



MINISTÉRIO DA  
CIÊNCIA, TECNOLOGIA,  
INOVAÇÕES E COMUNICAÇÕES



sid.inpe.br/mtc-m21c/2020/01.24.13.23-TDI

## A CRITICAL SOFTWARE PROCESS SELECTION FOR VERY SMALL ENTITIES (VSE)

Gledson Hernandes Diniz

Master's Dissertation of the Graduate Course in Engineering and Space Technology/Space Systems Engineering and Management, guided by Drs. Ana Maria Ambrosio, and Carlos Henrique Netto Lahoz, approved in November 14, 2019.

URL of the original document:

<<http://urlib.net/8JMKD3MGP3W34R/3UQR2LE>>

INPE  
São José dos Campos  
2019

**PUBLISHED BY:**

Instituto Nacional de Pesquisas Espaciais - INPE  
Gabinete do Diretor (GBDIR)  
Serviço de Informação e Documentação (SESID)  
CEP 12.227-010  
São José dos Campos - SP - Brasil  
Tel.:(012) 3208-6923/7348  
E-mail: pubtc@inpe.br

**BOARD OF PUBLISHING AND PRESERVATION OF INPE  
INTELLECTUAL PRODUCTION - CEPPII (PORTARIA Nº  
176/2018/SEI-INPE):****Chairperson:**

Dra. Marley Cavalcante de Lima Moscati - Centro de Previsão de Tempo e Estudos  
Climáticos (CGCPT)

**Members:**

Dra. Carina Barros Mello - Coordenação de Laboratórios Associados (COCTE)  
Dr. Alisson Dal Lago - Coordenação-Geral de Ciências Espaciais e Atmosféricas  
(CGCEA)  
Dr. Evandro Albiach Branco - Centro de Ciência do Sistema Terrestre (COCST)  
Dr. Evandro Marconi Rocco - Coordenação-Geral de Engenharia e Tecnologia  
Espacial (CGETE)  
Dr. Hermann Johann Heinrich Kux - Coordenação-Geral de Observação da Terra  
(CGOBT)  
Dra. Ieda Del Arco Sanches - Conselho de Pós-Graduação - (CPG)  
Sílvia Castro Marcelino - Serviço de Informação e Documentação (SESID)

**DIGITAL LIBRARY:**

Dr. Gerald Jean Francis Banon  
Clayton Martins Pereira - Serviço de Informação e Documentação (SESID)

**DOCUMENT REVIEW:**

Simone Angélica Del Ducca Barbedo - Serviço de Informação e Documentação  
(SESID)  
André Luis Dias Fernandes - Serviço de Informação e Documentação (SESID)

**ELECTRONIC EDITING:**

Ivone Martins - Serviço de Informação e Documentação (SESID)  
Cauê Silva Fróes - Serviço de Informação e Documentação (SESID)



MINISTÉRIO DA  
CIÊNCIA, TECNOLOGIA,  
INOVAÇÕES E COMUNICAÇÕES



sid.inpe.br/mtc-m21c/2020/01.24.13.23-TDI

## A CRITICAL SOFTWARE PROCESS SELECTION FOR VERY SMALL ENTITIES (VSE)

Gledson Hernandes Diniz

Master's Dissertation of the Graduate Course in Engineering and Space Technology/Space Systems Engineering and Management, guided by Drs. Ana Maria Ambrosio, and Carlos Henrique Netto Lahoz, approved in November 14, 2019.

URL of the original document:

<<http://urlib.net/8JMKD3MGP3W34R/3UQR2LE>>

INPE  
São José dos Campos  
2019

## Cataloging in Publication Data

---

Diniz, Gledson Hernandes.

D615c A critical software process selection for very small entities (VSE) / Gledson Hernandes Diniz. – São José dos Campos : INPE, 2019.

xxiv + 94 p. ; (sid.inpe.br/mtc-m21c/2020/01.24.13.23-TDI)

Dissertation (Master in Engineering and Space Technology/Space Systems Engineering and Management) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.

Guiding : Drs. Ana Maria Ambrosio, and Carlos Henrique Netto Lahoz.

1. Critical software. 2. Process selection. 3. Tailoring. 4. Profile.  
5. Very Small Entities (VSE). I.Title.

CDU 334.012.63:004.42

---



Esta obra foi licenciada sob uma Licença [Creative Commons Atribuição-NãoComercial 3.0 Não Adaptada](https://creativecommons.org/licenses/by-nc/3.0/).

This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](https://creativecommons.org/licenses/by-nc/3.0/).

Aluno (a): *Gledson Hernandes Diniz*

Título: "A CRITICAL SOFTWARE PROCESS SELECTION FOR VERY SMALL ENTITIES (VSE)"

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de **Mestre** em

**Engenharia e Tecnologia Espaciais/Eng. Gerenc. de Sistemas Espaciais**

Dr. Maurício Gonçalves Vieira Ferreira



Presidente / INPE / SJCampos - SP

Participação por Vídeo - Conferência

Aprovado  Reprovado

Dra. Ana Maria Ambrosio



Orientador(a) / INPE / São José dos Campos - SP

Participação por Vídeo - Conferência

Aprovado  Reprovado

Dr. Carlos Henrique Netto Lahoz



Orientador(a) / UNIVAP / São José dos Campos - SP

Participação por Vídeo - Conferência

Aprovado  Reprovado

Dr. Nilson Sant'Anna

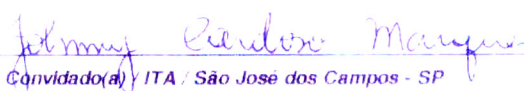


Membro da Banca / INPE / SJCampos - SP

Participação por Vídeo - Conferência

Aprovado  Reprovado

Dr. Johnny Cardoso Marques



Convidado(a) / ITA / São José dos Campos - SP

Participação por Vídeo - Conferência

Aprovado  Reprovado

Este trabalho foi aprovado por:

maioria simples

unanimidade



*“Life is really simple, but we insist on making it complicated.”*

*Confucius*





I dedicate this work to my family and friends.



## **ACKNOWLEDGEMENTS**

I would like to thank my research advisors, Prof. Ana Maria Ambrosio and Prof. Carlos Henrique Netto Lahoz, for believing in me, guiding, inspiring and supporting during my dissertation work. I also want to thank them for their great enthusiasm and unwavering patience towards me; how they explained me the way to develop this work and the dynamics of the academic world. Their intuition, work ideas, detailed knowledge, and unending quest to arrive at the correct contents consistent with the existing processes, provided me with an invaluable experience and benefits. I would also like to gratefully acknowledge and thank all my colleagues and coworkers from INPE to whom I am very thankful for their valuable time during the progress of the work and their valuable suggestions. I gratefully acknowledge all my teachers, my family members, specially my wife and daughters, and my friends who have greatly contributed towards my evolution and supported me continuously.



## ABSTRACT

Aligned with the worldwide trend of developing using small teams, most of the critical software has been developed by Very Small Entities (VSE), organizations with up to 25 people. Although there are many process models and standards, the majority of them do not specifically aim the needs of organizations, such as VSE, for whom ISO/IEC 29110 standard was created. The available processes models from ISO/IEC 29110, called Generic Profile Group, are applicable to VSEs that do not develop critical systems or software products. For their use, process models are customized to obtain the project's defined software process, considering individual characteristics. These models, such as the framework from European Space Standardization Coordination (ECSS), generally include provisions for customization based only on the software criticality level, and each organization should eventually select other criteria to indicate the risk that the project is prepared to assume by determining the application of the processes. The set of all possible software is very large, so a set of processes suitable for use by any potential organizations and projects would be excessively general or complex, and difficult to apply. Using standard terminology (documents, processes, activities, tasks, functions, and artifacts) that each organization understands is not a trivial task. Since process selection must be conducted in a thoughtful and disciplined manner, research has been conducted on the effects of project characteristics and their use for project evaluation. Selecting processes requires criteria to assess their relevance to project needs directing to process subsets according to the classification resultant from project evaluation. In this context, the objective of this dissertation is to propose a process selection approach applicable to critical software projects in VSE. Project evaluation is achieved by identifying specific criteria that influence projects and using them in a framework to assess their implications. The projects are classified based on the criticality rating of the software, along with the result of the project evaluation, indicating the use of different process profiles, selected from a common core of international standard requirements. The results show that the selection of project characteristics is a means to support the understanding of influence factors for process selection, and that ECSS processes can be applied to VSE, comprising appropriate process sets according to the evaluation of each project.

**Keywords:** Critical software. Process selection. Tailoring. Profile. Very Small Entities (VSE).



## UMA SELEÇÃO DE PROCESSOS DE SOFTWARE CRÍTICO PARA ENTIDADES MUITO PEQUENAS (VSE)

### RESUMO

Alinhado à tendência mundial de desenvolvimento usando equipes pequenas, a maioria dos softwares críticos têm sido desenvolvida por Entidades Muito Pequenas (VSE), organizações com até 25 pessoas. Embora existam muitos modelos e padrões de processo, a maioria deles não visa especificamente às necessidades de organizações como as VSE, para quem o padrão ISO/IEC 29110 foi criado. Os modelos de processos disponíveis nesse padrão, denominados *Generic Profile Group*, são aplicáveis às VSE que não desenvolvem sistemas ou produtos de software críticos. Para sua utilização, os modelos de processo são customizados para obter processo de software definido do projeto, considerando características individuais. Esses modelos, como o framework da Coordenação Europeia de Padronização do Espaço (ECSS), geralmente incluem provisões para customização com base apenas no nível de criticidade do software e cada organização deve eventualmente selecionar outros critérios para indicar o risco que o projeto está preparado para assumir determinando a aplicação dos processos. O conjunto de todos os softwares possíveis é muito grande, assim um conjunto de processos adequado para uso por quaisquer organizações e projetos em potencial seria excessivamente geral ou complexo, além de difícil de aplicar. Interpretar a terminologia padrão (documentos, processos, atividades, tarefas, funções e artefatos) de forma que cada organização entenda não é uma tarefa trivial. Como a seleção de processos deve ser realizada de maneira ponderada e disciplinada, pesquisas têm sido feitas sobre os efeitos das características de projeto e sua utilização para classificação de projetos. Selecionar os processos requer critérios para avaliar a sua relevância quanto às necessidades do projeto, gerando subconjuntos de processos selecionados de acordo com a classificação dos projetos. Nesse contexto, o objetivo desta dissertação é propor uma abordagem para a seleção de processos aplicável a projetos críticos de software em VSE. A avaliação dos projetos é obtida por meio da identificação de critérios específicos que os influenciam e sua utilização em uma estrutura para avaliar suas implicações. Os projetos são classificados a partir do nível de criticidade do software em conjunto com o resultado da avaliação dos projetos, indicando a utilização de diferentes perfis de processos, selecionados a partir de um “núcleo comum” de requisitos de padrões internacionais. Os resultados mostram que a seleção de características dos projetos é um meio de apoiar o entendimento dos fatores de influência para seleção de processos e, ainda, que os processos da ECSS podem ser aplicados para VSE, compreendendo conjuntos de processos adequados de acordo com a avaliação de cada projeto.

**Palavras-chave:** Software crítico. Seleção de processos. Adaptação. Perfil. Entidades Muito Pequenas (VSE).



## FIGURES LIST

	<u>Page</u>
Figure 1.1: The three process critical dimensions. ....	2
Figure 1.2 : The research ‘onion’.....	11
Figure 1.3: Dissertation organization.....	13
Figure 2.1: IDEF0 A-0 diagram example. ....	16
Figure 2.2: Life cycle process groups.....	19
Figure 2.3: Assessment dimensions.....	20
Figure 2.4: Software related processes in ECSS Standards. ....	25
Figure 2.5: Structure of the Continuous and Staged Representations. ....	29
Figure 2.6: MPS.BR elements.....	30
Figure 2.7: MPS.BR model.....	31
Figure 2.8: S4S contents. ....	32
Figure 2.9: Generic profile group’s contents.....	34
Figure 2.10: ISO/IEC 29110 PM and SI relationship. ....	36
Figure 3.1: Literature review process. ....	37
Figure 4.1: GAPS development.....	45
Figure 4.2: Tailoring criteria definition. ....	46
Figure 4.3: Project evaluation.....	54
Figure 4.4: Processes selection. ....	58
Figure 4.5: VSE critical profile group processes.....	65



## TABLES LIST

	<u>Page</u>
Table 1.1: MSME share distribution. ....	6
Table 2.1: Software criticality categories definition.....	26
Table 2.2: Function criticality description.....	27
Table 2.3: Comparison of Capability and Maturity Levels .....	29
Table 2.4: S4S processes. ....	33
Table 3.1: PICO framework. ....	38
Table 3.2: Search results – Reference date: 07/Oct/2019.....	39
Table 4.1: Names and descriptions of project factors.....	48
Table 4.2: Factor voting spreadsheet example.....	53
Table 4.3: Evaluation framework example. ....	56
Table 4.4: Software criticality related project classification.....	57
Table 4.5: VSE software project profile classification. ....	57
Table 4.6: Aspects related to the tailoring criteria.....	60
Table 4.7: S4S Processes.....	67
Table 4.8: ISO/IEC 29110 Basic Profile processes. ....	68
Table 5.1: Tailoring criteria selection.....	72
Table 5.2: Evaluation framework. ....	74
Table 5.3: Case projects framework grading results. ....	75
Table 5.4: Project classification for the space context.....	75
Table 5.5: Processes for GAPS Basic Critical Profile.....	77
Table 5.6: Processes for Intermediate GAPS Critical Profile.....	77
Table 5.7: Processes for GAPS Advanced Critical Profile.....	78
Table 5.8: Additional processes. ....	78



## LIST OF ACRONYMS AND ABBREVIATIONS

ACQ	Acquisition
C	C programming language
C++	Extension of C programming language
CBERS	China-Brazil Earth Resources Satellite
CI	Configuration Item
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMU	Carnegie Mellon University
CSP	Critical Space Profiles
CUS	Customer-Supplier
DEV	Development
DO	DOcument
DoD	Department of Defense
E	Engineering
ECSS	European Cooperation for Space Standardization
ENG	Engineering
EO	Earth Observation
ESA	European Space Agency
FAA	Federal Aviation Administration
GAPS	Generic Approach for Process Selection
GGPSw	<i>Grupo de Garantia do Produto de Software</i> (Software Product Assurance Group)
HB	Handbook
IDEF0	Integration Definition language 0
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFC	International Finance Corporation
INCOSE	International Council on Systems Engineering
INPE	<i>Instituto Nacional de Pesquisas Espaciais</i> (National Institute for Space Research)
IPD	Integrated Product Development
ISO	International Organization for Standardization

IT	Information technology
LOC	Lines Of Code
LOFI	Level Of FAA Involvement
M	Management
MA	<i>Modelo de Avaliação</i> (Assessment Model)
MAN	Management
MCTIC	<i>Ministério da Ciência Tecnologia, Inovações e Comunicações</i> (Ministry of Science, Technology, Innovations and Communications)
MN	<i>Modelo de Negócios</i> (Business Model)
MPS.BR	Brazilian Software Process Improvement
MSME	Micro, small and medium-sized enterprises
MR	<i>Modelo de Referência</i> (Reference Model)
N/A	Not Applicable
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organization
OECD	Organization for Economic Co-operation and Development
ORG	Organization
PA	Product Assurance
PAM	Process Assessment Model
PICO	Population/Problem, Intervention, Control/Comparison and Outcome
PM	Project Management
PRM	Process Reference Model
Q	Quality
QA	Quality Assurance
RAM	Reliability, Availability and Maintainability
RAMS	Reliability, Availability, Maintainability and Safety
RE	Regulated Environment
RTCA	Radio Technical Commission for Aeronautics
S4S	SPICE for Space
SCS	Safety Critical Software
SE	Software Engineering
SECM	System Engineering Capability Model
SEI	Software Engineering Institute
SI	Software Implementation

SME	Small and medium-sized enterprises
SOFTEX	Software Excellence Promotion Association
SP	Standardized Profile
SPA	Software Process Assessment
SPCMM	Software Process Capability/Maturity Models
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination
ST	Standard
SUP	Support
SVC	Services
SW	Software
SWE	Software Engineering
SwPA	Software Product Assurance
TM	Technical Memorandum
TSR	Total Score Result
TQM	Total Quality Management
VSE	Very Small Entity
WBS	Work Breakdown Structure
WG	Working Group
WP	Work Package
WTO	World Trade Organization





## SUMMARY

	<u>Page</u>
1 INTRODUCTION .....	1
2 CONCEPTUAL FOUNDATIONS .....	15
2.1 Process representation .....	15
2.2 Software quality .....	16
2.3 Software process standards and models .....	18
2.3.1 ISO/IEC 12207 .....	18
2.3.2 ISO/IEC 15504 (SPICE) .....	20
2.3.2.1 Process dimension .....	21
2.3.2.2 Capability dimension .....	22
2.3.3 ECSS-E-ST-40 and ECSS-Q-ST-80 .....	23
2.3.4 Capability Maturity Model Integration (CMMI) .....	27
2.3.5 MPS.BR .....	29
2.3.6 SPICE for Space (S4S) .....	31
2.3.7 ISO/IEC 29110 .....	33
3 LITERATURE REVIEW .....	37
3.1 Critical software processes tailoring .....	39
3.2 Software processes in small entities .....	41
3.3 Literature review analysis .....	43
4 GENERIC APPROACH FOR PROCESS SELECTION (GAPS) .....	45
4.1 Step 1 - Tailoring criteria definition .....	45
4.1.1 Project factors .....	47
4.1.2 Voting framework for criteria selection .....	53
4.2 Step 2 - Project evaluation .....	54
4.2.1 Evaluation structure .....	55
4.2.1.1 GAPS evaluation framework .....	56
4.3 Processes selection .....	58
4.3.1 Project aspects .....	59
4.3.2 Process profiles .....	65
4.3.2.1 Capability dimension .....	66

4.3.2.2	Process dimension.....	66
5	CRITICAL SPACE PROFILES (CSP).....	69
5.1	Projects.....	69
5.1.1	Project 1: On-board data handling application .....	69
5.1.2	Project 2: Ground control .....	70
5.1.3	Project 3: Application for remote sensing payload .....	71
5.2	Applying GAPS.....	71
5.2.1	Step 1 - Tailoring criteria definition – space context.....	71
5.2.2	Step 2 - Projects evaluation – space context .....	74
5.2.3	Step 3 - Process selection – space context.....	76
6	CONCLUSION.....	79
6.1	Limitations.....	80
6.2	Future work suggestion.....	81
6.3	Published works.....	81
	REFERENCES.....	83

## 1 INTRODUCTION

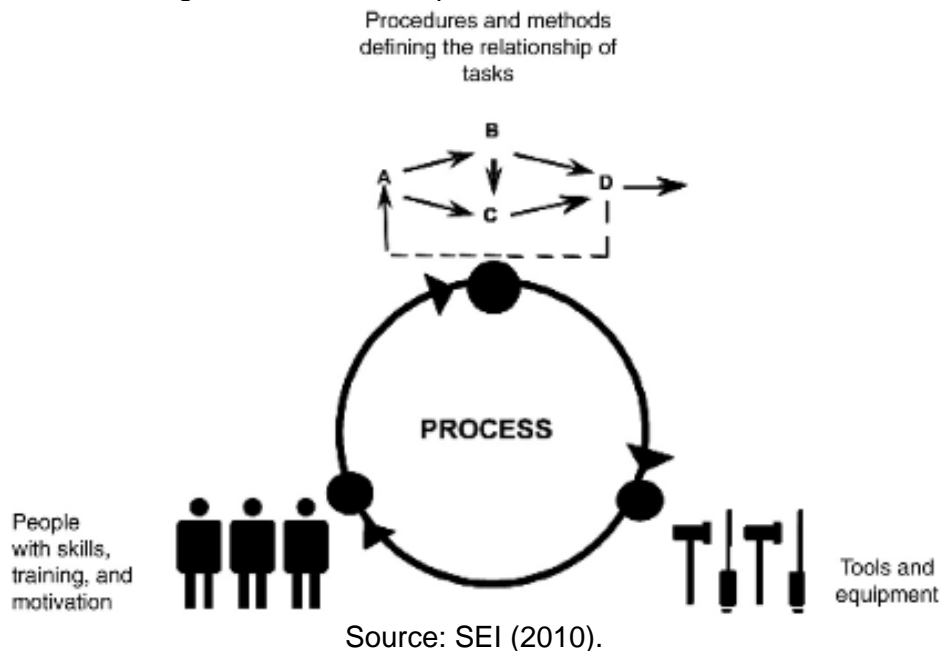
The number of system solutions provided by software and the complexity of the needs addressed to it are growing increasingly, turning software into an enabler of progress, making it a “complexity sponge” (NASA, 2009).

In the late 1960s, during a conference sponsored by NATO's Science Committee, the term Software Engineering (SE) emerged to establish the use of sound engineering principles such as product cost-effectiveness to get software running reliably and efficiently (NAUR; RANDELL, 1969 *apud* HIRAMA, 2011). SE makes use of prescriptive models, which consist of distinct sets of activities, actions, tasks, milestones, and consequent work products (programs, documentation, and data) (IEEE, 1990).

SE can be understood as a layered technology, establishing techniques and practices for software development for a wide range of applications and different types of devices (PRESSMAN, 2007). In this context, a software development process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and its related products (SEI, 2010). According to Miyashiro and Ferreira (2014), ongoing research and studies in order to achieve software quality are focused on its development processes.

The United States Department of Defense (DoD) sponsored the development of an assessment model called Capability Maturity Model for software (SW-CMM), developed by Carnegie Mellon University's Software Engineering Institute (CMU-SEI). SEI is responsible for evolving the CMM family, now Capability Maturity Model Integration (CMMI), and for conducting several other software engineering researches. In its *CMMI® for Development Version 1.3* (SEI, 2010), SEI presents the three critical dimensions that are held together by the organization's software process, shown in Figure 1.1: people; procedures and methods; tools and equipment.

Figure 1.1: The three process critical dimensions.



The three critical dimensions from Figure 1.1 are:

- People with skills, training and motivation: to achieve the desired result every process needs to be accepted, understood, and encouraged by the people involved;
- Tools and equipment: the use of tools that assist in the execution of the process needs to be inserted in a controlled and evolutionary way, making people control and use them; and
- Procedures and methods defining the relationship of tasks: to be instituted in a company, the process needs to be designed to naturally lead to good practices. It should be minimally bureaucratized and must allow some flexibility.

As the quality of development process is closely related to the quality of a software product, Process Reference Models (PRMs) have been used in order to assist companies organizing and disciplining at the development activities in order to increase the quality of their products and productivity (CASS;

VÖLCKER et al., 2001; EITO-BRUN, 2013; FELDT; TORKAR et al., 2010; WANGENHEIM; HAUCK et al., 2010).

Process 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; O'CONNOR; CLARKE, 2016). Standards published by committees, international technical entities or regulatory agencies influence software development through guidelines for processes and products considering their associated risks (MUNCH; ARMBRUNT et al., 2012).

Software processes have the potential to be highly complex (CLARKE; O'CONNOR; LEAVY, 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).

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 difficult to apply. Standard processes typically cannot be used without customization (GINSBERG; QUINN, 1995), therefore, PRMs are customized to obtain the project's defined software process (SEI, 2010).

Although the need to tailor software processes to specific project requirements is widely accepted, the way of doing it is frequently unclear (KALUS; KUHRMANN, 2013). Organizations such as the European Space Standardization Coordination (ECSS) (ECSS, 2017a) and the National Aeronautics and Space Administration (NASA) (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 project is likely to take and the extent to which the processes are made applicable. Research has been conducted on

the effects of project factors for the resulting software process and how to use this knowledge to choose those to be considered for processes tailoring (KALUS; KUHRMANN, 2013).

## **1.1 Context**

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

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) points out that critical software developments in regulated environments (RE) such as space, aeronautics, medical, railway, and nuclear must consider specific factors such as type of software product, role of software in the system, size of the system and level of risk.

Standards published by committees, international technical entities, or regulatory agencies influence software development through risk-based software process and product guidelines (MUNCH; ARMBRUNT et al., 2012). Typically, each domain of knowledge has its own software standard, such as RTCA/DO-178C (RTCA, 2011) for aeronautics and ECSS-E-ST-40C (ECSS, 2009a) and ECSS-Q-ST-80C (ECSS, 2017a) for space systems.

Standardization for software development processes comprises the concept of the Standardized Profile (SP), which is defined by the International Organization for Standardization (ISO) as a “set of one or more base standards and/or SPs, and, where applicable, the identification of chosen classes, conforming subsets, options and parameters of those base standards, or SPs necessary to accomplish a particular function”. A possible analogy is that a profile is like a

bill of materials composed of parts of standards such as ISO/IEC 12207 (ISO/IEC/IEEE, 2017) or ISO/IEC 15288 (ISO/IEC/IEEE, 2015).

The number of organizations that need to demonstrate compliance with regulatory standards is increasing and many of these standards-based regulations require the presence of explicit software processes (MUNCH; ARMBRUNT et al., 2012), for which Software Process Improvement (SPI) models have been employed (GORSCHEK; WOHLIN, 2006).

When used as a comparative basis for software process evaluation and/or improvement, these best practice frameworks have been called Software Process Capability/Maturity Models (SPCMM) (SALVIANO; FIGUEIREDO, 2008). A large variety of software process capability/maturity models have been used over the years, with a trend to the specialization of those models for specific domains and most of those models concentrated around the CMM/CMMI framework and the standard ISO/IEC 15504 (ISO/IEC, 2008) (WANGENHEIM; HAUCK et al., 2010).

Despite being comprehensive and rigorous evaluation models, prescriptive SPI models, such as CMMI and ISO/IEC 15504, are considered heavy. (KUILBOER; ASHRAFI, 2000) Literature reports that these heavy SPI models and their evaluation methods are considered expensive by small and medium-sized enterprises (SME) (CATER-STEEL, 2001; JOHNSON; BRODMAN, 1997; KELLY; CULLETON, 1999; LARYD; ORCI, 2000; VILLALÓN et al., 2002; SCHOEFFEL; BENITTI, 2015), which is related to the fact that process improvement models are not being extensively deployed and their influence in the software industry remains more at a theoretical level (LAPORTE; O'CONNOR; PAUCAR, 2015). The acronym SME is used as a generic term referring to organizations that are not large. Sometimes, the acronym gains the term "micro", becoming MSME, to emphasize the inclusion of smallest companies (BRUHN; HOMMES et al., 2017).

Although there is no mutually agreed definition about the names related to sizes, the Organization for Economic Co-operation and Development (OECD)

considers companies with up to 10 employees as micro; 10 to 50 employees as small, and 50 to 250 as medium. (WTO, 2016). Criscuolo et al. (2014) shows that MSMEs account for over 95% of all companies in 17 OECD countries, in addition to Brazil, representing 63% of total employment. As presented in Table 1.1, the majority of MSMEs are micro, representing 82.9%; 8 of more than 12 million enterprises covered (KUSHNIR; MIRMULSTEIN; RAMALHO, 2010).

Table 1.1: MSME share distribution.

Countries	% of micro ( < 10 employees)	% of small (10 to 50 employees)	% of medium (50 to 250 employees)
Developed	87.1	10.7	23.2
Developing	80.5	15.6	3.9
G20 developing	82.1	13.2	4.7
Other developing	80.5	14.9	4.5
Least developed	78.6	20.7	0.6
Total	82.9	13.8	3.3

Source: Adapted from WTO (2016).

Financial, structural, organizational, and managerial difficulties in MSMEs have led to investigations and developments in assessment methods that meet the needs of these organizations, called lightweight methods, typically tailored and in accordance to comprehensive and heavyweight methods. Developers of lightweight evaluation methods normally claim that their methods are successful based on some case studies and feedback from evaluated organizations (ALEXANDRE; RENAULT; HABRA, 2006; ANACLETO et al., 2004; CIGNONI, 1999; KUVAJA; PALO; BICEGO, 1999; ROUT; TUFFLEY et al., 2000). Literature has shown lightweight methods focused on the process, critical success factors and barriers on known frameworks, as well as lessons learned from case studies (KOMI-SIRVIÖ, 2004; NIAZI; WILSON; ZOWGHI, 2003, 2005, 2006).



Aligned with the world's MSME context, most of the critical software has been developed by Very Small Entities (VSE), organizations with up to 25 people. Although there are many standard processes, most of them do not specifically aim the needs of organizations such as VSE, for whom ISO/IEC 29110 standard (ISO/IEC, 2011a) was created. The SPs from ISO/IEC 29110, gathered in the Generic Profile Group, are applicable to VSEs that do not develop critical systems or software products.

## **1.2 Motivation**

This dissertation was primarily motivated by the scarceness of standards with explicit software processes selection approach that can be used by VSE developing critical software.

VSEs 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; BASRI; COLEMAN, 2010).

The Rapid Software Assessment approach (LAHOZ; RICHTER; RICO, 2015), presented at *ESA Software Product Assurance Workshop*, event promoted by the European Space Agency (ESA) (ESA, 2015), comprises elements from ISO/IEC 29110 and ECSS standards (LAHOZ, 2015), focusing on practices and requirements for software development in small entities.

Process tailoring needs to be performed in a thoughtful and disciplined manner. Using standard terminology (documents, processes, activities, tasks, functions, and artifacts) that each organization understands is not a trivial task. Research has been conducted on the effects of project characteristics and their use for project classification. Selecting processes requires criteria to assess their relevance to project needs, directing to process subsets according to the classification obtained from project evaluation. The subset of applicable

processes selected through project classification can vary, depending mainly on factors such as type, size, complexity, and phase of addressed project (KALUS; KUHRMANN, 2013).

Although it is commonly assumed that organizational performance is increased by using PRMs for process assessments and improvement (GOLDENSON; GIBSON, 2003), this has not been a common practice, as PRMs have been adopted by just a small number of organizations, mainly large and medium-sized ones (KALINOWSKI et al., 2015).

ISO/IEC 29110 set of standards clearly states that its contents are not intended for critical software developers. Nevertheless, one of ISO/IEC 29110 authors suggested Basic Profile as foundation for critical applications (LAPORTE, 2017), identifying the gaps to critical project requirements; and also pointed out that their management and engineering guides do not forbid addition of critical domain specific tasks or roles, such as Quality Assurance (QA).

In Brazil, the National Institute for Space Research (INPE) is one of the organizations responsible for space research efforts, including satellites development (ALBUQUERQUE, 2011). The author of this dissertation is part of INPE's Software Product Assurance Group (GGPSw), whose activities include the definition of requirements and processes for space software. Therefore, another motivation for research on this subject is the work of the ESA VSEs Focus Group (ESA, 2018), organized by specialists from space software domain focused on defining how small organizations could participate in space projects, developing a step-wise approach for lightweight software process assessment and improvement mostly based on ISO/IEC 29110 and ECSS standards. The author and advisors are closely involved in this group as it will contribute to INPE's Software Product Assurance Group.

### **1.3 Problem definition**

Most of the software has been developed by small groups (LAPORTE; O'CONNOR; PAUCAR, 2015); however most of the software development standards do not specifically aim the needs of small enterprises (O'CONNOR; BASRI; COLEMAN, 2010).

For many VSEs, 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; LOHIER, 2017). Currently, there is no standard process selection approach for critical software development that considers typical VSE's characteristics.

In the context of the software industry, the two main process models, CMMI and ISO/IEC 12207 (CRISÓSTOMO et al., 2016), besides regulated environments (RE) software standards, have a common core of principles, enabling the establishment of a common line of processes that be used in several RE (HAWKINS; HABLI; KELLY, 2013). These models containing a common set of processes can be used integrally or with some adaptation for two or more different domains, so it can be called a multi-domain solution. (MARQUES, 2016).

When customizing these models for a specific domain, there is a need for knowledge acquisition from domain experts. Despite the existence of several techniques and methods of knowledge acquisition, mostly based on interviews and analysis of texts in natural language (MOTODA; BOOSE; GAINES, 1991), they do not take into account or aim to support the customization of process models.

Therefore, there is the need of methods that provide systematic support for customization of software process models. So, in this context, the problem to be

addressed in this research is: *How to systematically perform software process model customization for critical software in VSE?*

#### **1.4 Objective**

The objective of this work is to propose an approach for process selection applicable to critical software in VSE. To achieve that objective, this dissertation considers identification of specific criteria that influence projects and their implications on processes considering the typical resources limitations of VSE.

A secondary objective of this work is to present an approach ready to use in VSE interested in developing space software, considering this context's specific features.

#### **1.5 Scope**

This dissertation focuses exclusively in the definition of processes, customizing software process models for specific domains or projects. The implementation of such processes is not considered as it may vary according to each project.

Although the processes presented in the ISO/IEC 29110 series are not the only possible to use in VSE software development, these were considered as basis in the scope of this work. Similarly, there are several standards that regulate the development of critical software, but in the scope of this work, the processes from ECSS standards are considered applicable.

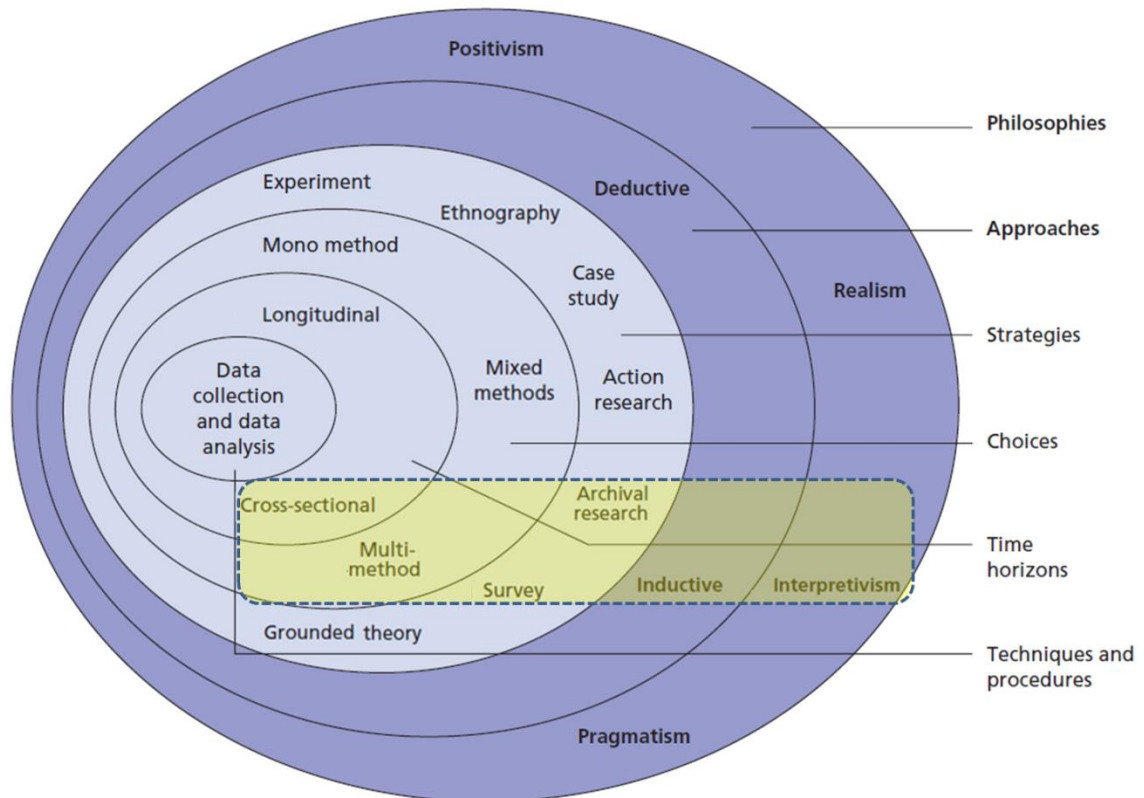
#### **1.6 Method**

The research methodology employed in academic work needs to be appropriate to the intended type of study, but the nature of the problem is what actually determines the choice of method (RICHARDSON, 1997).

Myers (1997) suggests that the choice of a specific qualitative research method is independent of the philosophical position adopted. Accordingly, Burton (2008), analyzing the research philosophy possibilities, argues that interpretive research is the most appropriate for the development and validation of software process models for a specific domain.

Saunders et al. (2009) proposed a structure in which the scientific method is represented in layers (research 'onion'), presented in Figure 1.2, in which the choices for the present work are highlighted.

Figure 1.2 : The research 'onion'.



Source: Adapted from Saunders et al. (2009).

In the approach of Saunders et al. (2009), presented in Figure 1.2, the scientific methodology first layer is the research philosophy choice, which guides all scientific work. In this perspective, this research is predominantly interpretive,

since it assumes that its object (process customization) is socially and contextually constructed and interpreted. Thus, it is difficult to establish a researcher independence from the research object. The next layer is the scientific approach, where the perspective of this work is aligned with the inductive approach, since it does not start from a pre-established hypothesis, but instead seeks to solve the problem based on the conclusions drawn from the investigated object. The strategy of this research involves the use of: archive research and survey without discarding other methods. The research method is mostly qualitative; however it also involves the quantitative approach in different research phases, so it can be qualified as mixed. Finally, the research time horizon is predominantly cross-sectional since data collection occurs for the survey and validation are unique events over time.

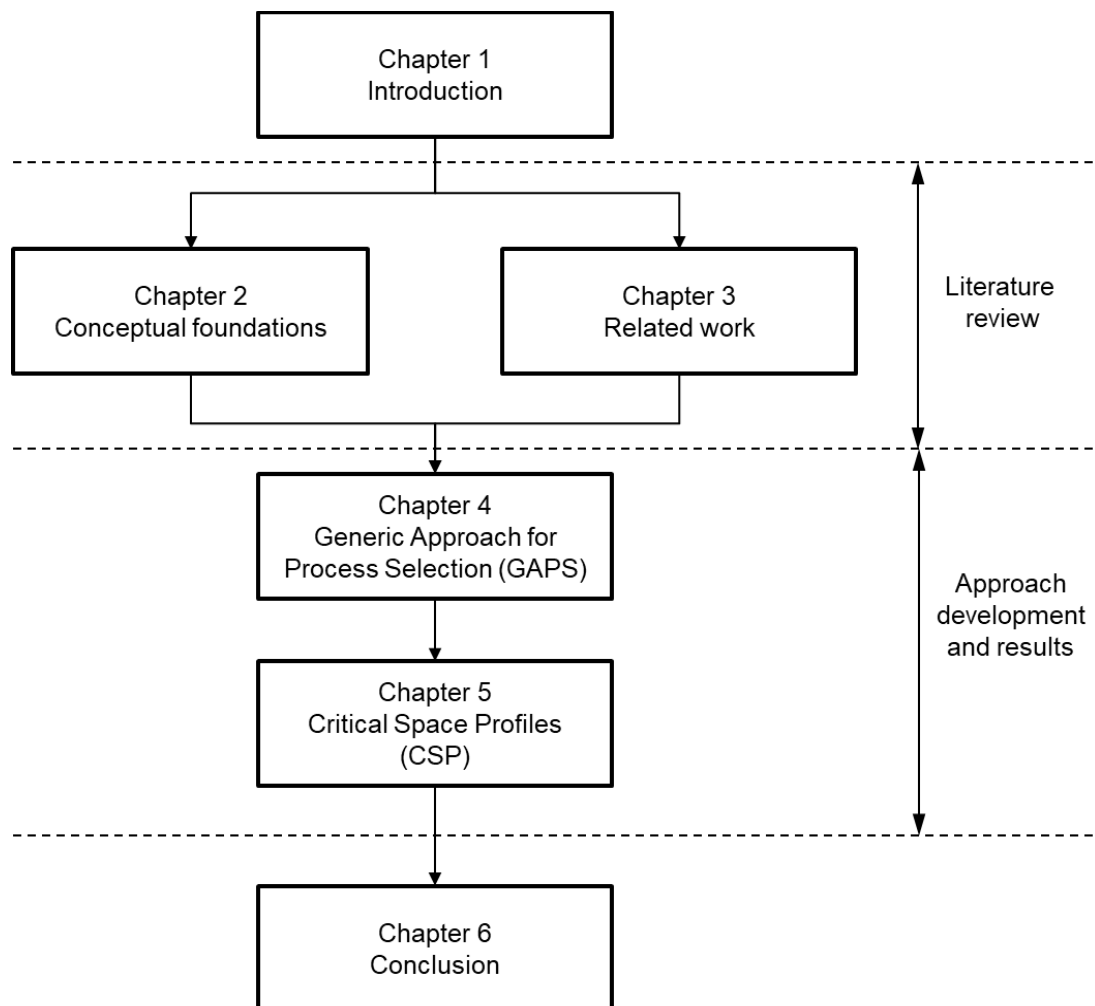
This work is mostly a mixed exploratory basic research, given its objective of developing an approach for process customization. It proposes a method for Software Engineering and not a new methodology, because it does not create a new world view, but takes advantage of the views proposed in process models, establishing a systematic structure of activities and tools to achieve a goal.

Both sets of standard processes, from ISO/IEC 29110 and ECSS, defined as part of this dissertation, consider the different roles and activities present in software development. In order to enable the selection of processes, these sets will be gathered to generate a single set containing the processes structured suitable for their selection in the VSE context, highlighting their sources. After which, information from INPE's space projects will be used to generate a version of the set of processes adequate for VSE developing space software.

## 1.7 Dissertation organization

The dissertation is organized into six chapters, as presented in Figure 1.3. This chapter describes the context, in which this paper is inserted, the factors motivating its accomplishment, its objectives, and method.

Figure 1.3: Dissertation organization.



Source: Author.

In chapter 2 the main concepts on which this work is based are presented, comprising an overview on process representation, software quality, software process standards and models.

Chapter 3 presents the search and identification of related works and a summary of the main works about processes, approaches, project impacting factors and needs of VSE.

Chapter 4 reports about conception and description of the Generic Approach for Process Selection (GAPS), a process selection approach considering relevant factors to software projects, identifying its required methods, activities, inputs and outputs.

Chapter 5 describes the use of GAPS with real projects information to generate the Critical Space Profiles (CSP), a set of profiles developed for VSE developing space software.

Chapter 6 summarizes the main findings and contributions of this research, including recommendations and suggestions for future work and the related publications.



## 2 CONCEPTUAL FOUNDATIONS

This chapter provides the foundation to establish the concepts on which this work is based, comprising an overview on:

- Process representation;
- Software quality;
- Software process standards and models; and
- Tailoring technique.

### 2.1 Process representation

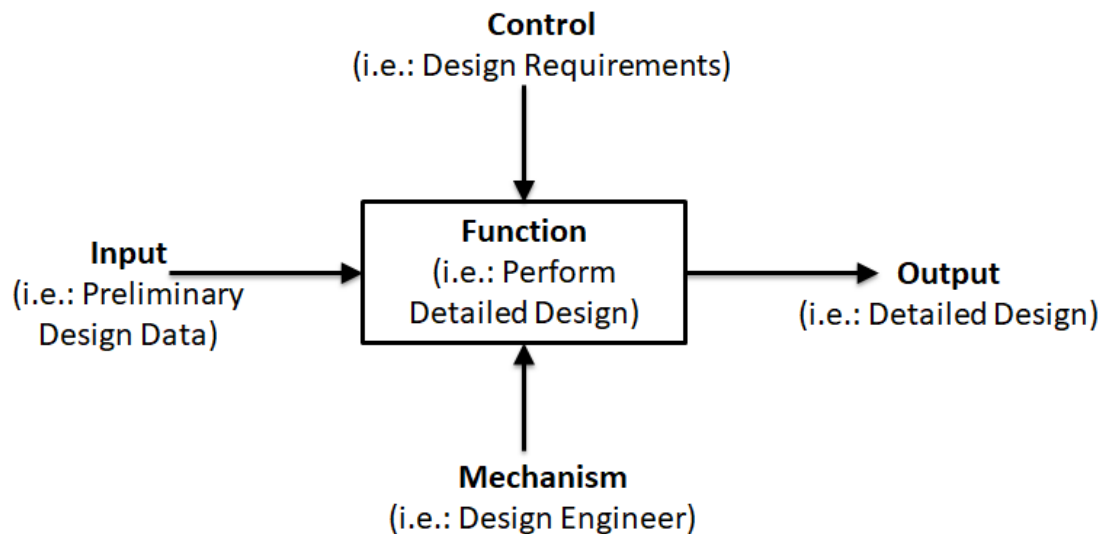
A process is defined as a set of interrelated or interacting activities that use inputs to deliver an output (ISO, 2015), which may be represented by a model, a physical or abstract representation used for calculations, predictions or further assessment (ECSS, 2012).

The Integration Definition language 0 (IDEF0) is a structured representation of the functions, activities or processes within the modeled system or subject area. IDEF0 is widely used in different enterprises and application domains to organize the system into a functional view that helps in identifying the functions to be performed and the data flows between them (FIPS, 1993).

Figure 2.1 presents the A-0 context diagram, which is a special case of IDEF0 that comprises a one-box diagram containing the function and arrows entering or leaving the box representing:

- Function - an activity, process, or transformation identified.
- Input - the data or object that are transformed into Output;
- Control - the conditions required to produce correct Output (data or objects modeled as Control may be transformed, creating Output);
- Mechanism - the means used to perform a Function; and
- Output - the data or objects produced by a Function.

Figure 2.1: IDEF0 A-0 diagram example.



Source: Adapted from FIPS (1993).

The function name shall be an active verb or verb phrase (i.e.: process parts, plan resources, conduct review, monitor performance) and the arrows, which identify data or objects needed or produced by the function, shall be labeled with a noun or noun phrase (i.e.: specifications, test report, requirements, detail design, directive, design engineer, board assembly).

## 2.2 Software quality

Quality is an intangible concept (HETZEL, 1984) with different definitions, such as: satisfaction of consumer needs and suitability for use (JURAN, 1988); ability of a set of characteristics inherent in a product, component or process to meet customer requirements (CHRISSIS; KONRAD; SHRUM, 2011); specification compliance and nonconformance prevention (CROSBY, 1979), and prevention and correction of deviations (HOYLE, 2001).

Software quality can be understood in different ways, depending on the stakeholder. For the developers, quality can be seen as meeting software

methods and standards. For managers, it may be the proximity of estimated project parameters (effort, cost and timeframe). Users can understand it as how easy the software is to use. And for customers, it may be achieved by meeting business needs, deadlines and costs.

ISO (2015) defines quality as the degree to which a set of characteristics of an object meets the requirements; the definition used in this work. Quality must be achieved and is not simply obtained; there is necessarily an effort to be spent on continuous process improvement.

Software Engineering (SE) has enabled the advent of software development and maintenance approaches. From the beginning, processes, methods, and techniques were created to develop increasingly complex software. In this context, Quality Assurance (QA) provides assurance that software products and processes meet the specified requirements. QA has its origin in Total Quality Management (TQM), an organizational management approach (principles, methods, and techniques) for continuous process improvement started in the 1980s. Before SE, software was usually developed and tested continuously until its functionality had been achieved and accepted by the customer. After SE, software needs to be standardized, documented and cost effective (HIRAMA, 2011).

The process improvement principles of statistical quality control from Shewhart (1931), refined by Deming (1986), Crosby (1979), and Juran (1988), were extended and applied to software at International Business Machines (IBM) and SEI by Humphrey (1989), originating the concept of process maturity proposed in the TQM-based CMM (currently CMMI).

Nowadays, it is common for companies to seek an effective quality management system. In this scenario, there are software models and standards that can be used to improve organizations' processes. The most known are the ISO/IEC 12207 standard and the SPICE, CMMI, and MPS.BR models.

## **2.3 Software process standards and models**

In regulated environments, which have impacts on society, standards are adopted to determine the rules to be followed, since the society expects to receive safe and reliable products and services. As a direct consequence, there are requirements to demonstrate that a software product is safe and reliable (MARQUES, 2016).

These standards have objectives or activities that must be met so software product can be approved for operation in its environment of use. Regulatory agencies usually require adherence to established norms and standards, such as ISO/IEC 12207, a standard that establishes a common framework for software lifecycle processes, guiding an understanding of all components of software procurement and delivery (ISO/IEC/IEEE, 2017); and ISO/IEC 15504, which is used as a framework for process evaluation models and methods (ISO/IEC, 2008).

Sections 2.3.1 to 2.3.7 describe the main process standards and models for regulated environments within the scope of this dissertation.

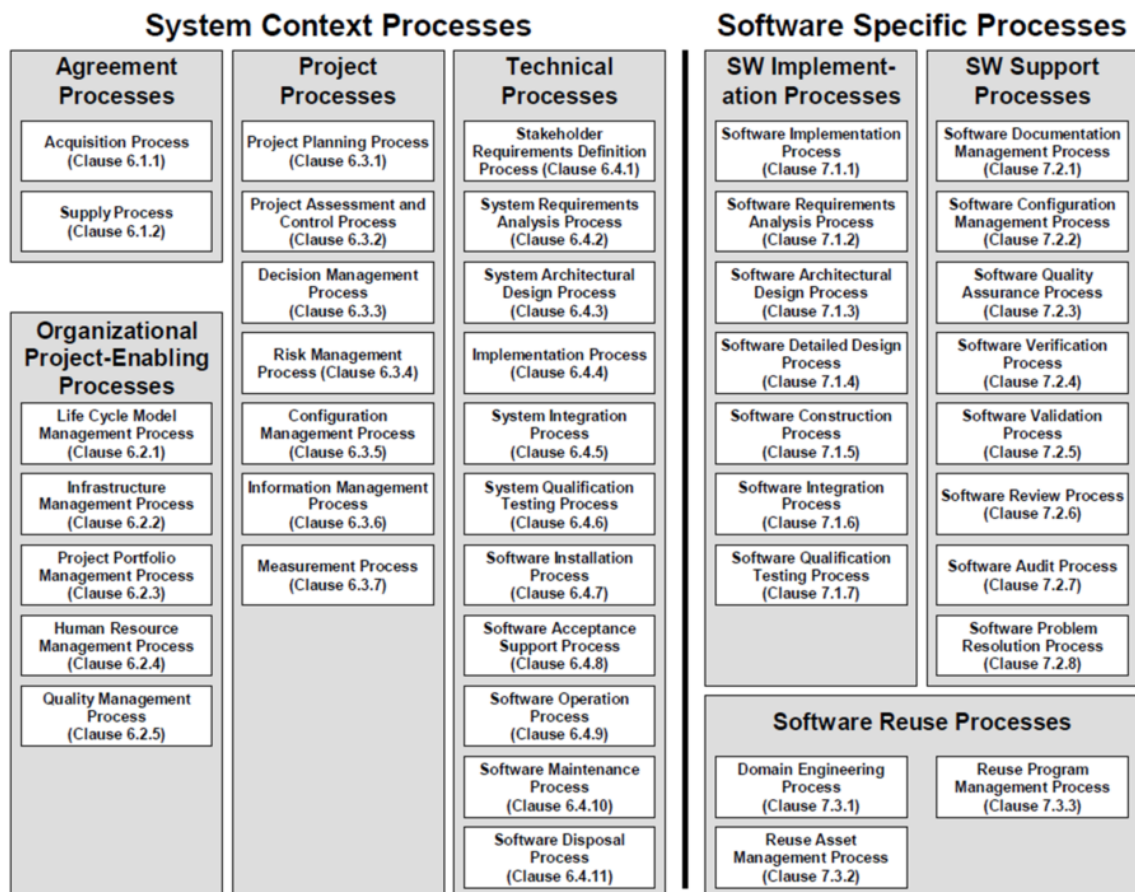
### **2.3.1 ISO/IEC 12207**

This standard's main objective is to establish a framework for software development and lifecycle processes based on which organizations can define their processes with a common language among the large number of methods, techniques, models, and standards that deal with quality. By applying this standard, the entire software development lifecycle, from requirements to maintenance, can be achieved in terms of product quality, budget, deadline, and resources defined in the project. ISO/IEC 12207:2008 applies to the acquisition of products and services, the supply, development, operation, maintenance, and disposal of software products, and the software portion of a system, whether performed internally or externally to an organization (ISO/IEC,

2008a). ISO/IEC/IEEE 12207:2017 is the standard's newest version, published in November 2017 (ISO/IEC/IEEE, 2017).

ISO/IEC 12207 describes each process in terms of its purpose and expected outcomes, and lists activities and tasks that need to be performed to achieve those outcomes. ISO/IEC 12207:2008, which is the version adopted in this work, comprises 43 processes, gathered into seven process groups, presented in Figure 2.2.

Figure 2.2: Life cycle process groups.



Source: ISO/IEC (2008a).

The purposes and outcomes of the processes constitute a Process Reference Model (PRM), which does not represent a particular process implementation

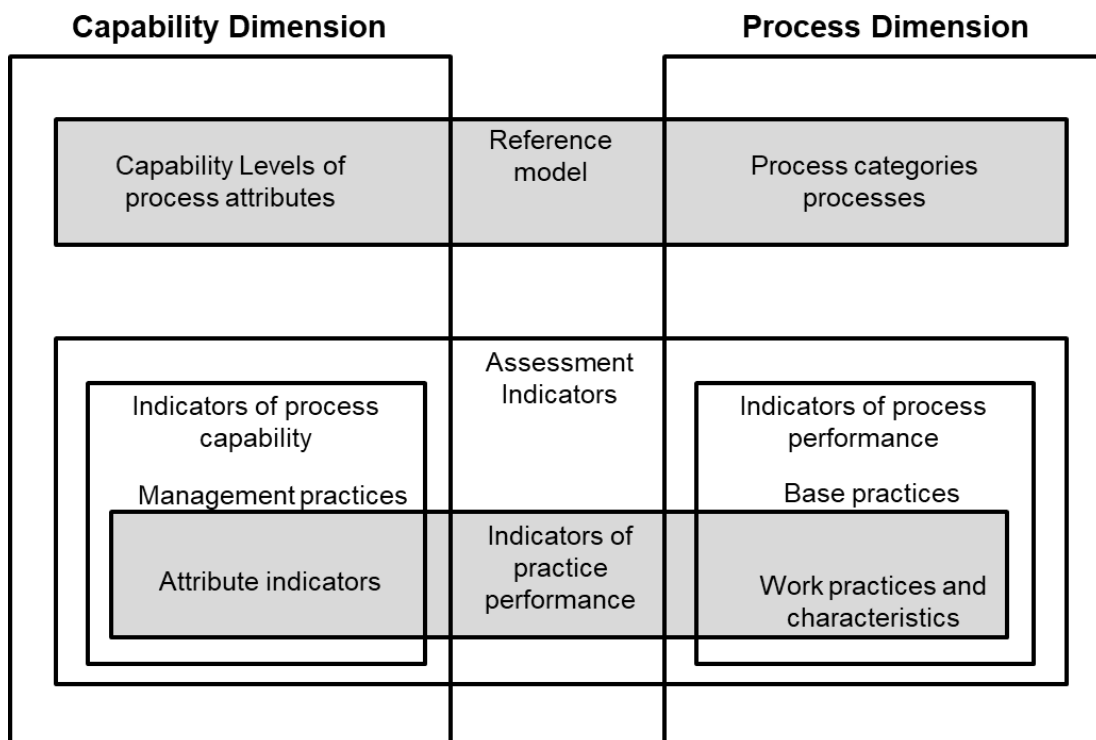
approach nor prescribes a system/software life cycle model, methodology or technique. Conversely, PRM is intended to be adopted based on business needs and application domain.

The organizations' defined processes are adopted by their projects in the context of the customer requirements and the outcomes are used to demonstrate accomplishment of the purpose of a process, helping to determine the capability of the implemented process and to provide source material to plan process improvement.

### 2.3.2 ISO/IEC 15504 (SPICE)

ISO/IEC 15504 *Information technology – Process assessment*, also known as *Software Process Improvement and Capability dEtermination* (SPICE), is a set of technical standards that establishes a framework for process evaluation and improvement based on two dimensions, as presented in Figure 2.3.

Figure 2.3: Assessment dimensions.



Source: Adapted from ISO/IEC (2008).

The two dimensions from the process assessment model architecture are:

- Process dimension: defined by the statements of process purpose and outcomes.
- Capability dimension: a series of process attributes representing the measurable characteristics of a process.

### **2.3.2.1 Process dimension**

The process dimension consists of a set of processes described in ISO/IEC 12207:2008, which presents a universal set of software processes divided into five categories considered essential to software engineering:

- Customer-Supplier (CUS): processes that directly impact the customer, support development and transition of the software to the customer, and provide correct operation and use of the software product and/or service.
- Engineering (ENG): processes that directly specify, implement, or maintain the software product, its relation to the system and its customer documentation.
- Support (SUP): processes which may be employed by any of other processes (including other supporting processes) at various points in software life cycle.
- Management (MAN): processes which contain practices of a generic nature which may be used by anyone who manages any type of project or process within a software life cycle.
- Organization (ORG): processes that establish organization's business goals and develop process, product, and resource assets which, when used in the projects, will help the organization achieve its goals.

Each process category comprises processes that are described in terms of a purpose statement of their unique functional objectives. The purpose of a particular process is addressed by activities named base practices, which identify "what" should be done without specifying "how". Base practices represent the unique, functional activities of the process, producing work products with defined sets of characteristics that may be used to assess a process.

### **2.3.2.2 Capability dimension**

Process capability is expressed in terms of process attributes grouped into capability levels identical to those defined in the reference model. Process attributes are features that can be evaluated on a scale of achievement, providing a measure that indicates a capability level, which constitute a rational way of progressing through six levels, incorporating nine process attributes:

- Level 0 – Incomplete: General failure in achieving the purpose of the process. There are little or no easily identifiable work products or outputs.
- Level 1 – Performed: Process' purpose is generally achieved, sometimes not rigorously planned and tracked. Individuals within the organization recognize that an action is performed and when it is required. There are identifiable work products for the process.
- Level 2 – Managed: There are work products planned and tracked according to specified standards and requirements. The primary distinction from the Performed Level is that the process delivers work products that fulfil expressed quality requirements within defined timescales and resource needs.
- Level 3 – Established: The process is performed and managed using a defined process based upon software engineering principles and with the resources necessary to establish the process definition also in place. The



primary distinction from the Managed Level is that the process uses approved, tailored versions of standard documented processes to achieve the defined outcomes.

- Level 4 – Predictable: The quality of work products is quantitatively known because detailed measures of performance are collected and analyzed, leading to a quantitative understanding of process capability and ability to predict and manage performance. The primary distinction from the Established Level is that the defined process is performed consistently within defined control limits to achieve its defined goals.
- Level 5 – Optimizing: Performance of the process is optimized to meet current and future business needs, and the process achieves repeatability in meeting its defined business goals. Quantitative process effectiveness and efficiency goals (targets) for performance are established and continuously monitored against business goals to obtain quantitative feedback that enables improvement based on analysis of the results. The primary distinction from the Predictable Level is that the defined and standard processes dynamically change and adapt to meet current and future business goals.

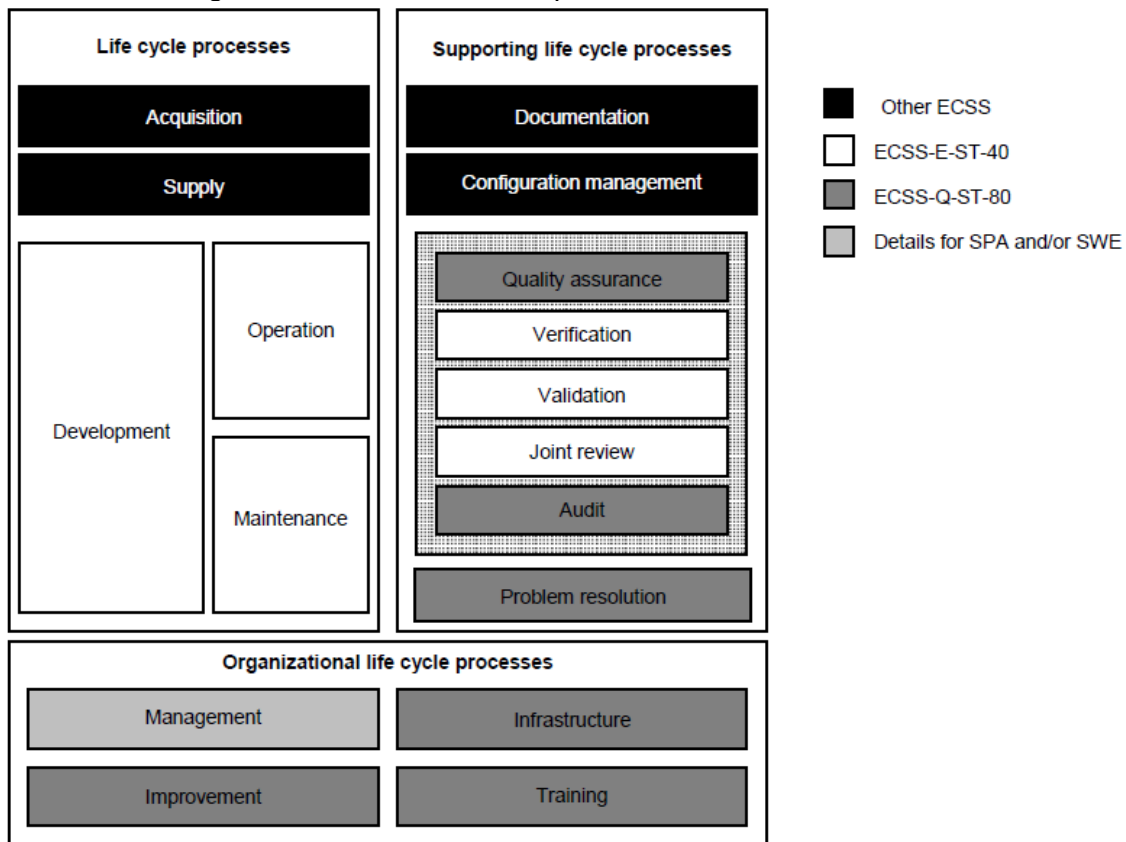
The measure of capability is based upon the nine process attributes of the reference model, which are evaluated on a four point ordinal scale of achievement to measure the aspects used for determining whether a process has reached a given capability level.

### **2.3.3 ECSS-E-ST-40 and ECSS-Q-ST-80**

Software Engineering (SE) and Software Product Assurance (SwPA) disciplines are elements of the Engineering (-E) and Product Assurance (-Q) branches, respectively. ECSS-E-ST-40 addresses the life cycle processes for software products (requirements definition, architectural design, development,

operations, and maintenance) for the different types of software: flight, ground, qualification, testing and verification (ECSS, 2009a). ECSS-Q-ST-80 defines requirements for Product Assurance (PA) of software development and maintenance for space systems to ensure that they run properly and safely in their operational environments. SE and PA standards also include requirements for non-deliverable software, which affects the quality of the space system (e.g. test and verification software). ECSS-Q-ST-80 complements ECSS-E-ST-40 with the PA aspects integrated into the space system SE processes. Together, both standards specify software development processes, schematically presented in Figure 2.4, which is applicable to all elements of a space system, defining the scope of the space software processes and their interfaces with management, which is addressed in Management (-M) branch of the ECSS System (ECSS, 2017a).

Figure 2.4: Software related processes in ECSS Standards.



Source: ECSS (2017a).

ECSS-E-ST-40 and ECSS-Q-ST-80 are organized according to the processes, including activities broken down into tasks in the form of process requirements, producing expected results. Tailoring these standards may have several drivers, such as dependability and safety aspects, software development constraints, product quality objectives, and business objectives. Tailoring for software development constraints needs to consider characteristics of the type (database, real-time) and system (embedded processor, web, host system), as well as the development environment, depending on which specific requirements for verification, review and inspection may be imposed.

ECSS standards present the criticality definition based on the severity of failures consequences (ECSS, 2009b), as described in Table 2.1, in which, for

each software 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 and the existing system compensating provisions. According to this classification, software of category A, B or C is defined as critical; consequently category D denotes non-critical software (ECSS, 2017a).

Table 2.1: Software criticality categories definition.

Criticality category	Definition
<b>A</b>	Software involved in category I functions AND: no compensating provisions exist
	Software included in compensating provisions for category I functions
<b>B</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; and - An operational procedure.
	Software involved in category II functions AND: no compensating provisions exist.
	Software included in compensating provisions for category II functions
<b>C</b>	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; and - An operational procedure.
	Software involved in category III functions AND: no compensating provisions exist.
	Software included in compensating provisions for category III functions.
<b>D</b>	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; and - 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 potential failure of the most critical function implemented (ECSS, 2017b) as shown in Table 2.2.

Table 2.2: Function criticality description.

<b>Severity</b>	<b>Function criticality</b>	<b>Criteria to assign criticality categories to functions</b>
<b>Catastrophic (Level 1)</b>	I	A function that can lead to events resulting in catastrophic consequences if not or incorrectly performed, or if presents anomalous behavior.
<b>Critical (Level 2)</b>	II	A function that can lead to events resulting in critical consequences if not or incorrectly performed, or if presents anomalous behavior.
<b>Major (Level 3)</b>	III	A function that can lead to events resulting in major consequences if not or incorrectly performed, or if presents anomalous behavior.
<b>Minor or Negligible (Level 4)</b>	IV	A function that can lead to events resulting in minor or negligible consequences if not or incorrectly performed, or if presents anomalous behavior.

Source: Adapted from ECSS (2017b).

#### 2.3.4 Capability Maturity Model Integration (CMMI)

A model is a representation of a set of system components or subject area. As such, the CMM was developed by the Software Engineering Institute (SEI) to comprise the essential elements of effective processes using the SE concepts of quality from TQM, embodying the process management premise “the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it” (SEI, 2010).

The Capability Maturity Model Integration (CMMI) was designed to reduce costs and inconsistencies by ensuring adherence to ISO/IEC 15504 through an integration of three CMM: Software CMM (SW-CMM), System Engineering Capability Model (SECM) and Integrated Product Development Capability Maturity Model (IPD-CMM).

CMMI is a set of best practice models for software processes and systems engineering intended for product and service development. It comprises best practices associated with development and maintenance that cover the product life cycle from conception to delivery and maintenance. CMMI is currently in version 1.3 and, since version 1.2, comprises CMMI-DEV (for Development),

CMMI-SVC (for Services) and CMMI-ACQ (for Acquisition). All CMMI models are produced from the framework, which contains 16 core process areas (cluster of related practices) that cover basic concepts that are essential to process improvement in any area of interest (i.e., acquisition, development, services).

CMMI does not specify a particular process flow or specific performance targets, but it specifies that organizations need to have processes that address development related practices. The processes can be mapped to the process areas, enabling the organization to track its progress against CMMI, so the processes do not necessarily map one to one.

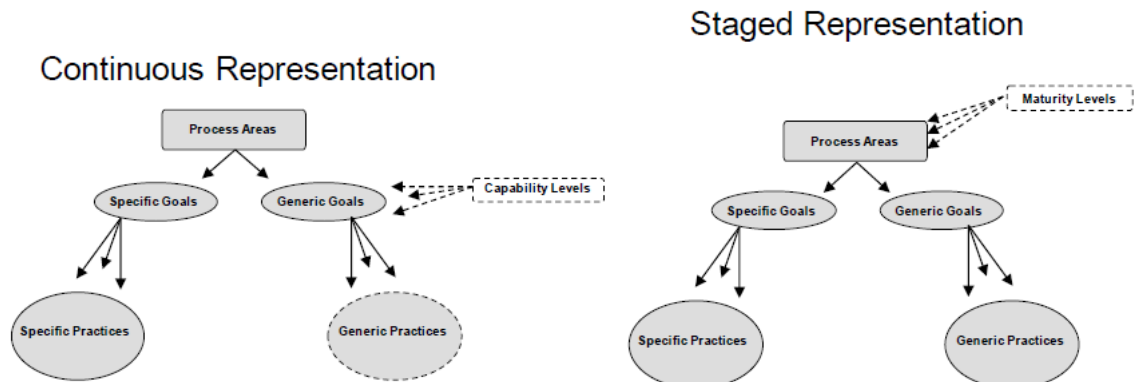
CMMI-DEV 1.3 contains 22 process areas: 16 core process areas, 1 shared process area, and 5 development specific process areas. All CMMI-DEV model practices focus on the activities of the developer organization, covering basic concepts to process improvement in any area of interest.

CMMs focus on organizations improvement using levels to describe an evolutionary path recommended for improving processes used to develop products or services. CMMI supports two improvement paths, using levels: capability levels and maturity levels. These two types correspond to two approaches to process improvement called representations:

- Continuous: uses capability levels to characterize the state of the processes relative to a process area; and
- Staged: uses maturity levels to characterize the overall state of the processes relative to the model as a whole.

Figure 2.5 illustrates the structures of the continuous and staged representations.

Figure 2.5: Structure of the Continuous and Staged Representations.



Source: Adapted from SEI (2010).

Table 2.3 compares the four capability levels to the five maturity levels; the differences are that there is no maturity level 0; there are no capability levels 4 and 5; and at level 1, the names used for capability level 1 and maturity level 1 are different.

Table 2.3: Comparison of Capability and Maturity Levels

<b>Level</b>	<b>Continuous Representation Capability Levels</b>	<b>Staged Representation Maturity Levels</b>
Level 0	Incomplete	
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4		Quantitatively Managed
Level 5		Optimizing

Source: Adapted from SEI (2010).

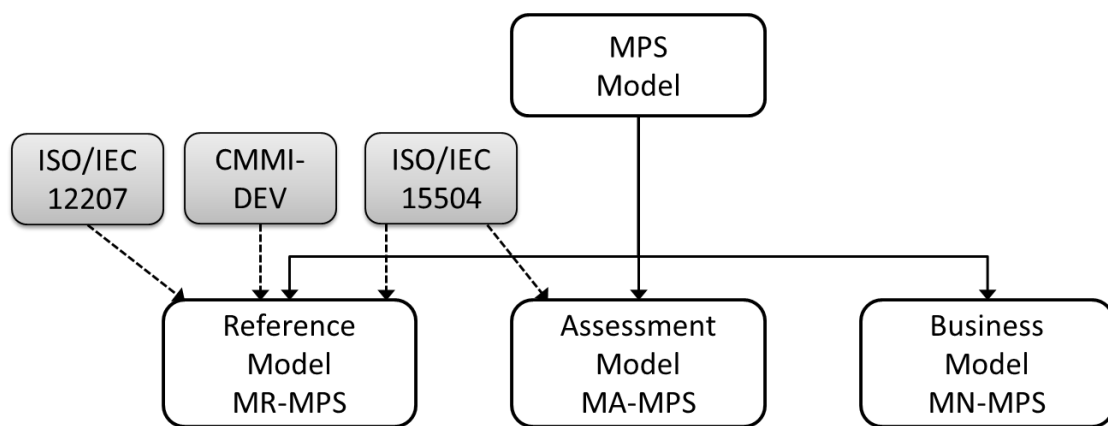
### 2.3.5 MPS.BR

The Brazilian Software Process Improvement program (MPS.BR), administered by the Software Excellence Promotion Association (SOFTEX), an entity of the

Ministry of Science, Technology, Innovations and Communications (MCTIC), is a movement for improvement and a process quality model focused on the reality of the small and medium software development market in Brazil. MPS.BR is based on CMMI, ISO/IEC 12207, and ISO/IEC 15504 standards, as well as the Brazilian market situation (SOFTEX, 2011).

MPS.BR is divided in three parts: reference model for software process improvement (MR-MPS), approach for software process improvement assessment (MA-MPS), and software process improvement business model (MN-MPS), as presented in Figure 2.6.

Figure 2.6: MPS.BR elements



Source: Adapted from SOFTEX (2011).

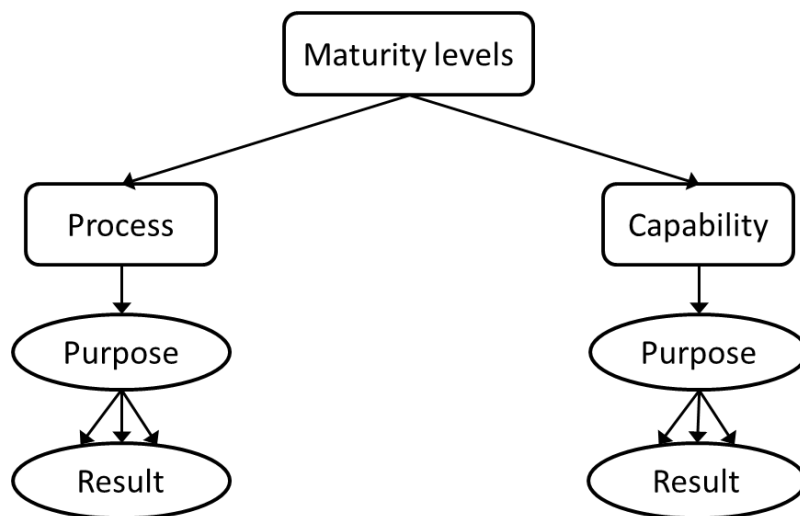
Differently from the other process standards, MPS.BR comprises seven maturity levels, from highest to lowest level: optimizing, quantitatively managed, defined, largely defined, partially defined, managed and partially managed.

MPS.BR comprises: fundamental processes (acquisition, requirements management, requirements development, technical solution, product integration, product installing and product release); organizational processes (project management, process adaptation for project management, decision analysis and resolution, risk management, organizational process assessment and improvement, project quantitatively management, causes analysis and



resolution, innovation and organization implementation); and support processes (quality assurance, configuration management, validation, measuring, verification and training). Each maturity level has process with associated capabilities, as presented in Figure 2.7.

Figure 2.7: MPS.BR model.



Source: Adapted from SOFTEX (2011).

In the Brazilian context, this model was developed to be an option with costs lower than the international standards, proportionating improvement opportunities for micro, small and medium companies.

Although CMMI is structured into 5 maturity levels and 22 process areas, while MPS.BR is organized into 7 maturity levels and 19 processes, there is an adherence between these models because of their common reference standards, ISO/IEC 12207 and ISO/IEC 15504.

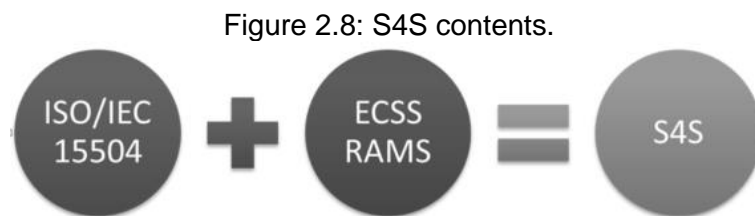
### 2.3.6 SPICE for Space (S4S)

The process assessment and improvement model defined in ECSS-Q-HB-80-02, called SPICE for Space (S4S), is the software process assessment model used by the European Space Agency (ESA) to assess the capability of ESA

contractors. S4S matches the aspects of space software from requirements definition to retirement (ECSS, 2010a), including metrics used to manage the development and to assess the quality of the development process. S4S is based and in accordance with ISO/IEC 15504, also known as SPICE, from which S4S inherits the assessment requirements, measurement framework, and the exemplary process assessment.

The S4S process assessment model (PAM), defined in ISO/IEC 15504, is composed by two main components: the process dimension and the capability dimension. The assessment model is directly mapped to the process list defined in the process reference model (PRM), based on ISO/IEC 12207, with the addition of some specific aspects from the aerospace industry.

S4S extends SPICE by adding processes and indicators related to Reliability, Availability, Maintainability and Safety (RAMS) requirements from ECSS standards, to ensure that software is developed to perform properly and safely, meeting the project's quality objectives, as presented in Figure 2.8.



Source: Author.

S4S is the main reference for the space software processes, comprising 3 process categories, divided into 9 groups of processes, containing a total of 52 processes, presented in Table 2.4, that are subdivided into activities and tasks.

Table 2.4: S4S processes.

Process category	Process group	Processes
Primary life cycle processes	Acquisition	6
	Supply	3
	Operation	2
	Engineering	12
Supporting life cycle	Supporting	12
Organizational life cycle processes	Management	7
	Process improvement	3
	Resource and infrastructure	4
	Reuse	3
Total		52

Source: Adapted from ECSS (2010a).

### 2.3.7 ISO/IEC 29110

The term “very small entity” (VSE) has been defined by ISO/IEC JTC1/SC7 Working Group 24 and subsequently adopted in the ISO/IEC 29110 process lifecycle standard as “an enterprise, organization, department or project having up to 25 people” (ISO/IEC, 2011b).

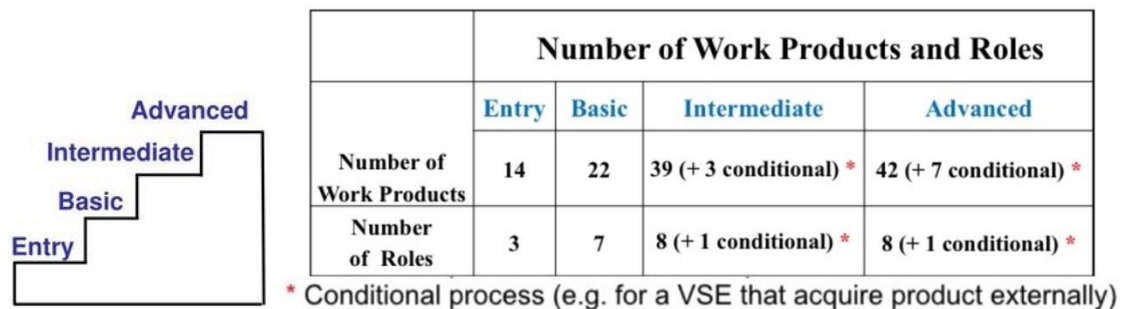
ISO/IEC 29110 series of International Standards and Technical Reports aim to assist and encourage very small software organizations in assessing and improving their software processes (O'CONNOR; LAPORTE, 2011a). Their approach (O'CONNOR; LAPORTE, 2011b) relies on the concept of ISO standardized profiles (SP) making use of pre-existing international standards, such as the software life cycle standard ISO/IEC 12207 and the documentation standard ISO/IEC 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.

In addition to 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; ALEXANDRE; O’CONNOR, 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 a form which allows its applicability 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; LAPORTE, 2010). In this context, the Generic profile group, chosen as reference in the present work, comprises a collection of four profiles (Entry, Basic, Intermediate, Advanced), as presented in Figure 2.9, proposing a progressive approach to satisfying most of VSEs as it does not imply any specific domain (ISO/IEC, 2011a).

Figure 2.9: Generic profile group’s contents.



Source: Adapted from ISO/IEC (2011a).

The descriptions of the four profiles from the Generic profile group are:

- Entry Profile: 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.
- Basic Profile: describes external or internal projects of a single application by a single team with no special risk or situational factors. To

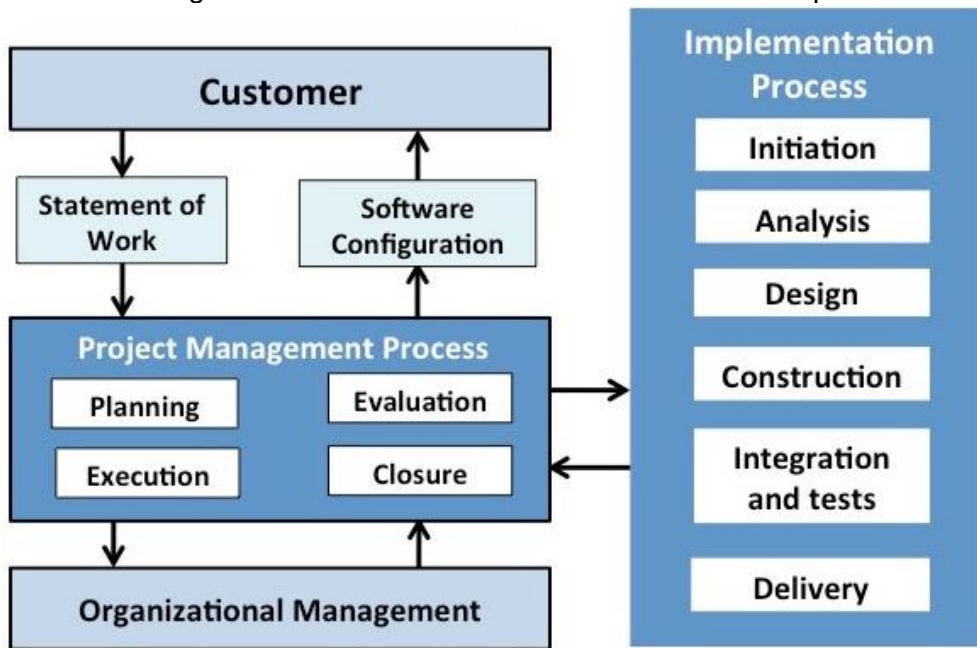
use this Profile, the VSE needs to fulfil basic entry conditions, e.g. documented project statement, feasibility analysis performed, training personnel, and infrastructure available.

- Intermediate Profile: 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 to fit with organizational goals and resources, obtain and provide necessary resources to perform, monitor and evaluate all projects.
- Advanced Profile: targeted at VSEs wanting to sustain and grow as an independent competitive system and/or software development business. In order to do so, it contains processes to move software in an orderly, planned manner into the operational status, so the system be functional in the operational environment, appropriately handle replaced or retired elements, and attends critical needs (e.g. per an agreement, per organizational policy, or for environmental, safety, and security aspects).

The Generic Group does not imply any specific domain so it can be used to provide software developers with a method to evaluate their processes with the purpose of identify possible improvements and assess their capability.

Although the Basic Profile is not meant to critical software developers, it has been chosen as reference for this work, because its two processes, Software Implementation (SI) and Project Management (PM), are defined from a subset of ISO/IEC 12207 process elements, and because it comprises a guide for ISO/IEC 15504 (SPICE), both also considered in ECSS software related standards. Figure 2.10 illustrates SI and PM processes relationship.

Figure 2.10: ISO/IEC 29110 PM and SI relationship.

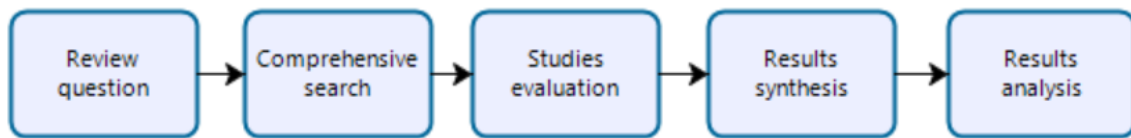


Source: ISO/IEC (2011b).

### 3 LITERATURE REVIEW

The literature review presented in this chapter comprises identification and synthesis of the papers with greater intersection with the present work. According to Pai et al. (2004), the core five steps of a literature review process are: (i) review question formulation; (ii) a comprehensive search; (iii) studies evaluation; (iv) results synthesis; and (v) results analysis. The Figure 3.1 presents the literature review process.

Figure 3.1: Literature review process.



Source: Author.

Because literature 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; GREEN, 2011).

Table 3.1 presents the PICO framework for this review.

Table 3.1: 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	-

Source: Author.

Based on Table 3.1 contents, the research question was: “What initiatives have been proposed for critical software processes tailoring in very small entities (VSEs)?”

The search was performed on the selected databases: “Science Direct”, at [www.sciencedirect.com](http://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 increases 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 as of January/2000, including journals and conference proceedings.

The number of publications identified by using the presented criteria is presented in Table 3.2.



Table 3.2: Search results – Reference date: 07/Oct/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
<b>Total</b>	<b>472</b>	<b>251</b>

Source: Author.

As Table 3.2 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.

### 3.1 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 (GORSCHER; WOHLIN, 2006), literature reports that these so-called heavy models and their evaluation methods are considered expensive by small organizations (CATERSTEEL, 2001; JOHNSON; BRODMAN, 1997; KELLY; CULLETON, 1999; LARYD; ORCI, 2000; SCHOEFFEL; BENITTI, 2015; VILLALÓN et al., 2002),

which is related to these models not being extensively deployed and their influence in software industry remains more at a theoretical level (LAPORTE; O'CONNOR; PAUCAR, 2015).

SPICE initially had several limitations. Rout et al. (2000) reviewed its evolution 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.

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.

Bujok et al. (2017) mapped standards from different domains 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 processes tailoring, mainly related to the variables used for software effort estimation (KALUS; KUHRMANN, 2013), also demonstrating the correlation between software quality metrics and aspects such as team skill (WANG; ZHAN; XU, 2006).

Kalus and 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 basis.

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

Estimation techniques may be applied for definition of 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; SHEPPERD, 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).

There are not enough studies of software estimation approaches, which support 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 and Molokken (2003) proposed a preliminary checklist, to be customized to include only relevant issues, structured on a project management framework, considering comprehensive scopes from 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 Software processes in small entities**

Given the limitations in terms of people and purchasing power that small organizations have due to their size, they face many challenges in running

process assessments (BASRI, 2011). Taking this into account, 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.

VSEs usually consider that SPI frameworks are either too expensive to deploy or do not take organizations' specific needs into consideration. Pettersson, et al. (2008) present a light weight assessment and improvement planning (iFLAP) that enables practitioners to ground 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; COLEMAN, 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. Laporte and 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 et al. (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 et al. (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).

### **3.3 Literature review analysis**

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 showed that the tailoring criteria must consider 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 mean to support the understanding of the influence factors for critical software projects in VSE context and, furthermore, to develop a notion on adequate tailoring.

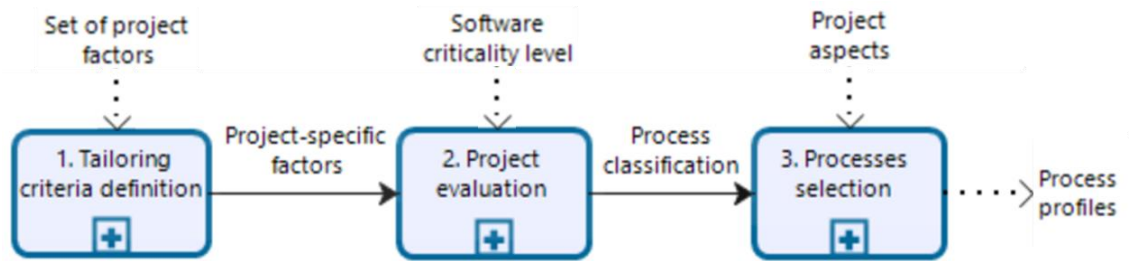
A systematic approach for process tailoring can be helpful in the VSE context, in which team-based expert estimation is usual, there is lack of documentation and new team members might not be aware of all activities and factors that should be accounted for during estimation. Frequently process tailoring 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 applicability and usability of adequate profiles for critical software in VSE, comprising simplified and flexible sets of processes according to each software project evaluation.

## 4 GENERIC APPROACH FOR PROCESS SELECTION (GAPS)

This chapter presents the Generic Approach for Process Selection (GAPS), a scheme for process selection proposed in this dissertation. GAPS development considers projects' characteristics and objectives, and is divided in 3 steps for adequate processes selection. Figure 4.1 illustrates the steps, the necessary input information and the output produced. The description of each step is given in the following sections.

Figure 4.1: GAPS development.

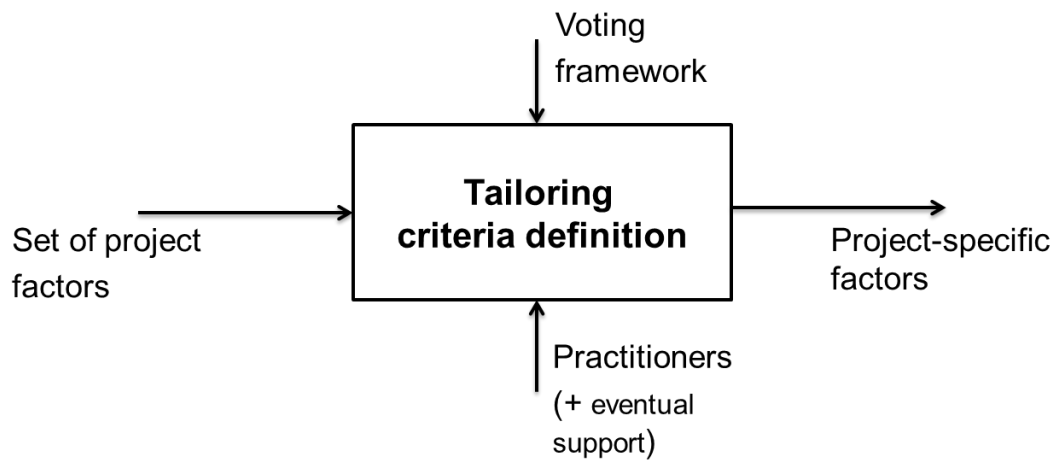


Source: Author.

### 4.1 Step 1 - Tailoring criteria definition

Defining the tailoring criteria consists of identifying project elements that impact process selection, based on the factors that may influence projects described in Table 4.1. Figure 4.2 comprises the tailoring criteria definition process, which encompasses the voting framework used in this step to determine project-specific factors.

Figure 4.2: Tailoring criteria definition.



Source: Author.

The IDEF-0 A-0 context diagram (refer to Figure 2.1: IDEF0 A-0 diagram example) elements presented in Figure 4.2 are:

- *Input (the data or object that are transformed into output):*  
*Set of project factors:* Reference list of factors that organizations can use to support the understanding of what influences their processes. In this work, the factors from Table 4.1 in section 4.1.1 are used, although there is no impeditive for using different sets of factors.
- *Control (the conditions required to produce correct output):*  
*Voting framework:* Structure used to evaluate the influence that the factors may have on processes.
- *Mechanism (the means used to perform a function):*  
*Practitioners:* project's product assurance representative (s) and/or project manager supported by experts if necessary.
- *Output (the data or objects produced):*  
*Project-specific factors:* List of factors considered relevant to the project under evaluation. Input to the next steps.



#### **4.1.1 Project factors**

The project factors are characteristics and/or situations that may have influence in the software process. Organizations can apply them in their own projects as means to support their influence and, furthermore, use them as a starting point to define appropriate tailoring criteria.

ECSS Standards present guidelines to tailor their processes based on the software criticality (ECSS, 2017a). Studies have proposed other criteria for processes tailoring, mainly related to software effort estimation (KALUS; KUHRMANN, 2013), also demonstrating the correlation between software quality metrics and aspects such as team skill (WANG; ZHAN; XU, 2006).

The pilot usages of ISO/IEC 29110 presented by Laporte et al. (2015) comprise a project classification into three categories (small, medium and large), based on characteristics such as project duration, team size, number of engineering specialties and cost.

Kalus and Kuhrmann (2013) presented a set of 49 factors that influence processes tailoring, whose names and descriptions are organized in Table 4.1, categorized in: (a) team - characteristics of the people involved in the project; (b) internal environment - organizational aspects of the project's entity; (c) external environment - context where the project takes place; and (d) objectiveness - product related features. Descriptions based on other references, mentioned opportunely in the table, support the analysis of each factor.

Table 4.1: Names and descriptions of project factors.

#	Name	Description
1	Size	The team size is an indicator for the effort of team coordination (WOLFE; CHACKO, 1983). While smaller teams can directly communicate the need for formalization increases if the team grows.
2	Distribution	Team distribution influences the interaction pattern in a project (HEEKS et al., 2001). Teams located in a single room can directly communicate while distributed teams need a more formalized communication.
3	Turnover	If a team member leaves a team, knowledge will be lost. Also, new team members affect the group dynamics (SALAS et al., 1999).
4	Previous Cooperation	If the team worked together in previous projects the need for getting familiar with the other team members may decrease which, in turn, may cause a less formal communication (XU; RAMESH, 2008).
5	Good Cooperation	If the team works in a good and collaborative manner, the need for formalized communication/documentation may decrease (XU; RAMESH, 2008).
6	Domain knowledge	<p>Little or missing knowledge with respect to the domain is a risk (XU; RAMESH, 2008).</p> <p>Note: Domain denotes a specified sphere of activity or knowledge, i.e. aerospace, aeronautics, nuclear.</p>
7	Tool Knowledge	<p>Little or missing knowledge with respect to the tools is a risk (WALLACE; KEIL, 2004).</p> <p>Note: Tools denote software that assists in the creation of new software, i.e. compilers, debuggers, visual programming tools.</p>
8	Technology knowledge	<p>Little or missing knowledge with respect to the technology is a risk (WALLACE; KEIL, 2004).</p> <p>Note: Technology denotes equipment developed from scientific knowledge for practical purposes, i.e. satellite subsystems and support equipment.</p>
9	Process knowledge	<p>Little or missing knowledge with respect to the process to be used is a risk (WALLACE; KEIL, 2004).</p> <p>Note: Process denotes the kind of practices, or methodology employed.</p>

Table 4.1: Names and descriptions of project factors.

#	Name	Description
10	Prototyping	The creation of prototypes is a strategy for risk mitigation, (BOEHM, 1991) and performance improvements, (BLACKBURN; SCUDDER; WASSENHOVE, 1996) applied to projects with a new domain or technology, volatile requirements, or with several solutions to be evaluated.
11	Clear project proposal	A clear project proposal contains basic goals and requirements (WALLACE; KEIL; RAI, 2004) crucial for the project's success (ZOWGHI; NURMULIANI, 2002).
12	Management availability	Top management is required to solve problems and to make project progress decisions. In critical project settings, missing top management availability is a risk. (WALLACE; KEIL; RAI, 2004) crucial for the project's success (ZOWGHI; NURMULIANI, 2002).
13	Management support	The top management should actively support a project, especially in critical situations (WALLACE; KEIL, 2004).
14	Project budget	Project budget influences the degree of formalism in a project. A little project budget usually implies a non-formalized process (less documentation), but also requires a strict controlling regarding to costs.
15	Project duration	Project duration influences the software process. While a "long" duration might cause risks (such as team turnover), a "short" duration is similar to a little budget (BLACKBURN; SCUDDER; WASSENHOVE, 1996).
16	Project type	Depending on the project type, different aspects of a process need to be stressed, e.g., type of requirements elicitation, addressed life cycle phases, system migration (COSTACHE; KALUS; KUHRMANN, 2011).
17	Project role	Each project has a specific role that characterizes it in relation to others. The project role influences the corresponding software process (PAASIVAARA; LASSENIUS, 2004).
18	Sub-contractors	Certain tasks can be performed by sub-contractors, so, a process should support synchronization defining what to be exchanged, and interfaces (HEEKS et al., 2001; PAASIVAARA; LASSENIUS, 2004). Also, the contractor's role changes as they become customer for sub-contractors.
19	Financial controlling	The emphasis on a financial controlling is important if the budget is critical. Intensive financial controlling results in self-contained documentation as well as increased participation in planning / decision making (KUHRMANN; TERNITÉ, 2005).

Table 4.1: Names and descriptions of project factors.

#	Name	Description	
20	Measurement	Measurement using KPIs is important, e.g., to provide the management with status information, to measure the performance and to conduct data for a company-wide controlling (OFFEN; JEFFERY, 1997). Measurement causes additional effort and also requires a more detailed reporting.	
21	Technical support	An unknown technical environment with little support may cause project risks (WALLACE; KEIL, 2004).	
22	Programing language	A new programming language can cause project risks and a programming language can influence the software architecture (COSTACHE; KALUS; KUHRMANN, 2011; PEDREIRA et al., 2007).	
23	COTS products	Integration of COTS products or components can shorten the development time, but requires attention for example to legal implications, test and integration procedures (TORCHIANO; MORISIO, 2004).	
24	Operating system	An operating system limits the available programming languages, tools and available COTS. Also, system requirements can limit the supported operating systems for the intended solution (PEDREIRA et al., 2007).	
25	Database system	Support for databases is essential, especially in business information systems (PEDREIRA et al., 2007).	
26	Tool infrastructure	Tools that shall be used in a specific phase, such as requirements engineering or coding, have to be defined. The definition of a tool infrastructure has implications, such as: tools that are not available have to be bought (KUHRMANN; KALUS, 2008; PEDREIRA et al., 2007).	
External Environment	27	Legal aspects	Projects may be critical in legal aspects, i.e. contract (LICHTENSTEIN, 2004) for several reasons. When not delivering software in time or with the functionality, claims may occur. Also, critical software requires documentation according to laws and regulations.
	28	Number of stakeholders	The higher the number of stakeholders the more time is required to negotiate all needs and requirements. Furthermore, the coordination effort increases (JIANG et al., 2009). A process should pay attention to the number of stakeholders by defining adequate communication and report pattern.
	29	Stakeholder availability	Similar to the top management, availability of stakeholders may be success factor. Its absence may be a project risk as it may cause delays (JIANG et al., 2009; WALLACE; KEIL, 2004).

Table 4.1: Names and descriptions of project factors.

#	Name	Description	
30	Stakeholder background	If the stakeholders' background is not adequate (e.g., new domain, technology, or innovative product), the process should provide different strategies, such as requirements elicitation to support for learning curves (JIANG et al., 2009).	
31	Requirements stability	The stability of requirements directly influences the entire approach (JIANG et al., 2009; WALLACE; KEIL, 2004; ZOWGHI; NURMULIANI, 2002). For example, the strategy for requirements elicitation, architecting the solution, implementation and test.	
32	Client process	If a client has its process and wants to align contractors with it, the processes have to support interfaces or procedures to comply with the client's requirements, e.g. ensuring CMMI levels (YONG; MIN; BAE, 2001).	
33	Client availability	The clients' availability influences their satisfaction since they can continuously monitor progress, i.e. agile methods propose the on-site client (HEEKS et al., 2001; HERBSLEB; MOCKUS, 2003). Regular deliveries in short cycles can compensate a missing availability.	
34	Type of contract	The type of the contract (LICHTENSTEIN, 2004) directly influences the process. For instance, fixed-price model vs. time & material leads to different strategies in handling change requests.	
35	User availability	End user availability is important during requirements engineering, integration and testing phases (HEEKS et al., 2001; HERBSLEB; MOCKUS, 2003) .	
36	User background	End user background/domain knowledge is important to decide about training. Appropriate acceptance strategies should consider the end users' knowledge (FERNÁNDEZ et al., 2012; XU; RAMESH, 2008).	
37	Trainings	Training requirements, e.g. during acceptance or deployment cause additional effort, like planning and creating material (BLACKBURN; SCUDDER; WASSENHOVE, 1996; WALLACE; KEIL; RAI, 2004).	
Objective	38	Complexity	Higher complexity causes more formal communication, configuration and change control (CAMCI; KOTNOUR, 2006; XIA; LEE, 2004).
	39	Innovation degree	The high degree of innovation may cause a more explorative approach to mitigate project risks (WALLACE; KEIL; RAI, 2004).
	40	Legacy system	Legacy system needs to be considered within requirements engineering and limits the solution, e.g., by compatibility requirements and data migration (KUHRMANN; TERNITÉ, 2005).

Table 4.1: Names and descriptions of project factors.

#	Name	Description
41	Legacy system documentation	If no documentation for a legacy system is available, higher effort in analyzing is expected. So, appropriate strategies should be provided, e.g. reverse engineering (FERNÁNDEZ et al., 2012; XU; RAMESH, 2008).
42	Domain	A domain implies standards, norms, regulations, and laws that need to be considered in a project. Depending on the domain, more potential criteria should be regarded in tailoring (PEDREIRA et al., 2007).
43	Conceptual solution	A process can contain domain-specific knowledge and best practices (ZAROOUR et al., 2015) to support the development.
44	Technical solution	The technical solution can be supported by specific contents, e.g., applications, design patterns, coding guidelines for programming languages (KUHRMANN; TERNITÉ, 2005; PEDREIRA et al., 2007).
45	Safety & Security	Safety & security is usually related to a comprehensive documentation of a project. The software process should, therefore, provide templates and indications about the documentation.
46	Hardware development	If hardware development is also part of a project, the process has also to provide corresponding artifacts (e.g., specifications and designs, test hardware, logistics) integrated in the software process (KUHRMANN; TERNITÉ, 2005).
47	Neighboring systems	The interfaces between a software system and its ecosystem need to be defined, since they directly influence the integration and test strategies (PEDREIRA et al., 2007).
48	User interface	If software has special requirements regarding the user interface, the design, implementation, and test should be part of the process (KUHRMANN; TERNITÉ, 2005).
49	System integration test	Requirements with regard to system integration should be reflected by the process, for instance, by defining integration strategies, providing a fundamental integration of project management, development, and quality assurance (KUHRMANN; TERNITÉ, 2005).

Source: Adapted from Kalus and Kuhrmann (2013).

#### 4.1.2 Voting framework for criteria selection

The adequate tailoring criteria selection takes into account specialists' knowledge. Because this approach is intended for very small entities, the voting tool is described for two professionals only, but there is no impeditive for having more specialists when available.

This approach considers the presence of the team roles presented in the Basic profile from ISO/IEC 29110 (ISO/IEC, 2011b) plus the product assurance (PA) responsible. The list of the roles comprises: PA responsible, analyst, customer, designer, programmer, project manager, technical leader and work team;

In order to provide the voting with some structure, GAPS encompasses a spreadsheet to register the voting to each factor presented in Table 4.2. In this table, the first two columns contain the name and description of each factor; the next column is used for voting, where the voter is required to fill each cell either with "YES" (impacts process tailoring) or "NO" (does not impact process tailoring). The last column must be filled either with a metric, for factors voted as "YES", or with a justification, for factors voted as "NO". Two voters must vote independently, with each voter filling one spreadsheet.

Table 4.2: Factor voting spreadsheet example

Factor		Does the item have an impact on VSE process selection? (Yes / No)	
Name	Description	Vote	(Yes) Metric / (No) Justification
Size	The team size is an indicator for the effort of team coordination. The need for formalization increases if the team grows.	Yes	Number of people.
Programing language	A new programming language can cause project risks. Furthermore, a concrete programming language can influence the software architecture in a project.	No	Project premise: new language is not to be used for critical software.

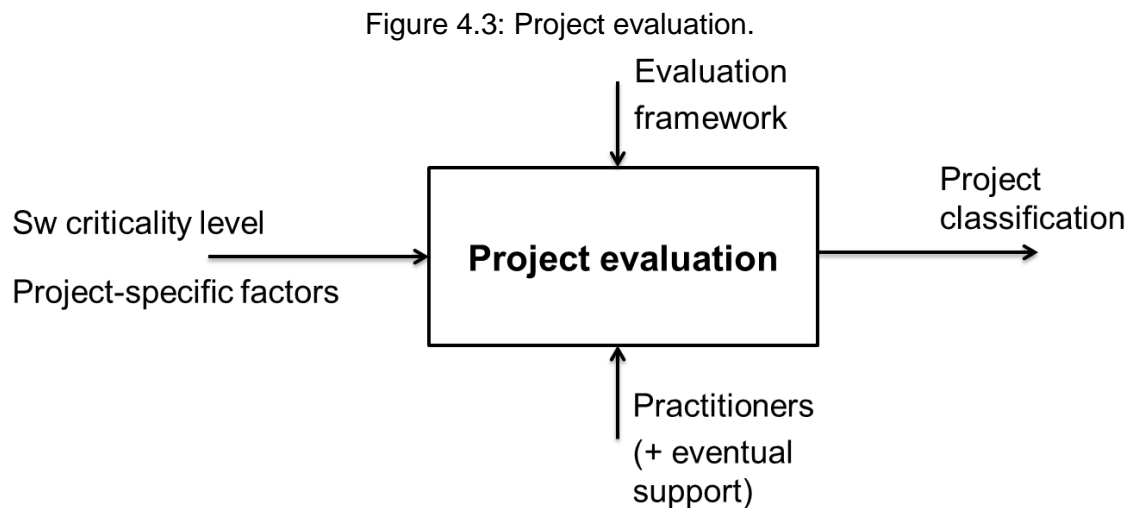
Source: Author.

After both specialists have voted, their results are compared. If the vote is different, the item is analyzed by both voters together, discussing about their rationales to reach a third, and final, joint vote. If necessary, a third person may be involved to reach a final decision.

The output of this step is the resulting set of project-specific factors, which is the set of tailoring criteria to be used on the next steps.

#### 4.2 Step 2 - Project evaluation

The project evaluation consists of obtaining a classification based on the result of assessing a project in relation to its criticality level and the influence of its project-specific factors (output from Step 1 – Tailoring criteria definition, in section 4.1), to which a structure is used. The elements involved in this step are presented in Figure 4.3.



Source: Author.

The IDEF-0 A-0 context diagram (refer to Figure 2.1: IDEF0 A-0 diagram example) elements presented in Figure 4.3 are:



- *Input (the data or object that are transformed into output):*  
*Software criticality level:* criticality level of the product being developed or modified, which comes from system-level analyses based on the severity of its possible failures consequences. The criticality assessment results in a classification of the project in one of the four different criticality categories presented in ECSS standards: A, B, C or D (refer to Table 2.1: Software criticality categories definition).  
*Project-specific factors:* list of factors considered relevant to the project, which is the output from Step 1 - Tailoring criteria definition (section 4.1).
- *Control (the conditions required to produce correct output):*  
*Evaluation framework:* structure used to evaluate the project-specific factors' level of influence, its result is a score (refer to section 4.2.1 Evaluation structure).
- *Mechanism (the means used to perform a function):*  
*Practitioners:* product assurance representative and/or project manager supported by experts if necessary.
- *Output (the data or objects produced):*  
*Project classification:* project categories related to the score from the evaluation framework and the software criticality classification.

#### **4.2.1 Evaluation structure**

The format reference for the structure is the method described in FAA Order 8110.49 Chg. 1 (FAA, 2011) used by the Federal Aviation Administration (FAA) for determining their level of involvement on projects. The level of FAA involvement (LOFI) is classified as HIGH, MEDIUM, or LOW; considering two major areas of criteria: software criticality level criteria and other relevant criteria.

#### 4.2.1.1 GAPS evaluation framework

The GAPS evaluation framework must be generated based on the set of project-specific factors (output from Step 1 – Tailoring criteria definition) with one question, along with a related metric and respective grade, for each criterion. Table 4.3 presents an example of questions with related metrics.

Table 4.3: Evaluation framework example.

	Criterion	Metric	Grade		
			0	5	10
Team	Size	# people	< 5	5 to 15	15 to 25
Project	Complexity (different technologies and/or disciplines)	# tech. / disc.	1	2 to 3	> 3
Score:			_____		

Source: Author, based on FAA (2011).

The structure presented in Table 4.3 accounts for the same possible grades (0, 5 or 10) for each item, so every criterion has the same weight in this evaluation, in which the product assurance representative or the project manager performs:

- (1) Assessment with the software team; and
- (2) Research about past performance of the organization, based on previous projects, audits, in-service problems, and other experiences.

Ideally, the determination of tailoring criteria and the criticality assessment have to be carried out and documented at the start of the project to enable the organization to plan and address the project details as early as possible.

Generating one classification for each combination of the project-specific factors would result in a great number of types. For that reason, GAPS comprises the project classification presented in Table 4.4 related to criticality categories.

Table 4.4: Software criticality related project classification.

Software criticality category (ECSS)	Project classification
D	ISO/IEC 29110 Basic
C	GAPS Basic or GAPS Intermediate
B	GAPS Intermediate or GAPS Advanced
A	GAPS Advanced or S4S

Source: Author.

As the number of software quality requirements increases according to the criticality of the software function, GAPS includes a **VSE Critical Profile group**. Similarly to the concept adopted in ISO/IEC 29110, this group comprises a collection of three profiles (**Basic**, **Intermediate**, and **Advanced**), subsets of S4S and ISO/IEC 29110 processes with a progressive approach, in which processes and conditions may be included according to the criticality and/or other criteria escalation.

This evaluation's score (s) and the software criticality level compose the two necessary axes to determine the project classification on Table 4.5, in which the scale range and the values for **X** and **Y** have to be defined by the practitioners.

Table 4.5: VSE software project profile classification.

Project profile classification				
Score	Software Criticality Level			
	A	B	C	D
$\underline{Y} < s$	S4S	GAPS Advanced	GAPS Intermediate	ISO 29110 Basic
$\underline{X} < s \leq \underline{Y}$	GAPS Advanced	GAPS Intermediate	GAPS Basic	ISO 29110 Basic
$s \leq \underline{X}$	GAPS Advanced	GAPS Intermediate	GAPS Basic	ISO 29110 Basic

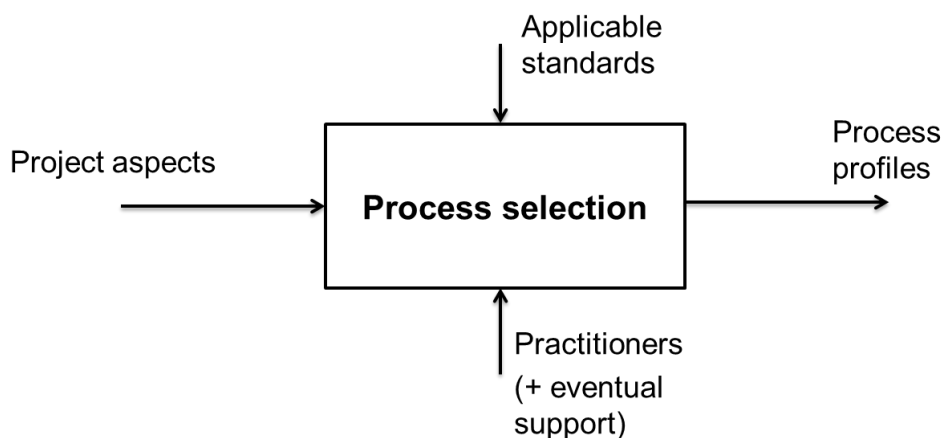
Source: Author.

Based on this classification, the VSE software project processes are determined, allowing the selection of adequate **profiles** (sets of **processes**) to be adopted.

### 4.3 Processes selection

The three profiles from GAPS (**Basic**, **Intermediate**, and **Advanced**) extend ISO's VSE definition to critical software, using the processes from ECSS-Q-HB-80-02 (S4S) and ISO/IEC 29110 Basic Profile to build sets of processes adequate to the project. Figure 4.4 presents the elements involved in this step.

Figure 4.4: Processes selection.



Source: Author.

The IDEF-0 A-0 context diagram (refer to Figure 2.1: IDEF0 A-0 diagram example) elements presented in Figure 4.4 are:

- *Input (the data or object that are transformed into output):*  
*Applicable standards:* ISO/IEC 29110 and S4S are considered applicable in this work. Other applicable standards, that contain the processes to be selected, may be considered depending on the project.
- *Control (the conditions required to produce correct output):*

*Project aspects*: aspects to be regarded according to the factors considered relevant to the project: stakeholder, project life cycle, project organization and knowledge (refer to section 4.3.1 Project aspects).

- *Mechanism (the means used to perform a function)*:

*Practitioners*: project's product assurance representative and/or project manager supported by experts if necessary.

- *Output (the data or objects produced)*:

*Process profiles*: profiles comprising the processes considered essential according to the projects' factors (refer to section 4.3.2 Process profiles).

#### **4.3.1 Project aspects**

The contents of GAPS profiles - Basic, Intermediate and Advanced - have to be defined based on their project/organization factors. Related literature (KALUS; KUHRMANN, 2013) indicates that, depending on the factors considered relevant to a project, certain aspects are to be considered when selecting the processes. The 49 relevant aspects from Table 4.6 influence aspects of the software process related to stakeholder, project life cycle, project organization and knowledge.

Table 4.6: Aspects related to the tailoring criteria.

#	Name	Stakeholder aspect	Life cycle aspect	Organization aspect	Knowledge aspect
1	Size	-	-	project documentation; number of (micro-) iterations; communication pattern	-
2	Distribution	-	-	project documentation; number of (micro-) iterations; communication pattern	-
3	Turnover	-	-	project documentation	-
4	Previous Cooperation	-	-	project documentation; communication pattern	-
5	Good Cooperation	-	-	project documentation	-
6	Domain knowledge	-	-	-	meetings/workshops; trainings; knowledge management infrastructure
7	Tool Knowledge	-	-	selection of appropriate tools w.r.t. the process's weight	trainings
8	Technology knowledge	-	prototype development	number of (micro-) iterations	trainings
9	Process knowledge	-	-	project documentation	-
10	Prototyping	-	requirements engineering; prototype development; fast feedback loops	-	meetings/workshops

Table 4.6: Aspects related to the tailoring criteria.

#	Name	Stakeholder aspect	Life cycle aspect	Organization aspect	Knowledge aspect
11	Clear project proposal	management involvement	requirements engineering; prototype development; fast feedback loops	-	meetings/workshops
12	Management availability	management involvement	-	project documentation; communication pattern	-
13	Management support	management involvement	-	project documentation; communication pattern	-
14	Project budget	-	financial project management	project documentation; communication pattern	-
15	Project duration	-	prototype development; fast feedback loops; planning pattern	-	-
16	Project type	-	requirements engineering; system architecture; integration and test	-	-
17	Project role	-	requirements engineering; system architecture; integration and test	-	-
18	Sub-contractors	-	requirements engineering; system architecture; integration and test	-	-
19	Financial controlling	management involvement	financial project management	project documentation; communication pattern	-
20	Measurement	management involvement	-	project documentation; communication pattern	-
21	Technical support	-	-	selection of appropriate tools w.r.t. the process's weight	-

Table 4.6: Aspects related to the tailoring criteria.

#	Name	Stakeholder aspect	Life cycle aspect	Organization aspect	Knowledge aspect
22	Programing language	-	system architecture; integration and test	-	trainings
23	COTS products	-	integration and test; prototype development; fast feedback loops	-	-
24	Operating system	-	system architecture; integration and test	-	-
25	Database system	-	requirements engineering; system architecture; integration and test; prototype development; fast feedback loops	-	-
26	Tool infrastructure	-	-	selection of appropriate tools w.r.t. the process's weight	-
27	Legal aspects	-	requirements engineering	project documentation	-
28	Number of stakeholders	customer involvement; end user involvement	-	project documentation; communication pattern	-
29	Stakeholder availability	customer involvement; end user involvement	fast feedback loops	project documentation; communication pattern	-
30	Stakeholder background	customer involvement; end user involvement	-	-	trainings; knowledge management infrastructure; meetings/workshops



Table 4.6: Aspects related to the tailoring criteria.

#	Name	Stakeholder aspect	Life cycle aspect	Organization aspect	Knowledge aspect
31	Requirements stability	-	prototype development; fast feedback loops; requirements engineering	-	-
32	Client process	-	requirements engineering; integration and test; fast feedback loops	-	-
33	Client availability	customer involvement; end user involvement	fast feedback loops	-	-
34	Type of contract	customer involvement; end user involvement	-	project documentation; communication pattern	-
35	User availability	end user involvement	requirements engineering; system architecture; integration and test; fast feedback loops; prototype development	-	-
36	User background	end user involvement	fast feedback loops; prototype development	-	meetings/workshops; trainings
37	Trainings	-	-	-	trainings
38	Complexity	-	requirements engineering; system architecture	project documentation; communication pattern	meetings/workshops
39	Innovation degree	-	prototype development; fast feedback loops	-	-
40	Legacy system	-	prototype development; integration and test; requirements engineering; system architecture	-	-

Table 4.6: Aspects related to the tailoring criteria.

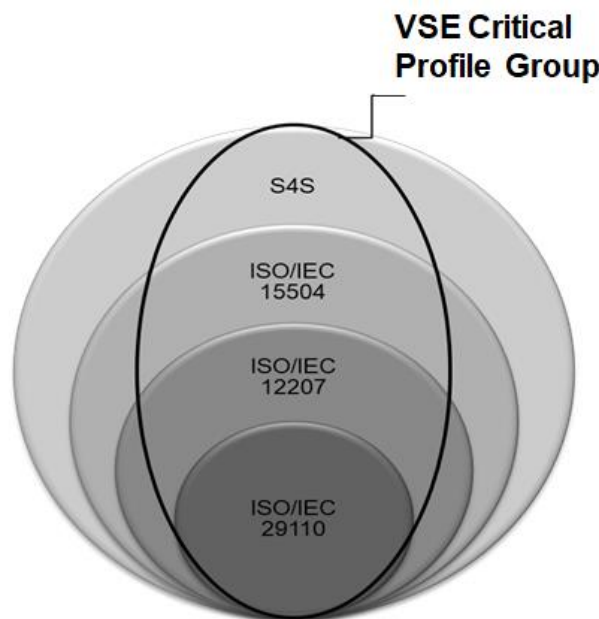
#	Name	Stakeholder aspect	Life cycle aspect	Organization aspect	Knowledge aspect
41	Legacy system documentation	-	prototype development; integration and test; requirements engineering; system architecture	-	-
42	Domain	-	requirements engineering; system architecture	-	-
43	Conceptual solution	-	requirements engineering; system architecture	-	meetings/workshops
44	Technical solution	-	requirements engineering; system architecture	-	-
45	Safety & Security	-	requirements engineering	project documentation; communication pattern	-
46	Hardware requirements	-	requirements engineering; system architecture; integration and test	-	-
47	Neighboring systems	-	requirements engineering; system architecture; integration and test	-	-
48	User interface	-	requirements engineering; integration and test	-	-
49	System integration test	-	requirements engineering; system architecture; integration and test	selection of appropriate tools w.r.t. the process's weight	-

Source: Author, based on Kalus et al. (2013).

### 4.3.2 Process profiles

The VSE Critical Profile group from GAPS is based on the processes from ECSS-Q-HB-80-02 (S4S), which encompasses the processes from ISO/IEC 29110, ISO/IEC 12207 and ISO/IEC 15504, as illustrated in Figure 4.5.

Figure 4.5: VSE critical profile group processes.



Source: Author.

Performing a set of processes lighter than the set defined according to the criticality level may affect the project management along with the product's dependability and safety. The risk taken by not performing a process must be analyzed and its consequences assessed. Any further tailoring is therefore associated to a risk analysis specific of each project. However, for the minimum set of processes for the critical profile group, the premise considered is that the following concepts, along with their related processes cannot be excluded:

- a. Definition of a development approach;
- b. Elicitation and reviewing of software requirements;
- c. Production, validation and acceptance of software;
- d. Quality assurance and
- e. Configuration management.

As the S4S process assessment model (refer to section 2.3.6), defined in ISO/IEC 15504 (refer to section 2.3.2), is composed by two main components: the capability dimension, a series of process attributes representing the measurable characteristics of a process; and the process dimension, defined by the statements of process purpose and outcomes.

#### **4.3.2.1 Capability dimension**

The capability dimension defines a measurement scale for the capability of any process (refer to section 2.3.2.2 Capability dimension). S4S considers ISO/IEC 15504 six-point scale for representing the capability level at which the process is performed. However, as one of the purposes of this work is to simplify the processes given the VSE context, the two points scale considered herein is derived from the scale proposed by Lahoz et al. (2015):

- Level 0: Not performed (incomplete) process;
- Level 1: Performed process.

For the three profiles within this VSE Critical profile group, the number of processes is increased from GAPS Basic to GAPS Advanced, while the capability dimension is kept at the same level.

#### **4.3.2.2 Process dimension**

The key reference for the critical software processes, S4S, mainly considers the requirements from ECSS-E-ST-40 (ECSS, 2009a) and ECSS-Q-ST-80 (ECSS, 2017a) to define 3 process categories, divided into 9 groups of processes, containing a total of 52 processes, subdivided into activities and tasks. The main difference between ECSS standards and ISO/IEC 29110 is that ECSS targets organizations of all sizes developing software from non-critical to highly critical software, while ISO/IEC 29110 is intended for VSE developing non-critical software only (LARRUCEA et al., 2016).

When adopting an S4S-compatible nomenclature, ISO/IEC 29110 defines 2 categories, or groups of processes, that contain a total of 10 processes, also subdivided into activities and tasks. Table 4.7 comprises S4S list of processes.

Table 4.7: S4S Processes.

Process category	Process group	Process	#
Primary life cycle processes	Acquisition	ACQ.1 Acquisition preparation	6
		ACQ.2 Supplier selection	
		ACQ.3 Contract agreement	
		ACQ.4 Supplier monitoring	
		ACQ.5 Customer acceptance	
		ACQ.6 Contract maintenance	
	Supply	SPL.1 Supplier tendering	3
		SPL.2 Product release	
		SPL.3 Product acceptance support	
	Operation	OPE.1 Operational use	2
		OPE.2 Customer support	
	Engineering	ENG.1 Requirements elicitation	12
		ENG.2 System requirements analysis	
		ENG.3 System architecture design	
		ENG.4 Software requirements analysis	
		ENG.5 Software Design	
		ENG.6 Software construction	
		ENG.7 Software integration	
ENG.8 Software testing			
ENG.9 System integration			
ENG.10 System testing			
ENG.11 Software installation			
ENG.12 SW and system maintenance			
Supporting life cycle	Supporting	SUP.1 Quality assurance	12
		SUP.2 Verification	
		SUP.3 Validation	
		SUP.4 Joint review	
		SUP.5 Audit	
		SUP.6 Product evaluation	
		SUP.7 Documentation	
		SUP.8 Configuration management	
		SUP.9 Problem resolution	
		SUP.10 Change request management	
		SUP.11 Safety and dependability assurance (*)	
		SUP.12 Independent software verification and validation (*)	

Table 4.7: S4S Processes.

Process category	Process group	Process	#
Organizational life cycle processes	Management	MAN.1 Organizational alignment	7
		MAN.2 Organizational management	
		MAN.3 Project management	
		MAN.4 Quality management	
		MAN.5 Risk management	
		MAN.6 Measurement	
		MAN.7 Information management	
	Process improvement	PIM.1 Process establishment	3
		PIM.2 Process assessment	
		PIM.3 Process improvement	
	Resource and infrastructure	RIN.1 Human resources management	4
		RIN.2 Training	
		RIN.3 Knowledge management	
RIN.4 Infrastructure			
Reuse	REU.1 Asset management	3	
	REU.2 Reuse program management		
	REU.3 Domain engineering		
(*) : added in S4S		<b>Total</b>	<b>52</b>

Source: Adapted from ECSS (2010a).

Table 4.8 presents the list of processes from ISO/IEC 29110 Basic Profile. In the context of this work, ISO/IEC 29110 *processes* were reclassified as *process groups* and, consequently, *activities* were reclassified as *processes*.

Table 4.8: ISO/IEC 29110 Basic Profile processes.

Process (= Process groups)	Activities (= Processes)	Number of processes
Project Management	PM.1 Project Planning	4
	PM.2 Project Plan Execution	
	PM.3 Project Assessment and Control	
	PM.4 Project Closure	
Software Implementation	SI.1 Software Implementation Initiation	6
	SI.2 Software Requirements Analysis	
	SI.3 Software Architectural and Detailed Design	
	SI.4 Software Construction	
	SI.5 Software Integration and Tests	
	SI.6 Product Delivery	
<b>Total</b>		<b>10</b>

Source: Adapted from ISO/IEC (2011).

## 5 CRITICAL SPACE PROFILES (CSP)

This chapter describes the development of the Critical Space Profiles (**CSP**), a set of profiles developed for VSE developing space software. **CSP** is generated using **GAPS** with information from three already concluded space projects to generate a version of **GAPS' VSE Critical Profile Group** for the space context, determining the contents of the related profiles (**Basic**, **Intermediate**, and **Advanced**).

The projects from the space area were subject to the following data collection methods:

1. Interview: to understand the organization and context;
2. Survey: to get information about the project's characteristics; and
3. Research: to get documented information about the projects and respective organizations.

### 5.1 Projects

The three software projects are: (1) On-board Data Handling Application, (2) Ground Control, and (3) Ground station application for remote sensing payload. These projects were selected for being representative of space software in VSE, as they cover three facets from space projects (space, control, and application segments) and were developed for different missions by different organizations that fit the description of very small entity (up to 25 people).

#### 5.1.1 Project 1: On-board data handling application

The development team of the On-Board Data Handling Application was composed of 7 members in different roles: 2 architects/developers, 1 developer, 1 tester/ test architect, 1 tester, 1 project manager, and 1 part-time PA representative. The product was developed for over 2 years, using C++ program language and consisted of approximately 20 thousand lines of code.

The functions performed by the On-Board Data Handling Application software include navigation, health monitoring of on-board equipment, command processing, service and payload subsystems management, and communications. In general, embedded space systems require real-time control and high reliability, and for this reason, on-board satellite systems must perform services compatible with other elements of the space vehicle and the ground systems.

### **5.1.2 Project 2: Ground control**

The development team of Ground Control software was composed of 7 members in different roles: 2 architects/developers, 2 developers, 1 tester/ test architect, 1 tester and 1 part-time PA representative. The product was developed for over 4 years, using C++ program languages and consists of approximately 280 thousand lines of code.

The Ground Control software targets, mainly, control of: subsystems (platform and payload), attitude, and orbit. Generally, payloads and platform subsystems do not require real-time control, except for switching operation modes or handling anomalies. Satellites typically fly autonomously until there is a need to command a change in operation mode or an anomaly forces them to automatically enter degraded operation. Currently, Ground Control has responsibility upon orbit control and maintenance of satellites. Computational systems are widely used on ground to support satellite control functions, distributed amongst Control Centers, Mission Centers and Earth Stations. Control Centers have the computational systems for mission operation, contemplating the main tasks necessary for operation and control of a satellite mission, such as orbit prediction and propagation; flight plan preparation; receiving and storing platform telemetry; implementation of the flight plan in real time.



### **5.1.3 Project 3: Application for remote sensing payload**

The development team of the application software for remote sensing payload was composed of 8 members in different roles: 1 architect, 3 developers, 1 tester/ test architect, 1 tester, 1 project manager and 1 part-time PA representative. The product was developed for over 3 years, using C++ and Python program languages and consists of approximately 120 thousand lines of code.

These software systems host the mission databases and the necessary means to for recording, processing and dissemination of data, such as imagery from Earth Observation (EO) satellites, to the users responsible for embedded equipment in the space vehicles. The complete system for remote sensing satellite comprises hardware and software architecture designed for generating the intended product with efficiency and quality, encompassing data ingestion and recording subsystems, with the possibility of remote access.

## **5.2 Applying GAPS**

The GAPS approach was applied following the process described in chapter 4. The practitioners involved were two Product Assurance specialists, each with over ten years' experience in the aerospace area and over five years' experience specific in space product assurance.

### **5.2.1 Step 1 - Tailoring criteria definition – space context**

The voting process previously described in Table 4.2 – vote (Yes or No); and either a metric (Yes) or a reason (No) – was used to the list of 49 factors from Table 4.1. This voting process resulted in the selection of 10 factors considered impacting to process tailoring. Table 5.1 presents the list of factors highlighting the chosen items (Yes).

Table 5.1: Tailoring criteria selection.

	Name	Vote	Yes: metric / No: why
<b>Team</b>	Size	Yes	Number of people
	Distribution	Yes	Number of "plants".
	Turnover	No	Considered in "Previous cooperation".
	Previous Cooperation	Yes	Time working together, in years.
	Good Cooperation	No	Absence of objective metrics.
	Domain knowledge	Yes	Experience with the domain, in years.
	Tool knowledge	Yes	Experience with the tool, in years.
	Technology knowledge	Yes	Experience w/ technology, in years.
	Process knowledge	Yes	Experience w/ process, in years.
<b>Internal Environment</b>	Prototyping	No	Within project revisions scope / newness evaluation.
	Clear project proposal	No	Within project revisions scope.
	Management availability	No	Assumed as premise.
	Management support	No	Assumed as premise.
	Project budget	Yes	Percentage of overall project (relative importance).
	Project duration	No	Process selection does not change.
	Project type	No	Considered in the criticality determination.
	Project role	No	Considered in the criticality determination.
	Sub-contractors	No	The process must be performed independently of the role in the customer-supplier chain.
	Financial controlling	No	Within project revisions scope / budget.
	Measurement	No	Within project revisions scope.
	Technical support	No	Considered in the team technical knowledge.
	Programing language	No	Considered in innovation degree.
	COTS products	No	Considered in the criticality determination.
Operating system	No	Assumed as premise / technology knowledge.	
Database system	No	Assumed as premise / tool knowledge.	
Tool infrastructure	No	Assumed as premise / tool knowledge.	
<b>External Environment</b>	Legal aspects	No	Considered in the criticality determination.
	Number of stakeholders	No	Considered in team and complexity.
	Stakeholder availability	No	Considered in team and complexity.
	Stakeholder background	No	Considered in team knowledge.
	Requirements stability	No	Assumed as premise: requirements frozen.

Table 5.1: Tailoring criteria selection.

	Name	Vote	Yes: metric / No: why
	Client process	No	Considered in team process knowledge.
	Client availability	No	Within project revisions scope.
	Type of contract	No	Considered in the criticality determination.
	User availability	No	Considered as premise: development follow up
	User background	No	Assumed as premise.
	Trainings	No	No additional processes.
<b>Objectiveness</b>	Complexity	Yes	Number of disciplines/ technologies involved
	Innovation degree	Yes	Percentage of new technology according to experts' evaluation.
	Legacy system	No	Considered in the degree of innovation.
	Legacy system documentation	No	Considered in team knowledge.
	Domain	No	Assumed as premise / criticality determination.
	Conceptual solution	No	Considered in team knowledge.
	Technical solution	No	Considered in team knowledge.
	Safety & Security	No	Considered in the criticality determination.
	Hardware development	No	No additional processes.
	Neighboring systems	No	Considered in complexity evaluation.
	User interface	No	No additional processes.
	System integration test	No	Assumed as premise.

Source: Author.

Because software projects may vary within the space context, a balance between generic and specific was considered to obtain an appropriate factors set. After the selection of applicable criteria, the coherence and consistency of the overall set of criteria were reviewed to mitigate the risk of conflict, duplication, or lack of necessary characteristic.

The resulting list comprises approximately 20% of the total of the factors evaluated from the research material. This set of criteria, applicable for profiles selection in VSE software projects, is not exhaustive and can be completed according to project needs. Some elements of this list are imposed to the project, whereas the others are subject to choice.

### 5.2.2 Step 2 - Projects evaluation – space context

After establishing the selected criteria for evaluating a software project in this context, they are assessed as shown in the framework presented in Table 5.2, comprising project factors divided in the categories product and project, comprising items, metrics and grades (0; 5 or 10).

Table 5.2: Evaluation framework.

		Criterion	Metric	Grade		
				0	5	10
<b>1. Team</b>	1.1 Size	# people	< 5	5 to 15	16 to 25	
	1.2 Distribution	# places	1	2	> 2	
	1.3 Previous cooperation	Time (years)	> 4	2 to 4	< 2	
	1.4 Domain knowledge (Average experience)	Time (years)	> 4	2 to 4	< 2	
	1.5 Tool knowledge (Average experience)	Time (years)	> 4	2 to 4	< 2	
	1.6 Technology knowledge (Average experience)	Time (years)	> 4	2 to 4	< 2	
	1.7 Process knowledge (Average experience: critical software or assessments, i.e. CMM, ISO 9001)	# projects	> 2	1 to 2	0	
<b>2. Project</b>	2.1 Project budget (Percentage of the whole system)	Project %	< 10	10 to 30	> 30	
	2.2 Complexity (different technologies and/or disciplines)	# tech. / disc.	1	2 to 3	> 3	
	2.3 Innovation degree (design novelty and new technology use)	Software %	< 10	10 to 30	> 30	
		<b>Score:</b>	_____			

Source: Author.

The grading results from applying the evaluation framework to the three cases are presented in Table 5.3.

Table 5.3: Case projects framework grading results.

	Criteria	Metric	Grade			Project		
			0	5	10	1	2	3
1. Team	1.1 Size	# people	< 5	5 to 15	16 to 25	5	5	5
	1.2 Distribution	# places	1	2	> 2	0	0	5
	1.3 Previous cooperation	Time (years)	> 4	2 to 4	< 2	5	0	0
	1.4 Domain knowledge	Time (years)	> 4	2 to 4	< 2	0	0	0
	1.5 Tool knowledge	Time (years)	> 4	2 to 4	< 2	5	0	0
	1.6 Technology knowledge	Time (years)	> 4	2 to 4	< 2	5	0	0
	1.7 Process knowledge	# projects	> 2	1 to 2	0	5	0	5
2. Project	2.1 Project budget	Project %	< 10	10 to 30	> 30	5	10	0
	2.2 Complexity	# tech./disc.	1	2 to 3	> 3	10	10	10
	2.3 Innovation degree	SW %	< 10	10 to 30	> 30	10	10	5
Score:						50	35	30

Source: Author.

For the space software, the score range for project classification was defined with the values presented in Table 5.4.

Table 5.4: Project classification for the space context.

Project profile classification				
Score	Software Criticality Level			
	A	B	C	D
$70 < s$	S4S	GAPS Advanced	GAPS Intermediate	ISO 29110 Basic
$35 < s \leq 70$	GAPS Advanced	GAPS Intermediate	GAPS Basic	ISO 29110 Basic
$s \leq 35$	GAPS Advanced	GAPS Intermediate	GAPS Basic	ISO 29110 Basic

Source: Author.

Therefore, applying the grading from Table 5.3 to the project classification from Table 5.4, results in the following classification:

- Project 1: criticality level A and  $35 < s \leq 70$ : GAPS Advanced;
- Project 2: criticality level B, and  $s \leq 35$ : GAPS Intermediate; and
- Project 3: criticality level C, and  $s \leq 35$ : GAPS Basic.

### **5.2.3 Step 3 - Process selection – space context**

By using the GAPS approach, each organization is able to select the processes relevant to their projects. Nevertheless, as reference for direct application, this section comprises the baselines for the three VSE Critical Profiles (Basic, Intermediate and Advanced), generated based on the presented projects analyses and the understanding of the influence of project factors on processes.

As determined in GAPS, the minimum set of processes for the critical profile group must comprise the following concepts, along with their related processes:

- a. Definition of a development approach;
- b. Elicitation and reviewing of software requirements;
- c. Production, validation and acceptance of software;
- d. Product assurance and
- e. Configuration management.

The processes listed for each profile are described with their original codes and names, with a three letters code for the S4S processes and a two letters code for the processes from ISO/IEC 29110. Moreover the processes related to the concepts listed above (a. to e.) are highlighted in Table 5.5.

- **GAPS Basic critical profile**

GAPS Basic Critical Profile is the first of the Critical Profile Group to be described in this work, with its processes divided in 3 process groups belonging to S4S main process categories, as shown in Table 5.5, totalizing 15 processes.

Table 5.5: Processes for GAPS Basic Critical Profile.

Process categories	Process groups	Process	Number of processes
Primary life cycle processes	Engineering	ENG.1 Requirements elicitation	8
		SI.1 Software Implementation Initiation	
		SI.2 Software Requirements Analysis	
		SI.3 Software Architectural and Detailed Design	
		SI.4 Software Construction	
		SI.5 Software Integration and Tests	
		SI.6 Product Delivery	
		ENG.8 Software testing	
Supporting life cycle	Supporting	SUP.1 Quality assurance	3
		SUP.8 Configuration management	
		SUP.9 Problem resolution	
Organizational life cycle processes	Management	PM.1 Project Planning	4
		PM.2 Project Plan Execution	
		PM.3 Project Assessment and Control	
		PM.4 Project Closure	
		<b>Total</b>	15

Source: Author.

- **GAPS Intermediate critical profile**

GAPS Intermediate Critical Profile is the second of the Critical Profile Group to be described in this work, comprising the 15 processes from the GAPS Basic Critical Profile, plus the 6 others shown in Table 5.6, totalizing 21 processes.

Table 5.6: Processes for Intermediate GAPS Critical Profile.

Process categories	Process groups	Process	Number of processes
Processes from GAPS Basic Critical Profile			15
Primary life cycle processes	Engineering	ENG.12 Software and system maintenance	1
Supporting life cycle	Supporting	SUP.2 Verification	4
		SUP.4 Joint Review	
		SUP.7 Documentation	
		SUP.11 Safety and dependability assurance	
Organizational life cycle processes	Resource and Infrastructure	RIN.1 Human resources management	1
		<b>Total</b>	21

Source: Author.

- **GAPS Advanced critical profile**

GAPS Advanced Critical Profile is the third of the Critical Profile Group to be described in this work, comprising the 21 processes from the GAPS Intermediate Critical Profile, plus the 4 others shown in Table 5.7, totalizing 25 processes.

Table 5.7: Processes for GAPS Advanced Critical Profile.

Process categories	Process groups	Process	Number of processes
Process from GAPS Intermediate Critical Profile			21
Supporting life cycle	Supporting	SUP.10 Change Request management	2
		SUP.12 Independent software verification and validation	
Organizational life cycle processes	Management	MAN.7 Information management	2
	Resource and Infrastructure	RIN. 3 Knowledge management	
<b>Total</b>			<b>25</b>

Source: Author.

- **Additional processes**

Depending on what factors have been elected as impacting to the project, the 4 processes listed in Table 5.8 are recommended.

Table 5.8: Additional processes.

Process categories	Process groups	Process	Number of processes
Supporting life cycle	Supporting	SUP.5 Audit	1
Organizational life cycle processes	Management	MAN.2 Organizational management	3
	Resource and Infrastructure	RIN.3 Knowledge management	
	Reuse	REU.3 Domain engineering	
<b>Total</b>			<b>4</b>

Source: Author.



## 6 CONCLUSION

This work identified and summarized the processes adopted by ISO/IEC 29110 Basic Profile and ECSS S4S, whose processes present similarities, representing opportunities to use them complementarily.

Software process in VSE critical projects can be selected based on the evaluation described in **GAPS**, which contributes with the selection of tailoring criteria, which we extracted from literature and we backed up with analysis on their relevance. The criteria selection is a manner to support the understanding of the influence factors for critical software projects in VSE context and, furthermore, to develop a notion on selection of adequate tailoring criteria.

Although our results do not allow for deriving complete sets of processes based on the implications of tailoring criteria on all possible software projects, **GAPS** supports the definition of tailoring criteria as the necessary variability of software processes is inherently given by the complexity of software. The contributed means to determine tailoring criteria and the processes selection provided herein, therefore, lay the foundation for further work.

The **GAPS** approach is meant to be applicable to any project, yet it is important to recognize that the subset of criteria selected can vary depending on the project phase. The early phases in a project lifecycle most often do not need a high percentage of the requirements available in the standards to be made applicable to achieve their objective. However, in order to establish an overall view of the phasing in of requirements, it is good practice that the initial selection of applicable processes covers all project phases including the development phase, which is typically the most demanding. With this initial selection established, appropriate and coherent subsets of the processes to be made applicable during the course of project implementation can then be selected to match the specific needs of the project phases.

Moreover, the applicability of **GAPS** to real cases has been analyzed. The resultant customized approach (**CSP**) comprises selected VSE projects factors, based on which a framework for software evaluation has been generated and used as means to support the selection of appropriate processes, considering

their specific implication on space projects, then a related critical profile group consisting of three profiles was defined. The results allow the project evaluation for a selection of adequate sets of processes (profiles) that consider the VSE project aspects. **CSP** is available to be used as baseline to direct application in software projects in the space context.

Consequently, the proposed objectives have been fulfilled by delivering a 3-steps process selection approach (**GAPS**) and a customized version of it with 3 profiles ready for use in VSE (**CSP**).

## 6.1 Limitations

This work does not aim to be exhaustive on defining the profiles to be used according to the resultant evaluation, which is based on specific conditions and, thus, has limitations.

Another possible limitation is having only one specialist in charge to use the proposed approach (**GAPS**) to select the criteria and the processes to comprise **CSP**. Although having more than one specialist participate in identifying and selecting the criteria and processes may reduce the possibility of excluding relevant ones, the strict application of a well-defined process help in identifying the appropriate items. To further increase the validity of the findings, **GAPS** includes a second practitioner performing these selections not affected by the findings of the first, however more evaluations may be necessary to validate the approach.

This evaluation approach is meant to be applicable to any project, yet it is important to recognize that the subset of criteria selected can vary depending on the project phase. The early phases in a project lifecycle most often do not need a high percentage of the requirements available in the standards to be made applicable to achieve their objective. However, in order to establish an overall view of the phasing in of requirements, it is good practice that the initial selection of applicable processes covers all project phases including the development phase, which is typically the most demanding. With this initial selection established, appropriate and coherent subsets of the processes to be

made applicable during the course of project implementation can then be selected to match the specific needs of the project phases.

## 6.2 Future work suggestion

Further studies are necessary to continue this work on the use of adequate profiles, comprising simplified and flexible sets of processes according to each software project evaluation to provide evidence on their feasibility with evaluation of their completeness, applicability and usability for critical software in VSE, detailing the standard processes used in this work, establishing prioritization of processes and criteria, implementing GAPS and in contexts other than space and using CSP with more space projects, including the contexts of the ESA's Critical VSE Focus Group and INPE's projects.

## 6.3 Published works

Within the scope of this dissertation, the following works have been published:

- Workshop em Engenharia e Tecnologia Espaciais (WETE).

Type: Conference / oral presentation.

Title: Criteria proposal for critical software development processes selection for space projects in Very Small Entities.

Year: 2018.

Link: <http://mtc-m16d.sid.inpe.br/rep/sid.inpe.br/mtc-m16d/2018/10.11.17.58?metadatarepository=sid.inpe.br/mtc-m16d/2018/10.11.17.58.54&ibiurl.backgroundlanguage=pt&ibiurl.requiredsite=mtc-m16d.sid.inpe.br+806&requiredmirror=sid.inpe.br/mtc-m19@80/2009/08.21.17.02.53&searchsite=bibdigital.sid.inpe.br:80&searchmirror=sid.inpe.br/bibdigital@80/2006/11.11.23.17&choice=briefTitleAuthorMisc>

- ESA Software Product Assurance and Engineering Workshop.

Type: Conference / oral presentation.

Title: An approach for classifying critical space software projects developed in Very Small Entities.

Year: 2019.

Link: <https://atpi.eventsair.com/QuickEventWebsitePortal/software-pa-workshop-2019/home/ExtraContent/ContentSubPage?page=1&subPage=7>

- A Aplicação do Conhecimento Científico nas Engenharias 3.

Type: Book / paper.

Year: 2019.

Title: APPROACH PROPOSAL FOR CRITICAL SOFTWARE PROCESSES SELECTION FOR SPACE PROJECTS IN VERY SMALL ENTITIES (VSE). ISBN 93243, DOI 10.22533.

- International Journal of Advanced Engineering Research and Science (IJAERS).

Type: Journal / paper.

Year: 2019.

Title: Critical Software Processes Tailoring and Very Small Entities: A Literature Review.

ISSN 2349-6495(P)| 2456-1908(O), DOI 10.22161/ijaers.

## REFERENCES

- ALBUQUERQUE, I. S. **Modelo para o gerenciamento da configuração e gerenciamento da informação e documentação do programa espacial brasileiro**. 2011. 152p. Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Gerenciamento de Sistemas Espaciais) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.
- ALEXANDRE, S.; RENAULT, A.; HABRA, N. OWPL: a gradual approach for software process improvement in SMEs. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATION, 32, 2006. **Proceedings...** IEEE Computer Society. 2006. p. 328-335.
- ANACLETO, A. et al. **MARES**: a method for process assessment in small software companies. Itajaí: Universidade do Vale do Itajaí, 2004. (Technical Report LPQS0012004).
- ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE 0 SOFTEX. **MPS.BR**: melhoria de processo do software brasileiro: guia geral. Campinas: SOFTEX, 2011.
- BASRI, S. O. R. V. A study of software development team dynamics in SPI. **Communications in Computer and Information Science**, v.172, p.143-154, 2011.
- BLACKBURN, J. D.; SCUDDER, G. D.; WASSENHOVE, L. N. V. Improving speed and productivity of software development: a global survey of software developer. **IEEE Transactions on Software Engineering**, v. 22, n. 12, p. 875-885, Dec. 1996. DOI: 10.1109/32.553636.
- BOEHM, B. W. Software risk management: principles and practices. **IEEE Software**, v. 8, n. 1, p. 32-41, Jan. 1991. DOI: 10.1109/52.62930.
- BRUHN, M. et al. **MSME finance gap**: assessment of the shortfalls and opportunities in financing micro, small, and medium enterprises in emerging markets. Washington: World Bank Group, 2017.
- BUJOK, A. B. et al. Approach to the development of a Unified Framework for Safety Critical Software Development. **Computer Standards & Interfaces**, v. 54, pt. 3, p. 152-161, Nov. 2017. DOI: 10.1016/j.csi.2016.11.013.
- BURTON, J. A. **A software risk management capability model for medical device software**. Thesis (PhD) - University of Limerick, Limerick, Ireland, 2008.

CAMCI, A.; KOTNOUR, T. Technology complexity in projects: does classical project management work? In: TECHNOLOGY MANAGAMENT FOR THE GLOBAL FUTURE CONFERENCE, 2006, Istanbul, Turkey. **Proceedings...** IEEE, 2006. p. 2181-2186.

CASS, A. et al. SPICE for SPACE: a process assessment and improvement method for space software development. **ESA Bulletin**, v.107, p.112-119, 2001.

CATER-STEEL, A. P. Process improvement in four small software companies. In: AUSTRALIAN SOFTWARE ENGINEERING CONFERENCE, 2001, Queensland, Australy. **Proceedings...** IEEE, 2001. p. 262-272.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. **CMMI**: guidelines for process integration and product improvement. Boston, USA: Addison-Wesley, 2011.

CIGNONI, G. A. Rapid software process assessment to promote innovation in SMEs. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, 1999, Milan, Italy. **Proceedings...** 1999.

CLARKE, P.; O'CONNOR, R. V.; LEAVY, B. A complexity theory viewpoint on the software development process and situational context. In: IEEE/ACM INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEM PROCESSES, 2016. **Proceedings...** New York: ACM, 2016. p. 86-90.

COSTACHE, D.; KALUS, G.; KUHRMANN, M. Design and validation of feature-based process model tailoring: a sample implementation of PDE. In: ACM SIGSOFT SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERINGS, 19., 2011. **Proceedings...** Szeged, Hungary: ACM Press, 2011. p. 464-467.

CRISCUOLO, C.; GAL, P. N.; MENON, C. The dynamics of employment growth: new evidence from 18 countries. **OECD Science, Technology and Industry Policy Papers**, v. 14, 2014. DOI: 10.1787/23074957.

CRISÓSTOMO, J. et al. Convergence analysis of ISO/IEC 12207 and CMMI-DEV: a systematic literature review. In: LATIN AMERICAN COMPUTING CONFERENCE, 42., 2016, Valparaiso, Chile. **Proceedings...** IEEE, 2016. p. 1-8.

CROSBY, P. B. **Quality is free**: the art of making quality certain. New York: McGraw-Hill, 1979.

DEMING, W. E. **Out of the crisis**. Cambridge, MA: MIT Center for Advanced Engineering, 1986.

EITO-BRUN, R. Comparing SPiCE for Space (S4S) and CMMI-DEV: identifying sources of risk from improvement models. In: INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS IMPROVEMENT AND CAPABILITY DETERMINATION, 2013. Bremen, Germany. **Proceedings...** Springer, 2013. p. 84-94.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-Q-ST-30-02C**: failure modes, effects (and criticality) analysis (FMEA/FMECA). Noordwijk, The Netherlands: ECSS, 2009b.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-E-ST-40C**: software. Noordwijk, The Netherlands: ECSS, 2009a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-Q-HB-80-02-Part1A**: software process assessment and improvement – part 1: framework. Noordwijk, The Netherlands: ECSS, 2010a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION ECSS. **ECSS-S-ST-00-01C**: glossary of terms. Noordwijk, The Netherlands: ECSS, 2012.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-Q-ST-80C-Rev.1**: software product assurance. Noordwijk, The Netherlands: ECSS, 2017a.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-Q-ST-30C-Rev.1**: **dependability**. Noordwijk, The Netherlands: ECSS, 2017b.

EUROPEAN SPACE AGENCY - ESA. ESA software product assurance and engineering workshop 2015. **ESA Conference Bureau**, 2015. Available from: <<http://old.esaconferencebureau.com/2015-events/15m02/introduction>>.

EUROPEAN SPACE AGENCY - ESA. **VSEs focus group**: critical profile. Paris: ESA, 2018.

FEDERAL AVIATION ADMINISTRATION - FAA. **Order 8110.49 Chg 1**. Washington: FAA, 2011.

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION - FIPS. **FIPS 183**: Integration Definition for Function Modeling (IDEFO). Gaithersburg: FIPS, 1993.

FELDT, R. et al. Challenges with software verification and validation activities in the space industry. In: INTERNATIONAL CONFERENCE ON SOFTWARE TESTING, VERIFICATION AND VALIDATION, 3., 2010, Paris, France. **Proceedings...** IEEE Computer Society, 2010. p. 225–234.

FERNÁNDEZ, D. M. et al. Field study on requirements engineering: investigation of artefacts, project parameters, and execution strategies. **Information and Software Technology**, v. 54, n. 2, p. 162-178, Feb. 2012. DOI: 10.1016/j.infsof.2011.09.001.

GINSBERG, M. P.; QUINN, L. **Process tailoring and the software capability maturity model**. Pittsburgh: Carnegie Mellon University, 1995. (CMU/SEI-94-TR-024).

GOLDENSON, D. R.; GIBSON, D. L. **Demonstrating the impact and benefits of CMMI: an update and preliminary results**. Pittsburgh: Carnegie Mellon Software Engineering Institute, 2003.

GORSCHKE, T.; WOHLIN, C. Requirements Abstraction Model. **Requirements Engineering Journal**, v.11, p.79-101, 2006.

HAWKINS, R.; HABLI, I.; KELLY, T. **The principles of software safety assurance**. Boston: International System Safety Conference (ISSC), 2013.

HEEKS, R. et al. Synching or sinking: global software outsourcing relationships. **IEEE Software**, v. 18, n. 2, p. 54-60, Mar./Apr. 2001. DOI: 10.1109/52.914744.

HERBSLEB, J. D.; MOCKUS, A. An empirical study of speed and communication in globally distributed software development. **IEEE Transactions on Software Engineering**, v. 29, n. 6, p. 481-494, June 2003. DOI: 10.1109/TSE.2003.1205177.

HETZEL, B. **The complete guide to software testing**. 2.ed. USA: QED Information Sciences, 1984.

HIGGINS, J. P. T.; GREEN, S. **Cochrane handbook for systematic reviews of interventions version 5.1.0**. The Cochrane Collaboration, 2011. Available from: [www.handbook.cochrane.org](http://www.handbook.cochrane.org).

HIRAMA, K. **Engenharia de software: qualidade e produtividade com tecnologia**. Rio de Janeiro: Elsevier, 2011.

HOYLE, D. **ISO 9000: 2000: an A–Z guide**. New York: Butterworth-Heinemann, 2001.



HUMPHREY, W. S. **Managing the software process**. Reading, MA: Addison-Wesley, 1989.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE. **IEEE Std 610.12-1990 (R2002)**: IEEE standard glossary of software engineering terminology. New York, USA: IEEE, 2002.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO. **ISO 9000:2015**: quality management systems: fundamentals and vocabulary. Geneva, Switzerland: ISO, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **ISO/IEC 15504 information technology**: Process assessment, Software Process Improvement and Capability Determination (SPICE). Geneva, Switzerland: ISO, 2008.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **ISO/IEC 12207:2008**: systems and software engineering: software life cycle processes. Geneva, Switzerland: ISO, 2008a.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **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, 2011a.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **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, 2011b.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **ISO/IEC/IEEE 15288:2015** systems and software engineering? system life cycle processes. Geneva, Switzerland: ISO, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION;  
INTERNATIONAL ELECTROTECHNICAL COMMISSION - ISO/IEC. **ISO/IEC/IEEE 12207:2017** - systems and software engineering: software life cycle processes. Geneva, Switzerland: ISO, 2017.

JIANG, J. J. et al. The relation of requirements uncertainty and stakeholder perception gaps to project management performance. **Journal of Systems and Software**, v. 82, n. 5, p. 801-808, May 2009. ISSN 0164-1212.

JOHNSON, D. L.; BRODMAN, J. G. Tailoring the CMM for small businesses, small organizations, and small projects. **Software Process Newsletter - IEEE Computer Society**, v. 8, 1997.

JØRGENSEN, M. A review of studies on expert estimation of software development effort. **Journal of Systems and Software**, v.70, n.1/2, p. 37-60, Feb. 2004. DOI: 10.1016/S0164-1212(02)00156-5.

JØRGENSEN, M.; MOLOKKEN, K. A preliminary checklist for software cost management quality software. In: INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE, 3., 2003, Dallas, USA. **Proceedings...** IEEE, 2003. p. 134-140.

JØRGENSEN, M.; SHEPPERD, M. A systematic review of software development cost estimation studies. **IEEE Transactions on Software Engineering**, v.33, n.1, p.33-53, 2007.

JURAN, J. M. **Juran on planning for quality**. New York: Macmillan, 1988.

KALINOWSKI, M. et al. Software process improvement results in Brazil based on the MPS-SW model. **Software Quality Professional**, p.14-28, Sept. 2015.

KALUS, G.; KUHRMANN, M. Criteria for software process tailoring: a systematic review. In: INTERNATIONAL CONFERENCE ON SOFTWARE AND SYSTEM PROCESS, 2013. **Proceedings...** New York: ACM. 2013. p. 171-180.

KELLY, D. P.; CULLETON, B. Process improvement for small organizations. **Computer**, v. 32, n. 10, p. 41-47, 1999. DOI: 10.1109/2.796108.

KOMI-SIRVIÖ, S. **Development and evaluation of software process improvement methods**. Thesis (PhD) - University of Oulu, Rovaniemi, Finland. 2004.

KUHRMANN, M.; KALUS, G. Providing integrated development processes for distributed development environments. In: WORKSHOP ON SUPPORTING DISTRIBUTED TEAM WORK AT COMPUTER SUPPORTED COOPERATIVE WORK, 2008, San Diego, USA. **Proceedings...** 2008.

KUHRMANN, M.; TERNITÉ, T. Including the microsoft solution framework as an agile method into the V-Modell XT. In: INTERNATIONAL WORKSHOP ON EVALUATION OF NOVEL APPROACHES TO SOFTWARE ENGINEERING, 1., 2005, Erfurt, Germany. **Proceedings...** 2005.

KUILBOER, J. P.; ASHRAFI, N. Software process and product improvement: an empirical assessment. **Information and Software Technology**, v.42, p.27-34, 2000.

KUSHNIR, K.; MIRMULSTEIN, M. L.; RAMALHO, R. **Micro, small and medium enterprises around the world: how many are there and what affects the count?** Washington: World Bank, 2010.

KUVAJA, P.; PALO, J.; BICEGO, A. TAPISTRY: a software process improvement approach tailored for small enterprises. **Software Quality Journal**, v.8, p.149-156, 1999.

LAHOZ, C. H. N. **S4S For Very Small Entities (VSE)**. [S.l: s.n.], 2015.

LAHOZ, C. H. N.; RICHTER, S.; RICO, D. E. **Rapid software process assessment in the space domain for Very Small Entities**. Frascati: European Space Agency. 2015.

LAPORTE, C. Y. **ISO/IEC 29110: profiles**. [Personal communication]. Message received by Gledson Hernandes Diniz on April 18, 2017.

LAPORTE, C. Y.; ALEXANDRE, S.; O'CONNOR, R. A software engineering lifecycle standard for Very Small Enterprises. **Communications in Computer and Information Science**, v.16, p.129-141, 2008.

LAPORTE, C. Y.; O'CONNOR, R. V. Software process improvement standards and guides for very small organizations: an overview of eight implementations. **CrossTalk, The Journal of Defense Software Engineering**, v. 30, n. 3, p. 23-27, 2017.

LAPORTE, C. Y.; O'CONNOR, R. V.; PAUCAR, L. H. G. Software engineering standards and guides for Very Small Entities: implementation in two start-ups. In: INTERNATIONAL CONFERENCE ON EVALUATION OF NOVEL APPROACHES TO SOFTWARE ENGINEERING (ENASE), 2015, Barcelona, Spain. **Proceedings...** IEEE, 2015. p. 5-15.

LARRUCEA, X. et al. Software process improvement in very small organizations. **IEEE Software**, v.33, n.2, p.85-89, 2016.

LARYD, A.; ORCI, T. Dynamic CMM for small organizations. In: ARGENTINE SYMPOSIUM ON SOFTWARE ENGINEERING, 2000, Tandil, Argentina. **Proceedings...** Academic Press, 2000. p. 133-149.

LICHTENSTEIN, Y. Puzzles in software development contracting. **Communications of the ACM**, v. 47, n. 2, p. 61-65, Feb. 2004. DOI: 10.1145/966389.966391.

MARQUES, J. C. **MACRE-SAR: an agile model for software requirements specification in regulated environments.** Thesis (PhD) - Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil. 2016.

MIYASHIRO, M. A. S.; FERREIRA, M. G. V. One approach to the use of the practices of CMMI-DEV V1.3 level 2 in a process of development of embedded systems. In : INTERNATIONAL CONFERENCE ON INFORMATION, 5., 2014, Greece. **Proceedings...** 2014.

MOLL, R. **A bird's eye view of SMEs and risk management.** Geneva, Switzerland: ISO, 2013.

MOTODA, H. . M. R.; BOOSE, J.; GAINES, B. Knowledge acquisition for knowledge-based systems. **IEEE Expert**, v. 6, n. 4, p. 53-64, Aug. 1991. DOI: 10.1109/64.85921.

MUNCH, J. et al. **Software process definition and management.** Berlin: Springer-Verlag, 2012.

MYERS, M. D. Qualitative Research in Information Systems. **MISQ Discovery**, v.21, n.2, p.241-242, May 1997.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION - NASA. **NASA study on flight software complexity.** Pasadena, CA, USA: NASA, 2009.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION - NASA. **NASA systems engineering handbook.** Washington, USA: NASA, 2017.

NAUR, P.; RANDELL, B. **Software engineering: a report on a conference sponsored by the NATO Science Comitee.** Brussels: [S.n.], 1969.

NIAZI, M.; WILSON, D.; ZOWGHI, D. A model for the implementation of a software process improvement: a pilot study. In: INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE, 3., 2003, Dallas, USA. **Proceedings...** IEEE, 2003.

NIAZI, M.; WILSON, D.; ZOWGHI, D. A maturity model for the implementation of software process improvement: an empirical study. **Journal of Systems and Software**, v.74, n.2, p.155-172, 2005.

NIAZI, M.; WILSON, D.; ZOWGHI, D. Critical success factors for software process improvement implementation: an empirical study. **Software Process: Improvement and Practice**, v.11, p.193-211, 2006.

O'CONNOR, R. V.; LAPORTE, C. Y. Towards the provision of assistance for Very Small Entities in deploying software lifecycle standards. In: INTERNATIONAL CONFERENCE ON PRODUCT FOCUSED SOFTWARE DEVELOPMENT AND PROCESS IMPROVEMENT, 11., 2010. **Proceedings...** 2010.

O'CONNOR, R.; BASRI, S.; COLEMAN, G. Exploring managerial commitment towards SPI in small and Very Small Enterprises. In: EUROPEAN CONFERENCE, 17., Grenoble, France. **Proceedings...** Springer-Verlag. 2010. p. 268-278.

O'CONNOR, R.; COLEMAN, G. Ignoring 'best practice': why irish software SMEs are rejecting CMMI and ISO 9000. **Australasian Journal of Information Systems**, v. 16, n. 1, June 2009.

O'CONNOR, R.; LAPORTE, C. Y. **Deploying lifecycle profiles for Very Small Entities**: an early stage industry view. Ireland: Springer-Verlag, 2011a. p. 227-230.

O'CONNOR, R.; LAPORTE, C. Y. Using ISO/IEC 29110 to harness process improvement in Very Small Entities. In: EUROPEAN SOFTWARE PROCESS IMPROVEMENT CONFERENCE, 18., 2011, Roskilde, Denmark. **Proceedings...** Springer-Verlag, 2011b. p. 225-235.

OFFEN, R. J.; JEFFERY, R. Establishing software measurement programs. **IEEE Software**, v. 14, n. 2, p. 45-53, Mar.-Apr. 1997. DOI: 10.1109/52.582974.

PAASIVAARA, M.; LASSENIUS, C. Collaboration practices in global inter-organizational software development projects. **Software Process Improvement and Practice**, v. 8, n. 4, p. 183-199, 2004. DOI: 10.1002/spip.187.

PAI, M. et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. **National Medical Journal of India**, v.17, n.2, p.86-95, 2004.

PEDREIRA, O. et al. A systematic review of software process tailoring. **ACM SIGSOFT Software Engineering Notes**, v. 32, n. 3, p. 1-6, May 2007. DOI: 10.1145/1241572.1241584.

PETTERSSON, F. et al. A practitioner's guide to light weight software process assessment and improvement planning. **Journal of Systems and Software**, v.8, n.6, p.972-995, 2008.

PINO, F. J. et al. Assessment methodology for software process improvement in small organizations. **Information and Software Technology**, v. 52, n. 10, p. 1044-1061, Oct. 2010. DOI: 10.1016/j.infsof.2010.04.004.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: McGraw Hill, 2007.

RICHARDSON, I. Quality function deployment: a software process tool? In: ANNUAL INTERNATIONAL QFD SYMPOSIUM, 3., 1997, Linköping, Sweden. **Proceedings...** Linköping University, 1997. p. 39-49.

RODRÍGUEZ-DAPENA, P.; LOHIER, P. How small organizations could participate in Space projects. In: IEEE INTERNATIONAL WORKSHOP ON METROLOGY FOR AEROSPACE, 2017, Padua, Italy. **Proceedings...** IEEE, 2017. p. 11-15.

ROUT, T. P. et al. The rapid assessment of software process capability. In: INTERNATIONAL SPICE CONFERENCE, 1., 2000 Limerick, Ireland. **Proceedings...** 2000.

RTCA. **RTCA/DO-178C** - software considerations in airborne systems and equipment certification. Washington: Federal Aviation Administration, 2011.

SALAS, E. et al. The effect of team building on performance: an integration. **Small Group Research**, v. 30, n. 3, p. 309-329, June 1999. DOI: 10.1177/104649649903000303.

SALVIANO, C. F.; FIGUEIREDO, A. M. C. M. Unified basic concepts for process capability models. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 20., 2008. **Proceedings...** 2008. 173-178.

SANCHEZ-GORDON, M.-L. et al. A standard-based framework to integrate software work in small settings. **Computer Standards & Interfaces**, v. 54, n. 3, p. 162-175, 2017. ISSN 0920-5489.

SAUNDERS, M. N. K.; LEWIS, P.; THORNHILL, A. **Research methods for business students**. 5.ed. [S.l.]: Pearson, 2009.

SCHOEFFEL, P.; BENITTI, F. B. 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**, p.1634-1643, 2015.

SHEWHART, W. A. **Economic control of quality of manufactured product**. [S.l.]: American Society for Quality Control, 1931.

SOFTWARE ENGINEERING INSTITUTE - SEI. **CMMI-DEV, V1.3 - CMMI for development, version 1.3**. Pittsburgh, USA: SEI, 2010.

- TORCHIANO, M.; MORISIO, M. Overlooked aspects of COTS-based development. **IEEE Software**, v. 21, n. 2, p. 88-93, Mar. 2004. DOI: 10.1109/MS.2004.1270770.
- USMAN, M. et al. Developing and using checklists to improve software effort estimation: a multi-case study. **Journal of Systems and Software**, v. 146, p. 286-309, 2018.
- VÉRAS, P. C. et al. A benchmarking process to assess software requirements documentation for space applications. **Journal of Systems and Software**, v. 100, p. 103-116, Feb. 2015. DOI: 10.1016/j.jss.2014.10.054
- VILLALÓN, J. A. et al. Experiences in the application of software process improvement in SMES. **Software Quality Journal**, v. 10, n.3, Oct. 2002. 261-273. DOI: 10.1023/A:1021638523413
- WALLACE, L.; KEIL, M. Software project risks and their effects and outcomes. **Communications of the ACM**, v. 47, n. 4, p. 68-73, Apr. 2004. DOI: 10.1145/975817.975819.
- WALLACE, L.; KEIL, M.; RAI, A. How software project risk affects project performance: an investigation of the dimensions of risk and an exploratory model. **Decision Sciences**, v. 35, n. 2, p. 289-321, May 2004. DOI: 10.1111/j.00117315.2004.02059.x.
- WANG, Z.-J.; ZHAN, D.-C.; XU, X.-F. STCIM: a dynamic granularity oriented and stability based component identification method. **ACM SIGSOFT Software Engineering Notes**, v. 31, n. 3, p. 1-14, May 2006. ISSN 0163-5948.
- WANGENHEIM, C. G. V. et al. Systematic literature review of software process capability/maturity models. In: INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS IMPROVEMENT AND CAPABILITY DETERMINATION, 2010, Pisa, Italy. **Proceedings...** Springer. 2010.
- WOLFE, J.; CHACKO, T. I. Team-size effects on business game performance and decision-making behaviors. **Decision Sciences**, v. 14, n. 1, p. 121-133, Jan. 1983. DOI: 10.1111/j.1540-5915.1983.tb00173.x.
- WORLD TRADE ORGANIZATION - WTO. **World trade report 2016: levelling the trading field for SMEs**. Geneva, Switzerland: WTO, 2016.
- XIA, W.; LEE, G. Grasping the complexity of IS development projects. **Communications of the AC**, v. 47, n. 5, p. 68-74, May 2004. DOI: 10.1145/986213.986215.

XU, P.; RAMESH, B. Using process tailoring to manage software development challenges. **IT Professional**, v. 10, n. 4, p. 39-45, 2008. DOI: 10.1109/MITP.2008.81.

YILMAZ, M.; O'CONNOR, R. V.; CLARKE, P. Effective social productivity measurements during software development: an empirical study. **International Journal of Software Engineering and Knowledge Engineering**, v.26, n.3, p.457-490, 2016.

YONG, I. C.; MIN, S. Y.; BAE, D. H. Tailoring and verifying software process. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 8., 2001, Macao, China. **Proceedings...** IEEE, 2001. p. 202- 209.

YOUSEFAL-TARAWNEH, M.; ABDULLAH, M. S.; ALI, A. B. M. A proposed methodology for establishing software process development improvement for small software development firms. **Procedia Computer Science**, v.3, p.893-897, 2011.

ZAROOUR, M. et al. 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**, v.101, p.180-192, Nov. 2015.

ZOWGHI, D.; NURMULIANI, N. A study of the impact of requirements volatility on software project performance. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 9., 2002, Gold Coast, Australia. **Proceedings...** 2002. p. 3-11.