the development activities, or new needs may surface because of unforeseen organisational or environmental changes and pressures [7]. If such changes are not taken into consideration, the original requirements will become incomplete and inconsistent with the new situation. Requirements are usually determined by individuals who often have conflicting needs and goals. In a global context these individuals may come from different national, organisational and team cultures with different values and preferences. An iterative process for requirements elicitation is proposed to lessen volatility risks.

The requirements volatility factor consists of the following sub-factors:

Scope creep: changes and uncontrolled growth in a project's scope at any point of the system and software development lifecycle [8[without addressing effects on budget, time, resources and customer approval. Adding unauthorized features or functionality to a project is usually a result of poor initial scope definition, poor change control, lack of identification of what is required in order to achieve the project objectives (poorly defined initial requirements), poor communication between stakeholders, as well as lack of initial product versatility and adaptability. Scope creep often requires increased amount of work to be done and increased costs. To avoid scope creep minor adjustments to the original project scope should be carefully re-estimated and communicated to involved stakeholders. Scope creep in Agile methodologies can be found within the iterations.

Ambiguous system and project requirements: unclear or unspecified requirements. Ambiguity in requirements elicitation can be attributed to communication problems between the customer/stakeholder and the requirements engineer. The customers may fail to explain their needs (sometimes even to understand them) or the system/software engineer may fail to understand the users' needs and the system context. Ambiguity in RE contributes to system difficulties because the RE process fails to specify unique requirements for the system (59) and leads to confusion, wasted effort and rework.

Continually changing system and project requirements: requirements that are not stable but are likely to change during the development process. Many projects start without a clear idea of what needs to be accomplished due to change in external factors, indecision, hidden assumptions, unfolded tacit knowledge etc. Increase in effort and cost is the most significant impact of changing requirements. An Agile rapid prototyping (8) or a Minimum Viable Product (MVP) approach are considered suitable in approaches of continuously changing requirements.

Ill-defined project goals: too many features or neglected critical functionality. Many projects slow down due to lack of clear definitions of aims, objectives, goals, and desired measurable outcomes. Goal analysis approaches emphasize the use of the goal notion for understanding or describing aspects of the real world, in an attempt to rationally find better ways of coping with the complexity of human affairs [38].

Abundance of features: a large quantity of software and system features. System/software engineers tend to develop more features than needed in pursuit of customer/user satisfaction [28]. This usually leads to increased complexity, increased effort and costs and is in discord with lean system development. A study in 2020 surveying 600 software companies in Canada, France, Germany, the UK and US showed that only 80% of developed features were used by the majority of the users [44]. We do not know if the unused features were those requested by customers, but the findings are worrying and prompt for caution regarding maximizing customer value whilst

minimizing waste. Systems and software requirements involve the identification of the dynamic linkage between markets, products and technologies through analysis of customer/user processes and prioritization of their activities [34]. A constant pace of development is difficult to establish where there is not a constant velocity of requirements identification.

Assumptions and Ambiguities: An assumption (whether implicit or invalid) is a statement that is assumed to be true, and it is invalid when the assumed statement is actually not true. Lehman [37] states that implicit and invalid assumptions cause a variety of problems. External changes continually invalidate the initial assumption set where "*inevitably some assumptions remain unknown, even unsuspected, until they cause incorrect results, misbehaviour or worse* [28]. In their extensive review of problems caused by assumptions Al Mamun and Hansson [1] concluded that implicit assumptions are usually made by lack of consciousness about the pitfalls of implicit assumptions or due to political reasons within the organisation. In addition, they identified that there is no awareness of the (implicit) assumptions among the stakeholders/people. Implicit assumptions thus represent tacit knowledge, which has been identified by the knowledge engineering discipline as volatile and thus challenging to preserve and transfer. *'Assumptions can arise from ambiguous descriptions (whether intentionally or unintentionally) which are a feature of natural language. Ambiguity can result from misunderstandings and misinterpretations of terms between developers and between the various stakeholders. Although the English language is the lingua franca of Computing and Software Engineering members of a multilingual and multicultural team might understand concepts slightly differently. Such misunderstandings and misinterpretations can lead to postponement of decisions and to the development of systems that do not behave as intended by the users although the developers may honestly believe they have followed the requirements*' [27].

Elwahab et al. [15] investigated the causes, the impact of requirements volatility and developed a framework for managing requirements volatility. The causes of requirements volatility were found to be attributed to:

i. Business environment changes: government regulations, factors related to market, competition, financial, policy, technological and legal changes etc.
ii. Changes in development requirements: requirements error, conflicting requirements, missing requirements, evolving user and technological needs, missing team members, cost upgrades etc.

The requirements errors were classified into:

i. People Requirement Errors: the communication gaps between stakeholders, poor participation between the development team and management team, limited understanding of domain knowledge, users with unreasonable timelines and limited knowledge of what they want;
ii. Process Requirement Errors: unsuitable processes, inadequate analysis of requirements, using old process and methodology, inadequate method of achieving objectives and requirements change during the project;
iii. Documentation Requirement Errors: Team may not understand the policy of the user's organisation or use of standards.

Requirements volatility was found to impact on increase in the size of the project, increased defect density, additional work, effort and cost (often >20%) and schedule duration [15]

The framework they propose includes the following phases:

i. Elicitation and analysis of business: The requirements are documented in the Software Requirements Specification (SRS), which aims to ensure that requirements are visible to and understood by all stakeholders.
ii. Specification validation: the requirements are validated according to the following requirement measures, namely: correctness, un-ambiguity, completeness, consistency, verifiability, modifiability, traceable, understandability of user needs, importance and stability.
iii. Requirements volatility causes: analysis of the requirements volatility causes.
iv. Change control management: The change control process, including details of the change and change decision according to volatility measurements (priorities, necessary status and working hours), approval prior to implementation and finally documentation in the change log before implementing the change, need to be documented and managed.

Important sources with detailed explanations of the different phases in requirements engineering can be found in [65-67].

Dasanayake et al. [11] depicted requirements volatility as shown in figure 1. They interviewed fifteen software architects involved in different projects in a large-scale software development company developing smart phone applications. The customers of the company were globally distributed in multiple development sites. Communication was challenging due to distance and time differences of the virtual teams, who used different terminology and concepts due to different domains, language barriers and cultural differences. Requirements elicitation was carried out by using techniques such as focus groups, beta testers, and direct customer communication. Customer needs were clarified during requirement analysis. The product owner, who made the decisions about the product road maps, figured out the high-level product requirements, and created requirements prioritizations. The product owner was the link between the software team and product management. The product owner, who also monitors the progress, had the responsibility to communicate with the product management to identify the product requirements and then convert them into user stories together with the software team. At this point, requirements volatility factors may include changing customer needs and evolving technological understanding.

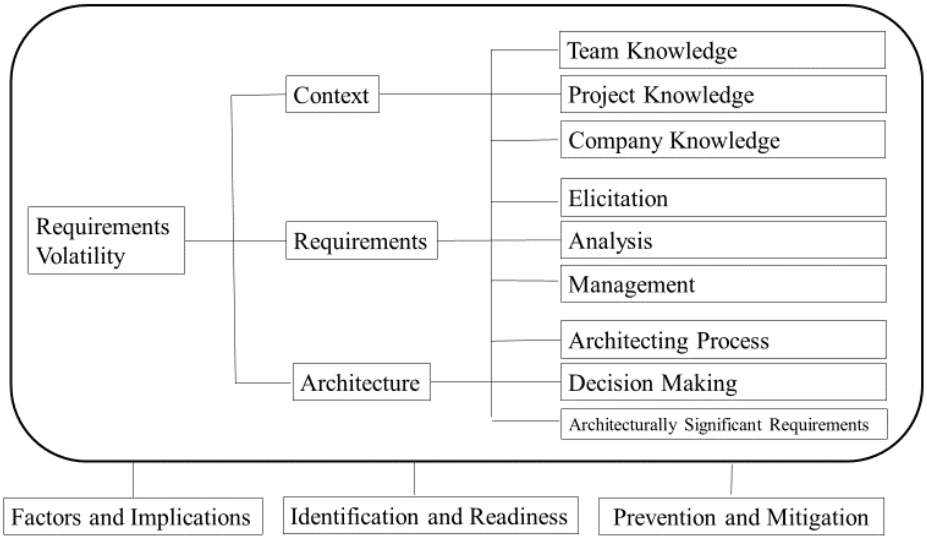

Figure 1 Requirements volatility depicted [adapted from 11]

Figure 1 shows that the context, the requirements themselves and the architecture are impacting on requirements volatility. Identification of the different volatility risk factors and the implications and consequences they cause contribute to the readiness of minimizing the impact of volatility on the development process and the final product. Motivation and prevention actions in the original framework are changed to prevention and mitigation actions, because when there is a risk or a problem we need ways to avoid it and if that is not possible to address it; in this way we minimize the factors that may contribute to volatility of requirements. Dasanayake et al. [11] identified that each team although working in the same company had different combination of characteristics such as project type, stakeholders and team culture, and hence also different readiness and solutions to requirements volatility.

Software architects were not directly involved in customer requirement elicitation and analysis. The final outcomes of the study showed that poor communication, distorted information, and external dependencies were the primary factors that impeded software teams from choosing the optimal course of action to design software architecture, lead to inadequate documentation, inability to trace design rationale and increased complexity of the architecture. Dasanayake et al. [11] state that the adoption of agile software development which is designed to accommodate possible future changes and was found to have better readiness for requirements volatility.

Similar results were found by Zowghi and Nurmuliani [63], who identified that the more frequent developers and customers communicated with each other during RE, the less volatile were the requirements. Their findings also suggested that the more user representatives were involved in the development team the more volatile were the requirements. This seems to suggest that frequent discussions with a small number of customer/user representatives are likely to cause less culturally based volatility. With fewer people involved ambiguity is likely to be minimised.

Thakurta and Ahlemann [61] investigated organisational practices dealing with requirements volatility, and how the adopted project execution strategy with regard to process model selection decisions influences on this risk. They identified and present in order of significance twelve different strategies to managing projects that had resulted in volatility as well as seven factors influenced by business when selecting execution strategy.

The management strategies identified are:

i. involving the business side in the project,
ii. project scope negotiation;
iii. rescheduling project deadlines;
iv. engaging in requirements management activities;
v. documentation of processes procedures and activities;
vi. adjusting project human resources;
vii. using expert knowledge;
viii. focusing on communications;
ix. reducing project complexity;
x. readjusting project effort;
xi. variable costing of additional requirements,
xii. architecting product to withstand change and training workforce.

The seven business factors include in order of significance:

i. influence of business/customers;
ii. complexity of the project;
iii. management preferences;
iv. requirements set of the project;
v. maturity of the development process;
vi. project initial estimates (size, effort);
vii. project ultimate objectives.

The influence of business/customers emerged as the highest determinant. They also identified a gradual shift towards customization. The observations of the researchers revealed that project managers tend to seek to manage the changes instead of trying to prevent them, decisions regarding selection of execution strategies in projects at risk were largely influenced by business, and finally agile methodologies were found to be the most suitable software development approach for handling volatility risks.

3 Culture as a situational context factor

Context is the situation, circumstances, or specific setting in which an event occurs. The requirements of a system are defined in a context where many factors, such as project, team, and organisation characteristics, as well as business domain, contract type and stakeholder involvement influence the results of decision taking [12). The situational context is the reason why something is occurring in a certain way, and it drives the actual behaviour and actions associated with the situation.

In this study we investigated the literature regarding the role of culture as a situational context factor. Situational factors are conditions in the environment that influence people's behaviour [31), and include societal, individual, and organisational factors [32, 33]. They influence the enculturation processes [individual and group socialization, knowledge acquisition, leaders, heroes and myths) that influence e.g., the Information Technology (IT) cultural assumptions, which in turn create certain outcomes (conflict over IT direction, innovation, integration of IT with business strategy). Similarly, the outcomes influence the situational factors and the enculturation processes, and the IT cultural assumptions also influence the enculturation process, in a recursive and ongoing process over time.

In the next section we examine different classifications of situational factors in RE in order to show how they are related to each other, and also in order to establish how culture fits into the classification. We present extracts from two classification schemes [8, 31] that cover situational factors which can lead to varying situations during the RE process in global environments.

3.1 Classification of Situational Factors

Requirements elicitation is the most complex and challenging task when it takes place within a certain organisation in collocated mode, since it is carried out by people (requirements engineers and stakeholders) with different knowledge, different professional values, terminology, and work-related values. In global environments Requirements Elicitation becomes even more complex and challenging [42, 51, 52]. Khan et al. [29, 30, 31] argue that it is of the utmost importance to evaluate the situational factors before selecting the appropriate process for any project, because otherwise situational factors can undermine the competence of the RE team to determine the constraints and characteristics of the system under development.

Table 1. Situational Factors Classification Categories (extracted from 31).

Categories	Individual Factors
Organisation	Organisation Resource, Organisation Strategies, Organisation Standards, **Organisation Culture,** Power Relations, Organisation Learning Exposure, Organisation Politics, Organisation size, Organisation Practices, Organisation Technical Maturity, Organisation Regulations, Organisation Environment, Organisation Structure, Organisation Process Maturity
Stakeholders	Team, Client, Stakeholder Relationship, Stakeholder Interaction Mode, Requirements Realization, Stakeholder Performance, Stakeholder maturity
Project	Problem Domain, Project Requirements, Project Characteristics, Requirement Evolution, Project Goals, Requirements Estimation, Project Risks
Process	Tasks, Techniques, Process Selection, Process Maturity, Tools, Management Knowledge, Management Performance, Management Decision, Quality Metrics, Defect Occurrence, Testing Accuracy

Khan et al. [31] categorized 37 situational factors, as shown in Table 1. Culture is mentioned explicitly as Organisation Culture within the Organisation category. However, this is misleading because culture is a human construction that exists wherever humans are involved. It includes intangible common aspects of social life and practices, way of communicating, thinking, and behaving that a certain group of people share [22, 23, 24]. Culture encompasses the social beliefs and behaviour, assumptions, values, attitudes and norms that can be used to define them as a collective. Hence, we understand that Culture cannot be omitted from the Stakeholders category, because we can identify different stakeholder cultures, team culture etc.. Similarly domain knowledge should be included in stakeholder category, because knowledge of the domain is imperative for understanding what requirements are needed.

Similarly, Clarke and O'Connor [8] developed a comprehensive Situational Factors Reference Framework shown in Table 2 that affects the software development process. The table consists of 8 categories divided into 44 individual factors and 170 sub-factors. They state that all situational individual factors have some level of influence on each other and on the system development process [8]. They conclude that such inter-dependencies create a complexity difficult to decompose or eliminate as many dependencies are generated by loose decisions taken on a lose foundation. We consider that these inter-dependencies can also be attributed to culture as an umbrella factor. In Table 2 we can see that Culture is explicitly mentioned only in the Personnel category. However, we postulate that culture operates in all categories. For example, the following categories in table 2 are linked to certain obvious factors, such as:

i. Organisation: organisational culture etc.;
ii. Operations: end-user culture etc.;
iii. Management: management culture etc.;
iv. Business: business culture etc..

Table 2. Situational Factors Classification Categories (extracted from 8)

Categories	Individual Factors
Personnel	**Culture**, Experience, Skills, Team size, Cohesion, Productivity, Disharmony, Commitment, Turnover
Technology	Emergent, Knowledge
Requirements	Feasibility, Rigidity, Standards, Changeability
Organisation	Maturity, Management commitment, Stability, Structure, Facilities, Organisation Size
Operation	End-Users, Prerequisites
Management	Accomplishment, Expertise, Continuity
Business	Customer satisfaction, Opportunities, Business drivers, External dependencies, Time to market, Payment terms, Potential loss
Application	Application size, Type of application, Complexity, Quality, Performance, Development phase, Degree of risk, Connectivity, Component reuse, Deployment profile, Predictability

With reference to Tables 1 and 2 there is a marked difference in the defined categories and the individual factors. Both perspectives can, however, be considered accurate, depending on what viewpoint the factors are investigated from. Since culture is '*the collective programming of the mind'* as defined by Hofstede [22, 23] it is certainly restricted to people, hence requirements engineers and stakeholders of the requirements elicitation process. When people come from different backgrounds with different behaviours, values, norms and patterns of perception and preferences, they see and understand things differently and put emphasis on different aspects. These cultural points

of view need to be identified and demystified during the requirements elicitation phase of any project for maximizing the production of high-quality products, and for ensuring the satisfaction of stakeholders.

Our experience from longitudinal studies about cultural influence on GSD [49, 50, 53, 51, 52, 57] is that culture is impacting on all the other often interrelated categories, and thus, a complex network of situational dynamics is created.

In Table 1 culture is referred to as organisational culture in the *Organisation* category. In Table 2, culture is mentioned in the *Personnel* category. Clarke and O'Connor [8] also sub-divided the cultural factor into team culture and resistance to change. These viewpoints are only a part of the truth. Culture is the cornerstone of the values and practices found in human behaviour's and therefore of enormous importance for all global transactions, including requirements elicitation.

Management of cultural diversity is similarly of the utmost importance in today's globalized world, because increasing number of companies involved in joint ventures, cross-national partnerships, outsourcing relationships and subsidiaries need to embrace people from a variety of ethnic backgrounds and cultures. Due to globalization, which has contributed to increased integration of labour markets, the need to develop cultural sensitivity is generally emphasized and a new awareness of the importance of understanding other cultures is rising [53). Ramesh et al. [46) for example advocate that project managers should evaluate the compatibility of each practice of the software lifecycle with the national culture of stakeholders to ensure that the selected methods/practices are appropriate for the cultural contexts in which they are applied.

An optimal systems development process is dependent on the situational characteristics of individual system development settings, such as the nature of the application or system under development, team size, requirements volatility and personnel experience [8] but studies on situational context in RE are very scarce.

4 RE in different Software Development Approaches

In this section we are looking at RE in different software development approaches. In the Software Process Improvement (SPI) approach the requirements elicitation process is considered a robust cornerstone. However, the SPI process is considered bureaucratic and time consuming and thus mainly used today for large safety-critical systems [55]. Ramasubbu et al. [45] examined a leading offshore software service company that employs over 19,000 people in 17 countries worldwide, on whether widely adopted structured software processes of the Capability Maturity Model Integrated (CMMI) are effective in mitigating the negative effects of work dispersion in offshore software development. The results from 42 projects completed in a two-year period indicated that adoption of structured process models have both a direct and a learning-mediated effect in mitigating the negative effect of work dispersion in offshore software development. Investments in the structured key process areas (KPAs) as specified by CMMI mitigate the negative effect of work dispersion on productivity and quality. Investments in conceptual learning contributed to improved quality; operational learning investments were associated with improved productivity.

In Agile and Lean software development, the RE process is by nature incremental and iterative because requirements are considered to evolve as the customers may change their mind or because of changes in the overall technical and socio-economic environment [62]. Agile methods provide a viable solution when the software to be developed has fuzzy or changing requirements [60]. In Agile development, the system and software requirements are incrementally specified and prioritized [58] in an iterative process along with the actual end-user/customer (often on-site) and other stakeholders. Valuable feedback regarding the product is received through sprint reviews, also created together with the customer/stakeholder. Hence, it can be said that one of the interactions that facilitate the RE process is the sprint review meetings that is used throughout the software lifecycle [26]. In Agile development there are also roles, such as the User Experience Designer, which is normally used to obtain feedback from the end user in an iterative process. During frequent planning meetings, new requirements are communicated to the development team [58], as design mockups/wireframes/user flows etc. The aims are to support the development team to mature regarding requirements and readiness [46]. Communication, both formal and informal, is crucial in agile development. Hildebrand et al. [21] studied how and to what extent Extreme Programming (XP), a popular agile methodology with pair programming, can be transferred to distributed development projects for large enterprise applications. They found that communication and coordination deteriorated due to spatial separation, temporal, and cultural factors. Social aspects, such as body language, may be difficult to interpret when using technical communication means, and informal communication is principally lacking in distributed environments. Working in different time zones is challenging and as cultural factors they mention differences in language and customs between different cultures. In this paper we try to analyse these external cultural factors and their influence on software elicitation. We also recognize that agile

DevOps, (Development and Operations) is a recent shift in the software engineering domain towards collaboration between development, quality assurance, and operations [5]. The main drivers in DevOps are fewer requirements' changes focused testing, quality assurance, and a fast delivery cycle [40). The aims of DevOps are to deliver value to customers faster and continuously through automation and a predefined way of handling the product; hence reducing problems arising from miscommunication between team members, and accelerating problem resolution [14, 36]. Requirements are defined through stakeholder participation in an advanced RE process with feature mapping and an explicit delivery model [14, 26]. Requirements elicitation is fundamental for the success of the entire system. Requirements are further shaped both by development feedback and through monitoring of operations. In short, we can say that DevOps places focus on the deployment of developed software, whether it is developed via Agile or other methodologies. DevOps also addresses gaps between developers and Information Technology (IT) operations/infrastructure, whilst Agile software development addresses communication gaps between end-users and developers. DevOps does not, to our knowledge, explicitly deal with cultural issues.

5 Requirements Elicitation, Trust, Knowledge Sharing and different maturity levels

5.1 Interorganisational trust

In requirements elicitation at least two organisations or two groups of people are involved. Trust between them is paramount for the success of the process. Ojukwu and Georgiadou [41] emphasised that Inter-Organisational Trust (IOT) is the directional relationship existing between one organisation and another or between one group of organisations and another group or another single organisation in which one party (the trustor) is willing to believe that the other party (the trustee) is willing to honour the terms of their business obligations. They developed an IOT model which they applied and validated in the context of ecommerce.

In the case of GSD, the trustor and trustee come from different cultures. This difference often engenders mistrust and misunderstanding which are likely to hamper the requirements elicitation process. Awareness and cultural sensitivity of this complexity is likely to build trust and thus minimise misunderstandings and facilitate the requirements elicitation process.

5.2 Knowledge sharing levels and maturity levels

Misunderstandings and miscommunication are likely to happen at all stages of the RE process particularly at the requirements elicitation stage. Differences in maturity levels between the developing and the user organisations, reluctance to share knowledge at personal, project team, or organisational level is bound to result in misunderstandings and failures. The I^{5}P Framework [16] visualises performance estimation through the alignment of process maturity and knowledge sharing. As the organisational maturity increases so does the knowledge sharing.

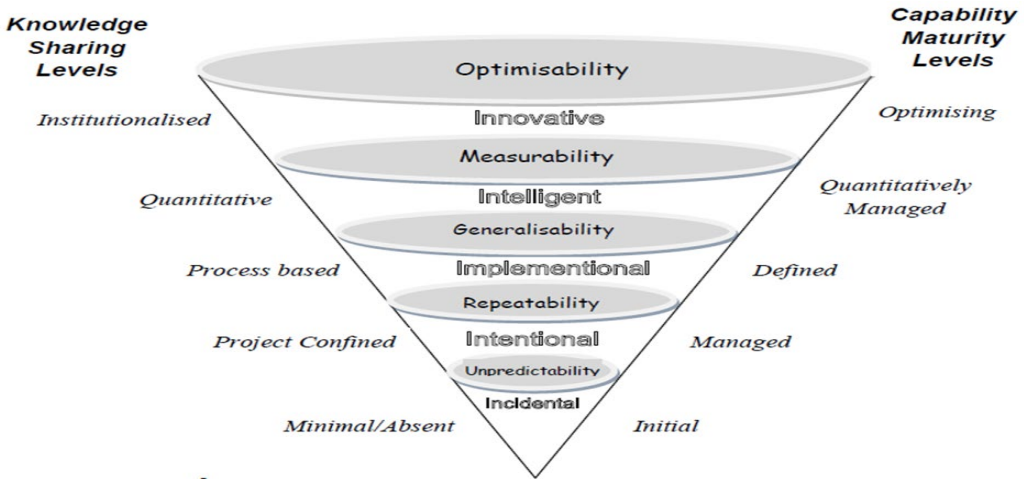

Figure 2 The I^{5}P Framework [16]

The growth of the Internet Of Things (IOT) and of knowledge sharing combined form the conditions for a successful requirements elicitation process.

6 Three case studies of potential cultural volatility causes

Alsanoosy et al. [4] argue that Power Distance (PD) as defined by Hofstede [22) dominates RE activities. In low PD countries lower ranking and higher-ranking employees consider each other as equals and decentralization is popular. On the other hand, high PD countries subscribe to the authority of bosses and to centralization. Zanini and Migueles [64] argue that the intersection between high PD, low trust, and strong Uncertainty Avoidance (UA) seems to create circularities that jeopardize the capacity to solve complex problems.

Case study 1: Requirements elicitation with people with antithetical cultural values

Lack of trust has a negative impact on the requirements elicitation process because it reduces the transparency and communication among stakeholders. Our experience from the Finnish KEMO project[1] showed similar results. The aim of the KEMO project was to help companies that manufacture parcelled goods to successfully begin their production in other counties by identifying, recognizing, and strengthening in advance factors and elements of successful production and logistics, and to eliminate potential risks. The Finnish national culture shows low PD and medium UA [22]. Eight Finnish managers from three different Finnish global organisations dealing with international operations with countries with high PD, high UA and collective characteristics, were interviewed. The results revealed that the first thing the Finnish requirements engineers did was to find out who is the highest ranked person in the hierarchy responsible for taking the final decision regarding the requirements and to consult that person *'because otherwise nothing will be decided.....in the meetings lower employees do not speak up, everything is always OK although it may not be'*. One interviewee in the KEMO project stated *'In the Nordic counties trust is a value. In many other countries trust is not a value. The only way you can build trust is through action and demonstrated capability.'*.... '*As a requirement engineer it is also important that you sit somewhere higher up physically so that they respect and trust you, even if you are at the same hierarchical level. They call me by Mr. and my Surname, while at home everybody other by their first name'*. It was also observed that power distance affects sensitivity of subordinates toward ethical behaviours of leaders. In a high level of PD ethical leaders have higher authority. We conclude that in the requirements elicitation process it is important that the requirements engineers build a trustworthy and ethical relationship with stakeholders.

Case study 2: Requirements elicitation in high PD countries

Similar behaviours were found by Siakas and Mitalas [56] and Siakas et al. [55] from a laboratory course in Software Quality Management in Higher Education (HE) in Greece, (high PD), where groups of four students in each group, simulated a real-life software development system. The educational system in Greece is dominantly teacher-centred and the students depend on the teacher who initiates all communication [22). The students who were responsible for the requirements elicitation asked the teacher to confirm every requirement due to fear of making mistakes, and of not receiving approval by the 'superior'; hence the whole requirements elicitation process was significantly slowed down, and the viewpoints of stakeholders where not taken into consideration if not proved by the teacher (person with higher social status).

Case study 3: Requirements elicitation in high PD countries

Experience from a software development project for a clothing manufacturer in Greece also provided evidence of this 'defence' aspect. The requirements engineer decided to observe the end-users in their daily work because the manager's decisions negatively affected the proposed solution. The difficulty that arose was to convince the manager that the end-users possessed more detailed context knowledge. The opinions of the 'lower-level' end-users needed to be taken into consideration, without creating a cultural conflict (disagreement feared to be interpreted as disrespect). Hence, we conclude that social status in high PD environments negatively affects the outcomes and the quality of the requirements elicitation activity, since lower-level stakeholders do not freely articulate their requirements since they are worried about conflicting with managers or receiving criticisms.

7 The Requirements Cultural Volatility Framework

Building on the context category of volatility presented in Figure 1 we can see that the context category includes three sub-categories, namely team info, project info and company info. We interpret the word info as characteristics, hence team, company and project characteristics, which we further analyse in Table 3.

Hofstede [22] emphasises that "*culture is not the property of individuals, but of groups. It is a collection of characteristics possessed by people who have been conditioned by similar socialisation practices, educational procedures and life experiences"*. Teams and organisations are built by groups of people who adopt certain unique

[1] http://www.uva.fi/materiaali/pdf/isbn_978-952-476-375-2.pdf

attitudes, behaviours, and values; hence they have their own cultures. Cultural identity is a part of a person's self-perception related to nationality, ethnicity, religion, social class, generation, and any kind of social group that has its own distinct culture. National culture expresses the ethnic values of individuals in a country. Countries possess distinct and relatively stable cultures [22]. Similarly, the organisational culture expresses a set of assumptions, beliefs, attitudes, and values that are shared by existing members and are taught to new members of an organisation. It includes prevailing patterns of behaviour and sentiments. Organisational culture is mainly created and maintained in existing frameworks by the founders and the leaders of an organisation through their value system. Organisational culture is often reflected as the result of management activity or is looked at through levels of practices, such as symbols, heroes, and rituals [49]. Team culture reflects assumptions, beliefs, attitudes, and values that are shared by team members. Professional culture is a result of the educational experiences and the socialization process that occur during education and working within a certain profession. Each profession reinforces common values, problem-solving approaches, and language/jargon/terminology. A person can simultaneously belong to and identify with several different cultures.

Table 3: Grouping of context characteristics influencing volatility

Team characteristics	**Organisation Characteristics**	**Project characteristics**

• Level of dispersion
• Cohesion
• Domain knowledge
• Language (native/foreign)
• National culture
• Organisational culture
• Team Culture
• Professional culture (multidisciplinarity)
	• National culture
	• Organisational culture
	  ○ Clan
	  ○ Hierarchical
	  ○ Democratic
	  ○ Disciplined
	• Learning orientation
	• Process orientation
		• Size/Scope/Complexity
		• Effort
		• Predefined processes
		• Process improvement approaches
		• Level of Stakeholder involvement
		  ○ Motivation & Constraint
		• Lifecycle models
		  ○ Waterfall, Spiral, V-Model, Agile DevOps

National values, as defined by Hofstede [22], constitute similar ways of acting and thinking (viewpoints, attitudes, habits, norms and practices) of a nation. The national culture (the ethnic values of the workforce) can be categorised in several dimensions, namely Power Distance, Uncertainty Avoidance, Collectivism/Individualism, Masculinity/Femininity, Long Term Orientation and Indulgence/Constraint and measured on a scale from zero to hundred.

The Organisational culture (the way the organisation is structured and managed) was depicted by Siakas and Georgiadou [50] in the C.HI.D.DI. Typology based on Hofstede's work-related cultural dimensions PD and UA. [22, 57]. These two dimensions are consider to also mirror organisational culture [22, 50, 57]. Each type belongs to a quadrant formed by two dimensions. On the horizontal axis Uncertainty Avoidance (UA) starting from weak UA and ending in strong UA is pictured, while on the vertical axis Power Distance (PD) starting from weak PD and ending in Strong PD is pictured. The four quadrants form a specific typology of organisational cultures pictured in Figure 2, which is called the C.HI.D.DI. (Clan, HIerarchical, Democratic, DIsciplined) typology of organisational cultures. These four types of culture should be considered as ideal types, because the exact measurement of culture is hindered by operational and conceptual problems and by the multidisciplinary complex character of the research domain.

Strong PD	Clan	Hierarchical
Weak PD	Democratic	Discipline
	Weak UA	Strong UA

Figure 2: The C.HI.D.DI. Typology of Organisational Cultures [50]

National cultures influence organisational cultures. Both national and organisational cultures influence team cultures. People can concurrently belong to many different cultures, e.g., through their participation in different teams. Thus, a complex network of people with different kinds of cultural influences need to work together on a project. Stakeholder involvement in a project is considered to increase the understanding of the development team of the project domain. Also, the requirements engineers and the stakeholders possess their own professional culture, namely technical and business-oriented cultures.

The actors involved in requirements elicitation (team from the side of the project development and stakeholders with different interests in the project) have different roles, objectives, backgrounds, domain knowledge, preferences, and priorities. Examples of actors are management (responsible for scope and financial decisions), the product owner/requirements engineer (responsibility to identify the product requirements) and the system/software team, domain experts etc. However, there are basically two key players during the requirements elicitation process, namely customers/users/managers/sales personnel etc. (all stakeholders that have an interest in the end-product) and requirements engineer/product owner (who needs to understand the customer/user expectations of the system and who controls that the agreed requirements are implemented). Hence, two professional cultures are involved, namely the user who views the system from a business perspective and the requirements engineer who views the system from a technological perspective. Thus, a communication gap is inherited in the communication process, which is a risk factor for misunderstandings and volatility. In global settings, with actors coming from different organisational, national and team cultures the communication gap is exaggerated [49].

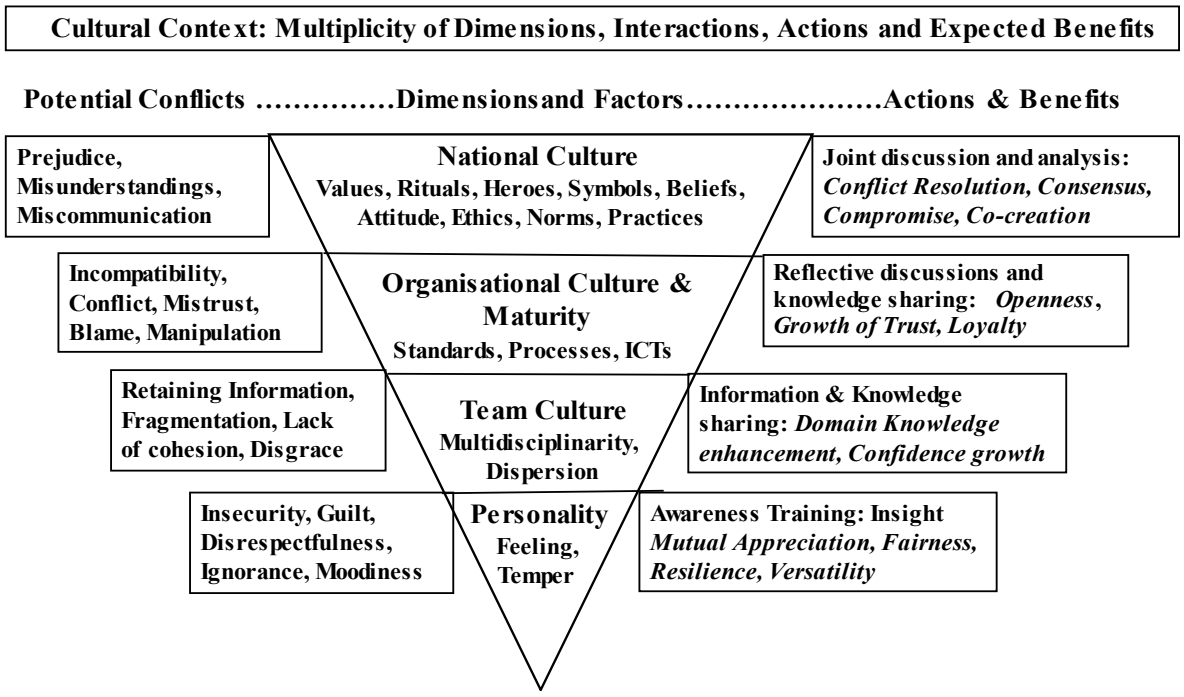

Figure 3: The Requirements Cultural Volatility Framework

The Requirements Cultural Volatility Framework shows that stakeholder involvement in the requirements elicitation activities is crucial because stakeholders are the final recipients of the system under development. However, stakeholders may be reluctant to spend time to explain to requirements engineer the functions of the system under development. In global settings the difficulty of common understanding is amplified due to national, organisational and team diversity. Thus, inevitably the requirements cultural volatility is increased. Figure 3 illustrates the multiplicity of dimensions and interactions. The upside-down triangle (at the top of the inverted triangle) in the middle shows the different cultural dimensions influencing behaviours. The broadest influence comes from the national culture people have grown up in. It mainly reflects the values, which are the deepest level of a culture. The next level is the organisational culture. The more mature an organisation is the stronger is the organisational culture and the stronger the influence on the employees (‘*the way we do things here’*). The following level is the team culture, which depending on the personalities and the influences from the organisational and national culture develops as time goes by and the team becomes coherent.. On the left side of the triangle the potential conflicts are shown stemming from the corresponding dimension on the same parallel. The examples of conflicts shown on the left side on four dimensions/levels are not exhaustive , but representative, for the corresponding dimension. For example, ‘retaining information’ is a cultural characteristic that can happen on all dimensions/levels, but is probably the most tangible on team level. On the right-side of the triangle we show proposed actions to be taken in order to address the corresponding conflicts. We also show in italics the expected benefits in personal, team, organisational and national cultures.

8 Conclusions and Further Work

The aim, of the research presented here, was to report on the challenges of requirements elicitation in multicultural environments. The focus was on Requirements Volatility in Multicultural Situational Contexts. An extensive literature review was carried out covering situational factors affecting the RE process and focussing on requirements volatility, its causes, the various manifestations, and conflicts. As a result of the insights gained a Requirements Cultural Volatility framework was developed to help organisations, teams, and individuals understand the cultural influence on human behaviour of actors in the requirements elicitation process.

As mentioned in the SPI Manifesto[2] '*it is very important to ensure the alignment of SPI initiatives and organisation culture'*. Nowadays software is increasingly developed by virtual teams involving system and software developers and stakeholders from i) subsidiaries in different countries with different national cultures, but same organizational culture, ii) joint ventures and outsourcing relationships with system and software developers and stakeholders from different countries with different national cultures and different organisations with different organisational cultures. In addition, we have different professional cultures between the system and software developer, who is technically oriented and the customer/user who is business oriented. In this paper we tried to emphasise the importance of understanding these differences of cultural values and instead of letting cultural differences become hurdles we claim that awareness and recognition of divergent values could be an advantage for the software process improvement.

Culture and cultural diversity are the most significant situational factors in the distributed requirements elicitation process. There are other important factors in requirements elicitation, but in a global context culture needs particular attention. The importance of raising awareness and understanding of the role of culture and cultural diversity for requirements volatility aims to improve the whole systems development process as well as the resulting products and services. The research presented in this paper reveals that cultural studies in the field of RE are insufficient, and that more empirical studies are needed particularly in view of novel technologies and ways of virtual ways of interacting which tend to result in espoused cultures.

9 References

1. Al Mamun, M.A., Hansson, J. Review and Challenges of Assumptions in Software Development, *2nd Analytic Virtual Integration of Cyber-Physical Systems Workshop (AVICPS),* Vienna, Austria. 2011.
2. Ali, N. and Lai, R. Managing requirements change in global software development, *Proceedings of the 2014 International Conference on Data and Software Engineering*, Indonesia. 2014.
3. Ali, N. and Lai, R. Requirements Engineering in Global Software Development: A Survey Study from the Perspectives of Stakeholders, *Journal of Software,* 13(10):520-532. 2018
4. Alsanoosy, T., Spichkova, M. and Harland, J. Cultural influence on requirements engineering activities: a systematic literature review and analysis. *Requirements Engineering*, 25:339–362, 2020
5. Benguria, G., Alonso, J., Etxaniz, I., Orue-Echevarria, L. and Escalante, M. Agile Development and Operation of Complex Systems in Multi-technology and Multi-company Environments: Following a DevOps Approach, *Systems Software and Service Process Improvement*, pp. 15-27. 2018.
6. Chakraborty, S., Sarker, S., Sarker, S. An Exploration into the Requirements Elicitation: A Grounded Approach, *Journal of the Association of Information Systems,* 11(4): 212-249. 2010.
7. Christel, M.G. and Kang, K.C. 1992. Issues in Requirements Elicitation, Technical Report, CMU/SEI-92-TR-012, ESC-TR-92-012, Carnegie Mellon University, Pittsburgh, Pennsylvania, US.
8. Clarke, P., and O'Connor, R.V. The situational factors that affect the software development process: towards a comprehensive reference framework*. Jl of Inf. Software Technology*, 54 (5), pp.433-447. 2012.
9. Conchuir, E. O., Holmström, H., Ågerfalk, P. J., and Fitzgerald, B. Exploring the assumed benefits of global software development, *1st Int. Conference on Global Software Journal of Software,* 531(13):159–168. 2018.
10. Damian, D. Stakeholders in global requirements engineering: Lessons learned from practice, *IEEE Software*, 24(2):21-27. 2007.
11. Dasanayake, S., Aaramaa, S., Markkula, J., and Oivo, M. . Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements? *Journal of Software: Evolution and Process,* DOI: 10.1002/smr.2160. 2019.
12. Dede, B., Lioufko, I. . *Situational Factors Affecting Software Development Process Selection*, MSc Thesis, Goetenborg University Sweden. 2010.
13. Dev, H., and Awasthi, H. A Systematic Study of Requirement Volatility during Software Development Process, *International Journal of Computer Science Issues*, 9(2):528-533. 2012.

[2] https://conference.eurospi.net/index.php/en/manifesto

14. Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N. . DevOps, *IEEE Software,* pp. 94-100. 2016.
15. Elwahab, K.A., Latif, M.A.E., Kholeif, S. Identify and Manage the Software Requirements Volatility: Proposed Framework and Case Study, *Int. J. of Adv. Computer Science and Applications,* 7(5):64-71. 2016.
16. Georgiadou, E., Siakas, K., Balstrup, B. The I^{5}P Visualisation Framework for Performance Estimation through the Alignment of Process Maturity and Knowledge Sharing, IJHCITP, Vol. 2 No 2. 2010.
17. Grunbacher, P. and Braunsberger, P. Tool Support for Distributed Requirements Negotiation: Lessons Learned. Cooperative Methods and Tools for Distributed Software Processes. *IEEE,* pp. 46--55. 2003.
18. Grunbacher, P., Halling, M., Biffl, S., Kitapci, H. and Boehm, B.W. Integrating Collaborative Processes and Quality Assurance Techniques: Experience from Requirements Negotiation. *Journal of Management Information System,* 20(4): pp. 9-29. 2004.
19. Herbsleb, J. D., and Moitra, D. Global Software Development, *IEEE Software*, 18(2):16-20. 2002
20. Herbsleb, J. D., and Mockus, A. An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, *29(3)*. 2003.
21. Hildebrand, T. Geisser,M., Kude, T.,Bruch, D. and Acker, T. . Agile Methodologies for Distributed CollaborativeDevelopment of Enterprise Applications, *International Conference on Complex, Intelligent and Software Intensive Systems,* IEEE Comp. Soc. pp. 540- 545. 2008
22. Hofstede, G. *Culture's consequences: comparing values, behaviours, institutions, and organisations*- 2nd Ed. - Thousand Oaks, Calif.; London: Sage Publications. 2001.
23. Hofstede, G., Hofstede, G. J., and Minkov, M. *Cultures and Organisations - Software of the Mind: Intercultural Cooperation and its Importance for Survival*. McGraw-Hill. 2010
24. Hofstede, G. Dimensionalizing Cultures: The Hofstede Model in Context. Online Readings in Psychology and Culture, 2(1), (doi.org/10.9707/2307-0919.1014). 2001.
25. Hsieh, Y. Culture and shared understanding in distributed requirements engineering, *Proceedings of the International Conference on Global Software Engineering*, Brazil. 2006.
26. Jabbari, R., bin Ali, N., Petersen, K., and Tanveer, B. What is DevOps? A Systematic Mapping Study on Definitions and Practices. *XP '16 Workshops Proceedings,,* article no. 12. 2016.
27. Kamsties E. Understanding Ambiguity in Requirements Engineering. In: Aurum A., Wohlin C. (eds) *Engineering and Managing Software Requirements*. Springer, Berlin, Heidelberg. 2005.
28. Kauppinen, M., Savolainen, J., Lehtola, L., Komssi, M., Töhönen, H., Davis, A. From feature development to customer value creation, 17th Int. requirements engineering conf., pp. 275–280. 2009.
29. Khan, H.H., bin Mahrin, M.N., and Malik, M.N. Situational Requirement Engineering Framework for Global Software Development: Formulation and Design. Bahria University, *JICT,* 9 (1):74-84. 2016.
30. Khan, H., bin Mahrin, M.N., and Chuprat, S. Situational Requirement Engineering Framework for Global Software Development. *Int. Conf. on Computer, Communication, and Control Technology*, Kedach, Malesia, pp. 224-229. 2014.
31. Khan, H. H., bin Mahrin, M. N., and Chuprat, S. Situational Factors Affecting Requirement Engineering Process in Global Software Development. *IEEE Conf. Open Systems,* Malaysia, pp. 118-21. 2013.
32. Kaarst-Brown, M. L. How Organisations Keep Information Technology Out: The Interaction of Tri-Level Influences on Organisational and IT Culture, Working Paper IST-MLKB, Syracuse University, US. 2004
33. Kaarst-Brown, M.L., and Robey, D. More on Myth, Magic and Metaphor: Cultural Insights into the Management of Information Technology in Organisations, *Inf. Techn. and People* 12(2):192-217. 1999.
34. Komssi, M., Kauppinen, M., Töhönen, H., Lehtola, L., Davis, A. Roadmapping problems in practice: value creation from the perspective of the customers, *Requirements Engineering,* pp. 45–69. 2015.
35. Lai, R., and Ali, N. A method of requirements management for global software development, *Advances in Information Sciences, Human and Science Publication*, USA, 2013,
36. Lampropoulos, G., Morcavallo, A., Salvi, L., Spiralska-Golak, I., Siakas, K. DevOps: The New Frontier of Industrial Software, in O. Khan, P. Marchbank, E. Georgiadou, P. Linecar, M. Ross, G. Staples in *Int. Experiences and Initiatives in IT Quality Management,* 27th SQM, Southampton, UK, pp. 119-130. 2019.
37. Lehman, M. *Development and Evolution*, Address at IDI, Oslo, 16 March. 2006.
38. Lehtola, L., Kauppinen, M., Vähäniitty, J., and Komssi, M. Linking business and requirements engineering: is solution planning a missing activity in software product companies? *Requirements Engineering*, Vol. 14, pp. 113–128.2009.
39. Mighetti, J. P., and Hadad, G.D.S. A Requirements Engineering Process Adapted to Global Software Development. *CLEI Electronic Journal,* 19 (3) Paper 7. 2016.
40. NewRelic 2018. *Navigating DevOps - What it is and why it matters to you and your business,* eBook, https://try.newrelic.com/rs/412-MZS-894/images/NewRelic_DevOps-101-Navigating-DevOps-eBook.pdf.
41. Ojukwu, D. and Gergiadou, E. Towards Improving Inter-Organisational Trust amongst SMEs – A Case Study from Developing Countries, *9th IFIP International Conference on the Social Implications of Computers in Developing Countries,* May, Sao Paolo, Brazil. 2007.

42. Pacheco, C., García, I., and Reyes, M. Requirements elicitation techniques: A systematic literature review based on the maturity of the techniques. *IET Software,* 12 (4):364-378. 2018.
43. Pandey, D., Suman, U., and Ramani, A.K. An Effective Requirement Engineering Process Model for Software Development and Requirements Management. *Int. Conference on Advances in Recent Technologies in Communication and Computing,* IEEE Comp. Soc., pp. 287 – 291. 2010.
44. Pendo-Study, 2020. https://wraltechwire.com/2020/01/28/pendo-study-with-80-of-features-not-used-software-execs-re-evaluating-success-metrics/
45. Ramasubbu, N., Mithas, S., Krishnan, M. S. and Kemerer, C. Work Dispersion, Process-Based Learning and Offshore Software Development Performance. *MIS Quarterly.* 32(2):437-458. 2008.
46. Ramesh, B., Cao, L., Kim. J.J., Mohan, K., and James, T.L. Conflicts and complements between eastern cultures and agile methods: an empirical investigation. *EJIS* 26(2):206–235. 2017.
47. Romero, M., Vizcaino, A., and Piattini, M. Towards a definition of completeness for global requirements elicitation. 13th Conf. on Innovation and Technology in Computer Science Education, pp. 364-364. 2008.
48. Shewell, C. *Good business communicates across cultures: A practical guide to communication.* Bristol: Mastek Publications. 2000.
49. Siakas, E., Rahanu, H., Georgiadou, E., Siakas, K. Towards Reducing Communication Gaps in Multicultural and Global Requirements Elicitation. In: Yilmaz M., Clarke P., Messnarz R., Reiner M. (eds), Systems, Software and Services Process Improvement. EuroSPI. vol 1442. Springer, pp. 257-277. 2021
50. Siakas, K., Georgiadou, E. A New Typology of National and Organisational Cultures to Facilitate Software Quality Management, in E. Georgiadou, G. King, P. Pouyioutas, M. Ross, G. Staples (eds), *Quality and Software Development: Teaching and Training Issues*, the 5th INSPIRE conf., London, pp. 213-226. 2000.
51. Siakas, K., Georgiadou, E., Siakas, D., and Rahanu, H. Developing effective teams in global multidiscipline engineering and manufacturing organisations. in X. Larrucea, I. Santamaria, R.V. O'Connor, R. Messnarz*, Systems, Software and Services Process Improvement*, Springer, Heidelberg, pp. 564-576. 2018.
52. Siakas, K., Georgiadou, E., and Siakas, D. Knowledge Sharing in Distributed Teams: Influence of National and Organisational Culture, in Khosrow-Pour, M (ed). *Entrepreneurship, Collaboration, and Innovation in the Modern Business Era,* Chapter11, April 2018, pp. 221-242. 2018.
53. Siakas, K, and Siakas, D. Cultural and Organisational Diversity Evaluation (CODE): A Tool for Improving Global Transactions. *Strategic Outsourcing Int. Journal,* 8 (2/3), pp. 206 – 228. 2015.
54. Siakas K., Siakas E. The Agile Professional Culture: A Source of Agile Quality, *Software Process: Improvement and Practice (SPIP) Journal*, John Wiley & Sons, Volume 12, Issue 6, pp. 597 – 610. 2007.
55. Siakas, K., Taouktsis, V., Pehlivanis, K., Voutsa, E, Gisis, T. Assessment – Improvement of a Balanced ScoreCard Application; Experiences from a Laboratory Course in Software Quality Management, *the 10th,* INSPIRE, 21-23 March 2005, Glouchestershire, UK, pp. 11-23. 2005.
56. Siakas, K., Mitalas, A. Experiences from the Use of the Personal Software Process (PSP) in Greece; Analysis of Cultural Factors in the 9th INternational Conference on Software Process Improvement - *Research into Education and Training*, Kent, UK, 05-07.04.2004, pp.11-21. 2004.
57. Siakas K. *SQM-CODE: Software Quality Management – Cultural and Organisational Diversity Evaluation*, PhD Thesis, London Metropolitan University, UK. 2002.
58. Schmidt, C.T., Kude, T., Tripp, J. Heinzl, A. and Spohrer, K. Team Adaptability in Agile Information Systems Development, 34th International Conference on Information Systems, Milan, Italy, pp.1-11. 2013
59. Söderman, J. How to be a good ombudsman. *11th Int- Ombudsman Institute World Conference*, Occasional paper nr. 80, https://www.theioi.org/publications/occasional-papers-1979-2008 (accessed 12.09.2019). 2008.
60. Thayer, R. H., and Royce, W. W. Software Systems Engineering. *IEEE System and Software Requirements Engineering.* Los Alamos, California. 1990.
61. Thakurta, R, and Ahlemann, F. Understanding Requirements Volatility in Software Projects – An Empirical Investigation of Volatility Awareness, Management Approaches and their Applicability, Proceedings of the *43rd Hawaii International Conference on System Sciences,* pp. 1-10. 2010.
62. Tjong, F. *Avoiding Ambiguity in Requirements Specifications,* PhD Thesis, Univ. of Nottingham, UK. 2008.
63. Zowghi, D., and Nurmuliani, N. A Study of the Impact of Requirements Volatility on Software Project Performance, *9th Asia-Pacific Software Engineering Conference*, Gold Coast, Australia. 2002.
64. Zanini, M.T. and Migueles, C. Building trust in a high-Power Distance context: the role of the perception of integrity in shared leadership, *Academy of Management Conference*, Chicago, August. 2018.
65. ISO/IEC/IEEE 26511:2018Systems and software engineering - Requirements for managers of information for users of systems, software, and services, https://www.iso.org/standard/70879.html (access 31.05.2022).
66. CMMI Requirements Development (RD), https://www.cmmi.co.uk/cmmi/RD.htm (accessed 31.05.2022).
67. SWEBOK V3, Guide to the Engineering Body of Knowledge, available at https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf, 31.05.2022).
68. Requirements Working Group, INCOSE, Guide for writing requirements, 2012