Critical Software Processes Tailoring and Very Small Entities (VSE): A Literature Review

Diniz, G. H [1]; Ambrosio, A. M. [2]; Lahoz, C. H. N. [3]

[1,2]National Institute for Space Research (INPE), São José dos Campos, SP, Brazil
[3]Vale do Paraíba University (UNIVAP), São José dos Campos, SP, Brazil

Abstract *— Aligned with the worldwide trend of developing using small teams, most of the critical software has been developed by small organizations, which demand a particular attention to establishment of process approaches suitable for them. Although there are many software standards, the majority of them do not specifically aim the needs of organizations such as Very Small Entities (VSEs). Standard processes are usually tailored based only on the software criticality, and so researches have been conducted about the effects of project characteristics on software processes and how to use them for processes tailoring. For systematically determining software processes, they need to be defined according to projects' characteristics and objectives. This work provides a review on project evaluation, process profiles, identification of factors which impact software processes, tools for classifying VSE software projects subject to processes tailoring. Results show the review organized in topics that surround the research objective, presenting the critical software and VSE scenario. Critical software and VSE standards comparison indicated that these processes present similarities, representing opportunities to use them complementarily. Accordingly, the projects' criteria selection is a means to support the understanding of the influence factors for critical software projects in VSE context and, furthermore, to develop a notion on adequate tailoring. A systematic approach can be helpful in the VSE context. Suggestions for future research are proposed based on the results.*

Keywords*— *Software, evaluation, process, tailoring, VSE.

I. INTRODUCTION

At present the objective of obtaining quality software products has led to the need of carrying out good software processes selection, for which a systematic method is an important aspect. This work explores the fundamental elements of the process selection, tailoring criteria and project evaluation.

Software development can be difficult and resource-consuming (Wiegers, et al., 2013). Therefore, managing its development activities in an organization is usually accomplished by introducing techniques, tools, best practices and process models (Naur, et al., 1969). According to SEI (2010), organizations should direct their efforts to three critical dimensions of the software development process: people; procedures and methods; tools and equipment.

For the three critical dimensions, standardization is a significant instrument for increasing quality and communication among stakeholders during conception, planning and implementation of projects, while it also helps to reduce risks and costs associated, making business more profitable as less time is spent on non-productive work (Yilmaz, et al., 2016).

Because product quality improvement is typically achieved by improving their processes, standards have been published by committees, international technical entities or regulatory agencies to influence software development through guidelines for processes and products considering their associated risks (Munch, et al., 2012). Software processes have the potential to be highly complex (Clarke, et al., 2016) and may be subdivided into tasks and activities. A **process** is a set of related activities performed for a particular purpose or outcome (like develop and maintain software products); a **task** is an action with inputs and outputs, which may be a requirement (must), recommendation (should) or permission (may); and an **activity** is a set of tasks (ISO, 2015).

Projects tailor software standard processes to develop their own defined processes, which account for the unique characteristics of the project. This tailored process is referred to in the Capability Maturity Model (CMM) as the project's defined software process, comprising a coherent, integrated set of distinct software engineering and management processes (SEI, 2010).

Standard processes typically cannot be used without customization, a tailoring (Ginsberg, et al., 1995), and although the need to tailor software processes to specific project requirements is widely accepted, the way of doing it is frequently unclear (Kalus, et al., 2013). The European Cooperation for Space Standardisation (ECSS) (2017a) and National Aeronautics and Space Administration (NASA) (2017) recommend tailoring their standard processes based on the software criticality level (ECSS, 2017b), and it is under responsibility of each organization to eventually select other criteria to indicate the risk that the project is prepared to take and the extent to which the processes are made applicable. Research (Kalus, et al., 2013) has been conducted on the effects of project factors for the resulting software process and how to use this knowledge to choose the ones to be considered for processes tailoring.

Process tailoring needs to be performed in a thoughtful and disciplined manner. Interpreting the standard terminology (i.e. documents, processes, activities, tasks, roles and artifacts) in such way that each organization understands is not a trivial task. Tailoring the selected processes to the project specificities requires criteria for evaluating the relevance of the activities to the overall project needs. The subset of applicable processes selected through project classification can vary, depending mainly on factors such as type, size, complexity and phase of the project being addressed.

Since the set of all possible software is very large, a set of processes suitable for use by all potential organizations and projects would be either excessively general or complex, and also difficult to apply. Consequently, different initiatives have taken place considering the software environments.

The objective of this research work is to perform a literature review on approaches for process selection applicable to critical software projects in Very Small Entities (VSE). The literature review aims to compare the VSE practices to the more consolidated critical software literature, and to explore the systems complexity environment where both intersect, by reviewing concepts related to identification of specific criteria that influence software projects and their implications on processes considering the typical resources limitations of VSE.

II. METHOD

Research method comprises bibliographic research with qualitative analysis for background and studies review. Background review comprises two topics: VSE and software criticality. And, complementarily, studies review can be grouped also in two topics: critical software processes tailoring and software process in VSE. The research outline is shown in *Fig. 1*.

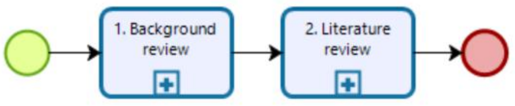

Fig. 1: Research outline

2.1 Background review

The *Background review* provides the foundation to situate the context to which this work has been addressed, comprising two topics: VSE and software criticality.

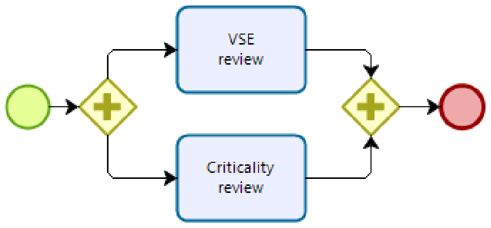

Fig. 2: Background review

In this work, the VSE contents come from ISO/IEC 29110; and, for software criticality, from ECSS material, which is based on ISO 9000 family of documents (which addresses various aspects of quality management), as well as on *ISO/IEC/IEEE 12207 - Systems and software engineering – Software life cycle processes* (an international standard for software lifecycle processes) and *ISO/IEC 15504 - Information technology – Process assessment*, also termed *Software Process Improvement and Capability dEtermination (SPICE)* (derived from ISO/IEC 12207 and from maturity models like CMM). The main sources of standards material are:

- ECSS-E-ST-40C. Space Engineering - Software.
- ECSS-Q-ST-80C-Rev.1. Space product assurance - Software product assurance.
- ECSS-Q-HB-80-02-Part1A. Space product assurance – Software process assessment and improvement – Part 1: Framework.
- ECSS-Q-HB-80-02-Part2A. Space product assurance – Software process assessment and improvement – Part 2: Assessor Instrument.
- ISO/IEC. (2011a). ISO/IEC 29110-4-1 - Software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 4-1: Profile specifications: Generic profile group.
- ISO/IEC. (2011). ISO/IEC TR 29110-5-1-2. Software Engineering - Lifecycle Profiles for Very Small Entities (VSEs) - Part 5-1-2: Management and engineering guide: Generic Profile Group: Basic Profile.

2.2 Studies review

The studies review comprises identification and a synthesis of the papers with greater intersection with the topics of interest. According to Pai et al. (2004), the core five steps of a systematic review process are: (i) review question formulation; (ii) a comprehensive search; (iii) studies evaluation; (iv) results synthesis; and (v) results analysis. Fig. 2 presents the systematic review process.

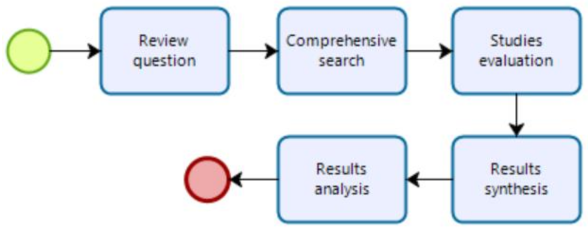

Fig.2: Studies review

Because systematic reviews are time-consuming, when a decision to conduct a review is made, the first step was to formulate a clear, focused question and prepare a protocol. The PICO (Population/Problem, Intervention, Control/ Comparison and Outcome) framework is often used to identify the four critical parts of a well-built research question. The protocol should specify the population (or the topic of interest), the intervention (or exposure) being evaluated, the comparison intervention (if applicable), and the outcome. (Higgins JPT, 2011)A focused question will help in conducting more specific searches of databases, and also in creating unambiguous criteria for selecting studies. TABLE 1 shows the PICO framework for this review.

Table 1: PICO framework

	Description	Keywords
Population/ Problem	Software processes tailoring	Process, tailoring
Intervention	Critical software processes in VSE	critical, small entities
Control/ Comparison	ECSS system + ISO/IEC 29110	ECSS, 29110
Outcome	Identification of initiatives on processes tailoring for critical software in VSE	-

Based on TABLE 1 contents, the research question was: "What initiatives have been proposed for critical software processes tailoring in very small organizations?"

The search was performed on the selected databases: "Science Direct", at www.sciencedirect.com and "IEE Xplore", at https://ieeexplore.ieee.org/Xplore, conducting searches using multiple, alternative terms combined with the Boolean operators "AND" and "OR" for the keywords from the PICO set. Using "OR" for each keyword explodes the search and make it highly sensitive (likely to yield thousands of results), while using AND dramatically narrows the search.

The search strings, defined using combinations of the keywords and extended by adding the term "software", were used in the title, abstract and keywords fields, focusing on exploring works in the field of software process published since January/2000, including journals and conference proceedings.

III. RESULTS AND DISCUSSION

3.1 Background review

The first topic presents the VSE fundamentals and related practices. The second topic presents concepts about software criticality based on the perspective presented by ISO and improved by ECSS.

3.1.1 Very Small Entities (VSE)

The term "very small entity" (VSE) has been defined by ISO/IEC JTC1/SC7 Working Group 24 and subsequently adopted for use in the ISO/IEC 29110 process lifecycle standard as being "an enterprise, organization, department or project having up to 25 people" (ISO/IEC, 2011b). They have important significance in contributing with valuable products and services as they represent a large majority of enterprises worldwide (Moll, 2013). Because of their size, VSEs differ from larger organizations, with most of the management processes performed more informally and less documented (O'Connor, et al., 2010).

Even though most of the space software has been developed by small groups (Lahoz, et al., 2015), most of the software development Standards do not specifically aim the needs of small enterprises (O'Connor, et al., 2010), a scenario that demands particular attention with establishment of process approaches suitable for small organizations.

For many small software companies, it is a major challenge implementing controls and structures to properly manage their software processes (Larrucea, et al., 2016), and the lack of formalism in their processes may have negative consequences, such as missing important activities and tasks, or having limited ways to demonstrate their quality and be recognized in their domain, consequently they may be put aside from projects (Rodríguez-Dapena, et al., 2017).

ISO/IEC 29110 series of International Standards and Technical Reports objectives to assist and encourage very small software organizations in assessing and improving their software processes (O'Connor, et al., 2011a). Their approach (O'Connor, et al., 2011b) relies on the concept of ISO standardized profiles (SP) making use of pre-existing international standards, such as the software life cycle standard ISO/IEC/IEEE 12207 and the documentation standard ISO/IEC/IEEE 15289. Relevant elements from those standards have been selected to compose subsets of applicable processes, referred to as VSE profiles, targeted to specific project types. The profiles are gathered in profile groups according to the classification of software projects, proposing a progressive approach that addresses most VSEs not involved in critical software development.

ISO/IEC (ISO/IEC, 2016) International Standards and Technical Reports were developed according to the characteristics and needs of VSEs. Beyond size, other factors may affect a profile preparation or selection, such as: Business Models (commercial, contracting, in-house development, etc.); Situational factors (such as criticality, uncertainty environment, etc.); and Risk Levels (Laporte, et al., 2008). Producing one profile for each combination of these factors would result in an unmanageable set of profiles. Consequently, VSE's profiles are grouped in such a way to be applicable to more than one category.

A profile group is composed by elements related by composition of processes (i.e. activities, tasks), by capability level, or both (O'Connor, et al., 2010). The Generic profile group, chosen as reference for this work, comprises a collection of four profiles (Entry, Basic, Intermediate, Advanced), proposing a progressive approach to satisfying most of VSEs as it does not imply any specific domain (ISO/IEC, 2011a).

The four profiles from the Generic profile group are:

- <u>Entry Profile</u>: targets VSEs working on small projects (e.g. at most six person-months effort) and for start-up VSEs that do not have significant experience with large software development projects, and so do not attract contract jobs from larger software firms.
- <u>Basic Profile</u>: describes external or internal projects of a single application by a single team with no special risk or situational factors. To use this Profile, the VSE needs to fulfil basic entry conditions, e.g. documented project statement, feasibility analysis performed, training personnel and infrastructure available.
- <u>Intermediate Profile</u>: describes the management of more than one project in parallel with more than one work team, comprising processes to identify opportunities, evaluate all agreements or requests from customers for fit with organisational goals and resources, obtain and provide necessary resources to perform, monitor and evaluate all projects.
- <u>Advanced Profile</u>: targeted at VSEs wanting to sustain and grow as an independent competitive system and/or software development business. For that it contains processes to move software in an orderly, planned manner into the operational status such that the system is functional in the operational environment, appropriately handle replaced or retired elements, and to attends critical needs (e.g. per an agreement, per organisational policy, or for environmental, safety, and security aspects).

3.1.2 Criticality

IEEE (2002) describes software "whose failure could have an impact on safety, or could cause large financial or social loss" as critical. According to (ECSS, 2017b), if a software error has the potential to cause human lives loss or other major or catastrophic consequences, the software is designated as Safety Critical Software (SCS).

Critical software can be found in several diverse standard regulated environments, such as: Aerospace, Aeronautics, Medical, Railway and Nuclear. Software developments in these different areas must considerer specific factors such as type of software product, role of software in the system, size of the system and level of risk. Software is found from top system functions down to firmware, including safety and mission critical functions, presenting different types of risks according to the variety of possible consequences of a failure in their different environments. (Marques, 2016)

Critical software main reference of processes is ISO/IEC 15504 (superseded by ISO 330xx series), also known as SPICE (Software Process Improvement and Capability dEtermination), which is based on the Process Reference Model (PRM) from ISO/IEC 12207. ECSS definitions are considered for the development of the present research, because the contents of the model defined in ECSS-Q-HB-80-02, called SPICE for Space (S4S), extend SPICE by adding processes and indicators related to specific RAMS (Reliability, Availability, Maintainability and Safety) requirements (*Fig. 3*) from ECSS standards, to ensure that software is developed to perform properly and safely, meeting the project's quality objectives.

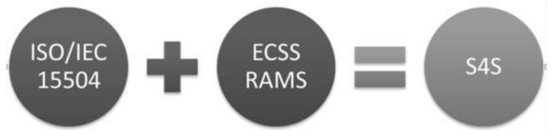

Fig.3: S4S contents

ECSS standards present the criticality definition based on the severity of failures consequences (ECSS, 2009), as described in TABLE 2, where, for each software product type described in the right column, a correspondent criticality category is assigned in the left column, based on the highest criticality of the functions implemented by the software and the existing system compensating provisions. According to this classification, software of criticality category A, B or C is defined as critical; consequently category D denotes non-critical software (ECSS, 2017a).

Table 2: Software criticality categories definition

Criticality category	Definition
A	Software involved in category I functions AND: no compensating provisions exist
	Software included in compensating provisions for category I functions
B	Software involved in category I functions AND: at least one of the following compensating provisions is available: - A hardware implementation - A software implementation; this software shall be classified as criticality A - An operational procedure
	Software involved in category II functions AND: no compensating provisions exist
	Software included in compensating provisions for category II functions
C	Software involved in category II functions AND: at least one of the following compensating provisions is available: - A hardware implementation - A software implementation; this software shall be classified as criticality B - An operational procedure
	Software involved in category III functions AND: no compensating provisions exist
	Software included in compensating provisions for category III functions
D	Software involved in category III functions AND: at least one of the following compensating provisions is available: - A hardware implementation - A software implementation; this software shall be classified as criticality C - An operational procedure
	Software involved in category IV functions AND: no compensating provisions exist

Source: Adapted from (ECSS, 2017a)

The software criticality category (A, B, C, D) is assigned based on safety and dependability aspects, considering the severity of the eventual failure of the most critical function it implements (ECSS, 2017b) as shown in TABLE 3.

Table 3: Function criticality description

Severity	Function criticality	Criteria
Catastrophic (Level 1)	I	A function that if not or incorrectly performed, or whose anomalous behavior, can cause one or more feared events resulting in catastrophic consequences
Critical (Level 2)	II	A function that if not or incorrectly performed, or whose anomalous behavior, can cause one or more feared events resulting in critical consequences
Major (Level 3)	III	A function that if not or incorrectly performed, or whose anomalous behavior, can cause one or more feared events resulting in major consequences
Minor or Negligible (Level 4)	IV	A function that if not or incorrectly performed, or whose anomalous behavior, can cause one or more feared events resulting in minor or negligible consequences

Source: adapted from (ECSS, 2017b)

3.2 Studies review

The number of publications identified by using the presented criteria is shown in TABLE 4.

Table 1: Search results – Reference date: 07/Nov/2019

Search string	Science Direct	IEEE Xplore
software AND process AND small entities	267	137
software AND ECSS OR 29110	13	68
software AND small entities AND tailoring AND process	54	10
software AND critical AND small entities	138	36
Total	**472**	**251**

As TABLE 4 shows, the initial search run on Science Direct returned 472 papers and on IEEE Xplore returned 251 papers in total. After a review of titles, duplicate and irrelevant papers were removed and the abstracts review resulted in the selection of 30 publications for further analysis.

After reading completely the selected publications, the data extracted was summarized in this section, divided into two main topics: Critical Software Process Tailoring and Software Processes in Small Entities.

The first topic presents the critical software processes tailoring fundamentals and current limitations analyzed through an historical perspective and according to topics of interest for this research. The second topic presents methodologies and best practices related to software processes in small entities.

3.2.1 Critical Software Processes Tailoring

As software development organizations' needs may vary according to multiple factors, any process model to be implemented should be capable of dealing with their differences. Although comprehensive top-down prescriptive models such as CMMI and ISO/IEC 15504 (SPICE) have been used (Gorschek, et al., 2006), literature reports that these so-called heavy models and their evaluation methods are considered expensive by small organizations (Cater-Steel, 2004) (Laryd, et al., 2000) (Johnson, et al., 1997) (Kelly, et al., 1999) (Villalón, et al., 2002) (Schoeffel, et al., 2015), which is related to these models not being extensively deployed and their influence in software industry remains more at a theoretical level (Laporte, et al., 2015).

SPICE initially had several limitations. Routa et al. (2007) reviewed the evolution of the Standard and the parallel achievements of the SPICE Project and the standardization effort in advancing the state of the art in process assessment and improvement. Their work presents the significant advances in understanding of the nature of process capability and its evaluation that have been made possible through SPICE, although it does not present the processes.

Because software malfunctions due to poorly written requirements may cause financial loss, Véras et al. (2015) proposed a benchmark, with 3 checklists to assess the quality of space software specifications, providing a simple and effective way to identify weaknesses and maturity degree of requirements documents. The checklists were applied to telecommand and telemetry software in the Requirements Definition phase.

In (Bujok, et al., 2016) standards from different domains are mapped revealing the presence of common requirements and the potential for the identification of a "Common Core" to be used as a unified framework, addressing the need to comply with multiple international standards regulations in safety critical domains.

Studies have proposed criteria other than criticality for tailoring development processes, mainly related to the variables used for software effort estimation (Kalus, et al., 2013), also demonstrating the correlation between software quality metrics and aspects such as team skill (Wang, et al., 2006).

Kalus & Kuhrmann (2013) present a Systematic Literature Review about criteria for software process tailoring, comprising the dependencies between different criteria and their influence in the software process, concluding that the consequences of the criteria usage remain abstract and are to be interpreted on a project-per-project basis. Their set of 49 project factors that influence software processes tailoring is organized and presented, comprising the names and brief descriptions of project factors categorized in: team - characteristics of the people involved in the project; internal environment - organizational aspects of the project's entity; external environment - context where the project takes place; objectiveness - product related features.

Pedreira, et al. (2007) conducted a study about the current practice in software process tailoring, concluding that existing approaches for process tailoring are defined in specific environments, and that a general framework should be developed. The idea of a generic systematic framework is corroborated by (Xu, et al., 2008), that present an investigation about software projects challenges based on interviews, concluding that tailoring affects the software process and environment, and that excessive tailoring can undermine process repeatability and consistency.

Estimation techniques may be applied for the definition of project processes. The main methods for estimation are based either on algorithmic estimation models or on expert estimation techniques, commonly used for appraising software development effort (Jørgensen, et al., 2007). Expert estimation is considered a light process, involving a small number of documentation, as expert estimation relies on expertise to subjectively assess the involved factors, using experts "intuition" alone or combined with historical data and/or checklists, when available, to make estimates (Jørgensen, 2004).

Software estimation approaches lack studies supporting them in detail, though the usual checklist consists of the typical activities (e.g., requirements management, design, prototype, testing, documentation etc.) in a software project (Usman, et al., 2018).

Jørgensen & Molokken (2003) proposed a preliminary checklist, to be customized to include only relevant issues, structured on a project management framework considering scopes comprehending since the typical estimation activity until different project phases. In the VSE critical software context, it may not be feasible to use long checklists covering aspects beyond the typical estimation.

3.2.2 Software Processes in Small Entities

Given the limitations in terms of people and money that small organizations have due to their size, they face many challenges in running process assessments (Basri, 2011). Considering this, the assessment method proposed by Pino et al. (2010) sets out the elements needed to assist with diagnosing the process step-by-step in small organizations developing non-critical software while seeking to make the assessment application economically feasible in terms of resources and time.

VSE usually consider that SPI frameworks: are either too expensive to deploy or do not take organizations' specific needs into consideration. Pettersson et al. (2008) presents a light weight assessment and improvement planning (iFLAP) that enables practitioners to base improvement efforts on the issues that are the most critical for the specific organization. Their packaged improvement framework, containing both assessment and improvement planning capabilities, was applied to non-critical software case studies, without presenting the software processes involved.

Evidence has shown that the majority of very small organizations are not adopting existing standards and best practice models because they perceive them as developed by and orientated towards large organizations, therefore pointing out the relevance of the number of people involved in a software project (O'Connor, et al., 2009).

Zarour, et al. (2015) analyzed the reasons behind small organizations failures in Software Process Improvement (SPI). They investigated, through a literature review, the pieces of knowledge and their frequencies that form the best practices for the successful design and implementation of lightweight software process models. They do not present the software processes, but classify a set of 38 best practices into five main categories, covering all aspects of the assessment, namely: assessment method, supportive tool, procedure, documentation, and users.

Yousefal-Tarawneh et al. (2011) proposed the use of XP as software development model and CMMI as SPI model because, SPI traditional models were developed to help large and very large organizations. They present their development process improvement framework, which does not consider Safety Critical Software aspects, comprising the method's stages for developing suitable software by using CMMI-DEV V1.2.

Sanchez-Gordon et al. (2017) reviewed relevant standards, such as ISO/IEC 29110, ISO 10018, OMG Essence and ISO 33014, to develop a framework to integrate human factors in software processes. Their proposed approach integrates international standards in a comprehensive, yet practical, framework addressing the human factors of small companies developing non-critical software. And Laporte & O'Connor (2017) presented an overview of eight implementations process improvement standards and guides for non-critical software in VSE, with a four-stage roadmap to support process improvement activities using ISO/IEC 29110.

Laporte, O'Connor, & Paucar (2015) present seven case studies involving pilot usage of ISO/IEC 29110, comprising a project classification into three categories (small, medium and large), based on characteristics such as duration, team size, number of engineering specialties and engineering fees. This study demonstrated that it is possible to plan and execute non-critical software projects in small settings using proven practices to significantly reduce the number of discrepancies.

Rodríguez-Dapena & Lohier (2017) proposed a step-wise approach to participate in space projects in a feasible way, adding processes from ECSS-Q-HB-80 (S4S) and capability from ISO/IEC 15504 to one of the profiles presented in ISO/IEC 29110. This approach considers different subsets of processes and levels of process capability, but it is only applicable for software criticalities levels D (non-critical) and C (low criticality).

IV. CONCLUSION

The purpose of this review was to outlook the trends in critical software development studies in VSE within the past twenty years, identifying which practices have been applied to adapt standards and models to software projects.

Many studies have been proposed to describe process tailoring for software development. The reviewed publications make evident that the tailoring criteria must regard the project specificities to define what processes need to be performed. Furthermore, the methods to select criteria and processes are varied and the development organization is in charge of defining how to implement.

From the research reviewed, it is clear that standard processes are very immersed and widely practiced throughout in development organizations. Along with this, it is also clear that the field of processes tailoring is varied and continues to be studied and analyzed in order to most benefit the product quality. Critical software process tailoring in VSE is still an open issue, though, as the results show scarce research for critical software processes considering the VSE context. This topic is very important as at its center is a concern with helping VSE become better and demonstrate the quality of their processes and products, consequently suggesting the potential of VSE processes within critical software projects scope.

Critical software and VSE standards comparison indicated that these processes present similarities, representing opportunities to use them complementarily. Accordingly, the projects' criteria selection is a means to support the understanding of the influence factors for critical software projects in VSE context and, furthermore, to develop a notion on adequate tailoring.

A systematic approach for process tailoring can be helpful in the VSE context, where team-based expert estimation is usual, there is lack of documentation and new team members might not be aware of all activities and factors that should be accounted for during estimation. Frequently process tailoring is informally performed in VSE and the lack of a documented approach is also likely to result in the loss of useful experience from previous projects.

Further studies are necessary on the use of adequate profiles, comprising simplified and flexible sets of processes according to each software project evaluation, providing evidence on their feasibility with evaluation of their completeness, applicability and usability for critical software in VSE.

REFERENCES

[1] Basri, S. O. (2011). A study of software development team dynamics in SPI. Systems, Software and Services Process Improvement (EuroSPI 2011), vol. 172, 143-154. Springer-Verlag.

[2] Bujok, A. B., MacMahon, S. T., Grant, P., Whelan, D., Rickard, W. J., & McCaffery, F. (2017). Approach to the development of a Unified Framework for Safety Critical Software Development. Computer Standards & Interfaces.

[3] Bujok, A. B., MacMahon, S. T., McCaffery, F., Dick Whelan2, B. M., & Rickard, W. J. (2016). Safety Critical Software Development – Extending Quality Management System Practices to Achieve Compliance with Regulatory Requirements. International Conference on Software Process Improvement and Capability Determination.

[4] Cater-Steel, A. (Dec de 2004). PhD Thesis. An evaluation of software development practice and assessment-based process improvement in small software development firms. Queensland, Australia: Griffith University.

[5] Clarke, P., O'Connor, R., & Leavy, B. (2016). A complexity theory viewpoint on the software development process and situational context. Proceedings of the International Conference on Software and Systems Process (ICSSP '16). New York, NY, USA: ACM.

[6] ECSS. (2009). ECSS-Q-ST-30-02C. Space product assurance - Failure modes, effects (and criticality) analysis (FMEA/FMECA). ECSS.

[7] ECSS. (2009a). ECSS-E-ST-40C. Space Engineering - Software. Noordwijk, The Netherlands: ESA Requirements and Standards Division.

[8] ECSS. (2010a). ECSS-Q-HB-80-02-Part1A. Space product assurance – Software process assessment and improvement – Part 1: Framework. Noordwijk, The Netherlands: ESA Requirements and Standards Division.

[9] ECSS. (2010b). ECSS-Q-HB-80-02-Part2A. Space product assurance – Software process assessment and improvement – Part 2: Assessor Instrument. Noordwijk, The Netherlands: ESA Requirements and Standards Division.

[10] ECSS. (2017a). ECSS-Q-ST-80C-Rev.1. Space product assurance - Software product assurance. Noordwij, The Netherlands: ESA Requirements and Standards Division.

[11] ECSS. (15 de Feb de 2017b). ECSS-Q-ST-30C-Rev.1. Space product assurance - Dependability. ECSS.

[12] Ginsberg, M., & Quinn, L. (November de 1995). Process tailoring and the software capability maturity model. Technical Report CMU/SEI-94-TR-024. Pittsburgh, PA, USA: Software Engineering Institute.

[13] Gorschek, T., & Wohlin, C. (11 de 2006). Requirements Abstraction Model. Requirements engineering journal, pp. 79-101.

[14] Higgins JPT, Green S (editors). *Cochrane Handbook for Systematic Reviews of Interventions* Version 5.1.0 [updated March 2011]. The Cochrane Collaboration, 2011. Available from www.handbook.cochrane.org

[15] IEEE. (2002). IEEE Std 610.12-1990(R2002) . IEEE Standard Glossary of Software Engineering Terminology. USA.

[16] ISO. (2015). ISO 9000:2015. Quality management systems — Fundamentals and vocabulary. ISO.

[17] ISO/IEC. (2011a). ISO/IEC 29110-4-1 - Software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 4-1: Profile specifications: Generic profile group. Geneva - Switzerland: ISO.

[18] ISO/IEC. (2011b). ISO/IEC TR 29110-5-1-2 - Software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 5-1-2: Management and engineering guide:

Generic profile group: Basic profile. Geneva - Switzerland: ISO.

[19] Johnson, D. L., & Brodman, J. G. (vol 8 de 1997). Tailoring the CMM for small businesses, small organizations, and small projects. Software Process Newsletter - IEEE Computer Society.

[20] Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. Journal of Systems and Software , pp. 37–60.

[21] Jørgensen, M., & Molokken, K. (2003). A preliminary checklist for software cost management Quality Software. 2003 Proceedings Third International Conference, 134–140. IEEE.

[22] Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering.

[23] Kalus, G., & Kuhrmann, M. (2013). Criteria for Software Process Tailoring: A Systematic Review. ICSSP 2013 Proceedings of the 2013 International Conference on Software and System Process, 171-180. New York: ACM International Conference Proceeding Series.

[24] Kelly, D. P., & Culleton, B. (1999). Process improvement for small organizations. Computer, 32(10), 41-47. IEEE.

[25] Lahoz, C. H., Richter, S., & Rico, D. E. (2015). Rapid Software Process Assessment in the space domain for Very Small Entities. ESA Software Product Assurance Workshop. Frascati, Italy.

[26] Laporte, C. Y., & O’Connor, R. V. (2017). Software Process Improvement Standards and Guides for Very Small Organizations: An Overview of Eight Implementations. CrossTalk - The Journal of Defense Software Engineering, 30(3).

[27] Laporte, C. Y., O’Connor, R. V., & Paucar, L. H. (2015). Software Engineering Standards and Guides for Very Small Entities: Implementation in two start-ups. 10th International Conference on Evolution of Novel Approaches to Software . Barcelona, Spain.

[28] Laporte, C. Y., O'Connor, R. V., & Paucar, L. H. (2016). The Implementation of ISO/IEC 29110 Software Engineering Standards and Guides in Very Small Entities. Evaluation of Novel Approaches to Software Engineering. ENASE 2015. Communications in Computer and Information Science, 599, 162-179. Barcelona, Spain: Springer.

[29] Laporte, C., Alexandre, S., & O’Connor, R. (2008). A Software Engineering Lifecycle Standard for Very Small Enterprises. EuroSPI2008, 129 - 141. Springer-Verlag.

[30] Larrucea, X., O’Connor, R. V., Colomo-Palacios, R., & Laporte, C. Y. (March-April de 2016). Software Process Improvement in Very Small Organizations. IEEE Software, pp. 85 - 89.

[31] Laryd, A., & Orci, T. (2000). First Argentine Symposium on Software Engineering. 133-149. Tandil: ASSE.

[32] Marques, J. C. (2016). MACRE-SAR: An Agile Model for Software Requirements Specification in Regulated Environments. Doctor Thesis.

[33] Moll, R. (2013). A bird’s eye view of SMEs and risk management. ISO Focus+. Geneva, Switzerland: International Organization for Standardization.

[34] Munch, J., Armbrunt, O., Kowalczyk, M., & Soto, M. (2012). Software Process Definition and Management. Berlim: Springer-Verlag.

[35] NASA. (2017). NASA Systems Engineering Handbook. NASA SP-2016-6105 Rev2. National Aeronautics and Space Administration.

[36] NASA Office of Chief Engineer. (2009). NASA Study on Flight Software Complexity. Pasadena, CA, USA: Systems and Software Division - Jet Propulsion Laboratory - California Institute of Technology.

[37] Naur, P., & Randell, B. (1969). Software Engineering: a report on a conference sponsored by the NATO Science Comitee. Brussels: NATO.

[38] O’Connor, R., & Laporte, C. (21-23 de June de 2010). Towards the Provision of Assistance for Very Small Entities in Deploying Software Lifecycle Standards. 11th International Conference on Product Focused Software Development and Process Improvement (Profes2010).

[39] O'Connor, R., & Coleman, G. (2009). Ignoring ‘Best Practice': Why Irish Software SMEs are rejecting CMMI and ISO 9000. Australasian Journal of Information Systems, Vol. 16,(No. 1).

[40] O'Connor, R., & Laporte, C. (2011a). Deploying Lifecycle profiles for Very Small Entities: An Early Stage Industry View. Proceedings of 11th International SPICE Conference on Process Improvement and Capability dEtermination, CCIS Vol. 155. Springer-Verlag.

[41] O'Connor, R., & Laporte, C. (2011b). Using ISO/IEC 29110 to Harness Process Improvement in Very Small Entities, Workshop on SPI in SMEs. 18th European Software Process Improvement Conference, CCIS Vol. 172. Springer-Verlag.

[42] O'Connor, R., Basri, S., & Coleman, G. (2010). Exploring Managerial Commitment towards SPI in Small and Very Small Enterprises. Systems, Software and Services Process Improvement, CCIS Vol. 99, 268-278. Springer-Verlag.

[43] Pai, M., McCulloch, M., Gorman, J. D., Pai, N., Enanoria, W., Kennedy, G., . . . Colford, J. M. (2004). Systematic reviews and meta-analyses: an illustrated, step-by-step guide. Natl. Med. J. India, pp. 89-95.

[44] Pedreira, O., Piattini, M., Luaces, M. R., & Brisaboa, N. R. (2007). A Systematic Review of Software Process Tailoring. ACM SIGSOFT Software Engineering Notes, Volume 32 (Issue 3). New York: ACM .

[45] Pettersson, F., Ivarsson, M., Gorschek, T., & P.Öhman. (2008). A practitioner's guide to light weight software process assessment and improvement planning. Journal of Systems and Software.

[46] Pino, F. J., Calvache, C. J., Garcia, F., & Piattini, M. (2010). Assessment methodology for software process improvement in small organizations. Information and Software Technology, 52(10), 1044-1061.

[47] Rodríguez-Dapena, P., & Lohier, P. (2017). How small organizations could participate in Space projects.

[48] Rodríguez-Dapena, P., & Lohier, P. (2017). How small organizations could participate in Space projects.

[49] Routa, T. P., Emam, K. E., Fusani, M., Goldenson, D., & Jung, H.-W. (2007). SPICE in retrospect: Developing a standard for process assessment. Journal of Systems and Software.

[50] Schoeffel, P., & Benitti, F. B. (2015). Factors of influence in software process improvement: A comparative survey between micro and small enterprises (MSE) and medium and large enterprises (MLE). IEEE Latin America Transactions, pp. 1634-1643.

[51] SEI. (Nov de 2010). CMMI-DEV, V1.3. CMMI for Development, Version 1.3. Pittsburgh, PA, USA: Carnegie Mellon University.

[52] Usman, M., Petersen, K., Börstler, J., & Neto, P. S. (2018). Developing and using checklists to improve software effort estimation: A multi-case study. The Journal of Systems and Software, 146. Elsevir Inc.

[53] Véras, P. C., Villani, E., Ambrosio, A. M., Vieira, M., & Madeira, H. (2015). A benchmarking process to assess software requirements documentation for space applications. The Journal of Systems and Software.

[54] Villalón, J. A., Cuevas, A. G., SanFeliu, G. T., DeAmescua, S. A., García, S. L., & Pérez, C. M. (10 de 2002). Experiences in the application of software process improvement in SMES. Software Quality Journal, pp. 261-273.

[55] Wang, Z.-J., Zhan, D.-C., & Xu, X.-F. (May de 2006). STCIM: a dynamic granularity oriented and stability based component identification method. ACM SIGSOFT Software Engineering Notes, 31, 3, 1-14. New York, NY, USA: ACM.

[56] Wiegers, K., & Beatty, J. (2013). Software Requirements 3. Washington, USA: Microsoft Press.

[57] Xu, P., & Ramesh, B. (2008). Using Process Tailoring to Manage Software Development Challenges. IT Professional, 10, 39-45.

[58] Yilmaz, M., O'Connor, R. V., & Clarke, P. (2016). Effective Social Productivity Measurements during Software Development - An Empirical Study. International Journal of Software Engineering and Knowledge Engineering.

[59] Yousefal-Tarawneh, M., Abdullah, M. S., & Ali, A. B. (2011). A proposed methodology for establishing software process development improvement for small software development firms. Procedia Computer Science.

[60] Zarour, M., Abran, A., Desharnais, J., & Alarifi, A. (Nov de 2015). An investigation into the best practices for the successful design and implementation of lightweight software process assessment methods: A systematic literature review. The Journal of Systems and Software 101, pp. 180-192.