FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

EQualPI: a Framework to Evaluate the Quality of the Implementation of the CMMI Practices

Isabel de Jesus Lopes Margarido

JURI VERSION 1.0

Programa Doutoral em Engenharia Informática

Advisor: Raul Moreira Vidal, FEUP

Co-advisor: Marco Paulo Amorim Vieira, FCTUC

Porto, January 21, 2016

EQualPI: a Framework to Evaluate the Quality of the Implementation of the CMMI Practices

Isabel de Jesus Lopes Margarido

Programa Doutoral em Engenharia Informática

Porto, January 21, 2016

Abstract

The Capability Maturity Model Integration® (CMMI) allows organizations to improve the quality of their products and customer satisfaction; reduce cost, schedule, and rework; and make their processes more predictable. However, this is not always the case, as there are differences in performance among CMMI organizations, depending not only on the context of the business, projects, and team, but also on the methodologies used in the implementation of the model practices. CMMI version 1.3 is more focused on the performance of the organisations than previous versions. However, the Standard CMMI Appraisal Method for Process Improvement^{SM} (SCAMPI) is not focused on evaluating performance.

To evaluate practices performance it is necessary to consider the goal of executing the practice, and the quality of implementation of a practice is reflected on its outputs. Therefore, if we can establish a relationship between the methods used to implement a practice and the performance results of that practice, we can use such relationship in a framework to evaluate the quality of implementation of the practice. We consider that it is possible to objectively measure the quality of implementation of the CMMI practices by applying statistical methods in the analysis of organisations' data, in order to evaluate process improvement initiatives and predict their impact on organisational performance.

In this research we develop a framework to evaluate the quality of the implementation of the CMMI practices that supports the comparison of the quality of the implementation before and after improvements are put in place. Considering the extent of the CMMI model we demonstrate the framework in the Project Planning's Specific Practice 1.4 "Estimate Effort and Cost". We consider that the quality of implementation of this practice is measured by the Effort Estimation Accuracy, defined by a set of controllable and non-controllable factors, and it can be improved by acting on the controllable factors. To implement and validate our framework we conducted literature reviews, case studies on high maturity organisations, data analysis of a survey performed by the Software Engineering Institute (SEI) and on the Team Software Process^{SM} (TSP) Database, which we used to build a regression model, and conducted an experiment with students to define a process improvement.

This Ph.D. thesis provides to software development organisations a framework for self-assessing the quality of the implementation of the CMMI practices, EQualPI. The framework is also useful to the CMMI Institute, to evaluate the performance of the organisations from one SCAMPI A to the next. The framework is already populated with a performance indicator model to evaluate the quality of implementation of the effort estimation process, recommendations to support organisations willing to implement CMMI to avoid a set of problems and difficulties, factors to consider when implementing measurement and analysis for CMMI high maturity levels. Additionally, with the validation of the EQualPI framework, we provide the procedure we used to analyse data from the SEI TSP Database and define process variables and a defects classification specific for requirements.

Resumo

O Capability Maturity Model Integration® (CMMI) permite às organizações melhorar a qualidade dos seus produtos e satisfação dos seus clientes; reduzir custos, calendário e a necessidade de refazer trabalho. Com o CMMI os processos passam a ser mais previsíveis. No entanto nem sempre é este o caso, dado que há uma diferença de desempenho entre organizações que usam CMMI, que depende não somente do negócio, projectos e equipas mas também das metodologias usadas na implementação das práticas modelo. A versão 1.3 do CMMI é mais focada na performance das organizações do que as anteriores, no entanto o seu método de avaliação, Standard Appraisal Method for Process Improvement ^{SM} (SCAMPI), não tem como objectivo avaliar o desempenho das organizações.

Para avaliar o desempenho de práticas é necessário considerar o objectivo de as executar e que a qualidade de implementação de uma prática se reflecte nos seus resultados. Por esse motivo, podemos estabelecer uma relação entre os métodos utilizados na implementação de uma prática e os resultados da sua performance. Consideramos que é possível medir objectivamente a qualidade de implementação das práticas do CMMI aplicando métodos estatísticos na análise dos dados de organizações, para dessa forma avaliar as iniciativas de melhoria de processos e prever o impacto que essas melhorias vão ter na performance da organização.

Nesta investigação científica deselvovemos uma *framework* para avaliar a qualidade de implementação das práticas CMMI que permite comparar a qualidade da implementação antes de depois de introduzir uma melhoria. No entanto, o CMMI é extenso, por esse motivo vamos demonstrar a *framework* na área específica do processo de Planeamento de Projectos 1.4 "Estimar esforço e custo". Consideramos que a qualidade de implementação desta prática é medida através da precisão da estimativa de esforço, definida por um conjunto de factores controláveis e não controláveis, e que o seu valor pode ser melhorado actuando sobre os factores controláveis. Para implementar e validar a nossa framework efectuámos revisões de literatura, casos de estudo em organizações de alta maturidad, análises de dados sobre um inquérito realizado pelo *Software Engineering Institute* (SEI) e sobre a base de dados do Team Software Process^{SM} (TSP), que utilizámos para implementar um modelo de regressão linear, e conduzimos uma experiência com estudantes para definir uma melhoria de processo.

Como resultado desta tese de Doutoramento disponibilizamos às organizações a EQualPI, uma *framework* de auto-avaliação da qualidade de implementação das práticas CMMI. Esta *framework* também é útil para o CMMI Institute poder avaliar o desempenho das organizações aquando da recertificação. A EQualPI tem já incluído um modelo de um indicador de performance para avaliar a qualidade de implementação da prática de estimação de esforço, um conjunto de recomendações de suporte às organizações que pretendem implementar o CMMI evitando um conjunto de problemas e dificuldades, bem como recomendações sobre factores a considerar quando se implementa a prática de *Measurement Analysis* em níveis de alta maturidade. Adicionalmente, da validação da EQualPI, resultou o procedimento que seguimos na análise dos dados TSP que se encontram na base de dados do SEI, e na definição das variáveis de processo. Resultou também uma lista de

classificação de defeitos específica para documentos de requisitos.

Acknowledgments

This adventure would have not been possible without the love and support of my better half, who took care of me in all the hard moments; my parents and my sisters who are always there for me and understood my long absences to do this research; my native reviewer and Mena.

I thank the SEI for receiving me so well, in particular Paul Nielsen, Anita Carlton, Rusty Young, Eileen Forrester, Gene Miluk, Jim Over, Mike Conrad, Bob Stoddard and Jim McCurley. Special thanks to Dave Zubrow and Bill Nichols for the great work we did together, and Dennis Goldenson for his advice and support.

I thank my SEPG friends Mike Campo, Kess Hermus and Mia for the great moments we spent together and CMMI talks.

My PhD colleagues, Professors and CISUC colleagues for all discussions and good times.

A big thank you to my MBFs, for all the hangouts and my friends, close and absent, for supporting me.

I also have to thank my supervisors, co-authors and reviewers, who contributed to this research.

Finally, I cannot finish without thanking to the person without whom I could not have done this research, Watts Humphrey.

Isabel de Jesus Lopes Margarido

Contents

List of Figures

List of Tables

Acronyms and Definitions

Acronyms

AIM	Accelerated Improvement Method
BSC	Balanced Score Card
BU	Business Unit
CAR	Causal Analysis and Resolution (process area)
CI	Configuration Item
CI, II	COCOMO I, COCOMO II
CISQ	Consortium of IT Software Quality
CISUC	Centre for Informatics and Systems of the University of Coimbra
CL	Capability Level
CM	Configuration Management (process area)
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMMI-ACQ	CMMI for Acquisition
CMMI-DEV	CMMI for Development
CMMI-SVC	CMMI for Services
CMU	Carnegie Mellon University
COCOMO	Constructive Cost Model
DAR	Decision Analysis and Resolution (process area)
DEI	Department of Informatics Engineering
DL	Deliverable
DoD	Department of Defence (funding the SEI)
DOE	Design of Experiments
EEA	Effort Estimation Accuracy
f	Function
FCT/UNL	Faculty of Science and Technology, Universidade NOVA de Lisboa
FCTUC	Faculty of Sciences and Technology, University of Coimbra
FEUP	Faculty of Engineering, University of Porto
FI	Fully Implemented
FL	Fuzzy Logic
FPA	Function Point Analysis
FSS	Feature Subset Selection
GA	Genetic Algorithm
GG	Generic Goal
GH	Generic Hypothesis
GP	Genetic Programming
GQiM	Goal Question (Indicator) Metric
GQM	Goal Question Metric

H0	Null Hypothesis
H1	Alternative Hypothesis
HML	High Maturity Level
HP	Hewlett-Packard
ICSE	International Conference on Software Engineering
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IPM	Integrated Project Management (process area)
ISAM	Integrated Software Acquisition Metrics
ISO	International Organisation for Standardisation
IT	Information Technology
ITIL	Information Technology Infrastructure Library
KLOC	Thousand Lines of Code
KPI	Key Performance Indicators
LI	Largely Implemented
LSR	Least Squares Regression
M2DM	Metamodel Driven Measurement
MA	Measurement and Analysis (process area)
MARE	Mean Absolute Relative Error
MER	Magnitude Error Relative
MIEIC	Integrated Master in Informatics Engineering and Computation
MinBU	Minimum number of Business Units
MK II FPA	Mark II Function Point Analysis
ML	Maturity Level
MLP	Multy-layer Perceptron
MMR	Multidimensional Measurement Repository
MMRE	Mean Magnitude Relative Error
MRE	Magnitude Relative Error
N/A	Not Applicable
NI	Not Implemented
NY	Not Yet
ODM	Ontology Driven Measurement
OID	Organisational Innovation and Deployment (process area)
OMG	Object Management Group
OPM	Organisational Performance Management (process area, CMMI V1.3)
OPP	Organisational Process Performance (process area)
OT	Organisational Training (process area)
PA	Process Area
PB	Publication
PBC	Performance Benchmarking Consortium
PI	Performance Indicator
PIm	Partially Implemented
PMC	Project Monitoring and Control (process area)
PMI	Project Management Institute
PMP	Project Management Professional
PP	Project Planning (CMMI process area)
PPB	Process Performance Baseline
PPM	Process Performance Model

PPQA	Process and Product Quality Assurance (process area)
PRED	Percentage of Predictors
Price-S	Parametric Review Information for Costing and Evaluation – Software
PROBE	PROxy-Based Estimation Method
ProDEI	Doctoral Program in Informatics Engineering
PSM	Practical Software Measurement
PSO	Particle Swam Optimization
PSP	Personal Software Process
QPM	Quantitative Project Management (process area)
QUASAR	Quantitative Approaches on Software Engineering and Reengineering
RBF	Radial Basis Function
RD	Requirements Development (process area)
REQM	Requirements Management (process area)
RSK	Risk
RSKM	Risk Management (process area)
SAM	Supplier Agreement Management (process area)
SBO	Software Benchmarking Organisation
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SD	Standard Deviation
SEER-SEM	Software Evaluation and Estimation Resources – Software Estimation Model
SEI	Software Engineering Institute
SEMA	Software Engineering Measurement and Analysis
SG	Specific Goal
SLIM	Software Lifecycle Management
SME	Small Medium Enterprise
SoftEng	Software Engineering Laboratory (FEUP)
SP	Specific Practice
SPI	Software Process Improvement
SPR	Software Productivity Research
SVM	Support Vector Regression
TIES	Software Engineering Research Topics
TRW	Tandem Random Walk
TS	Technical Solution (process area)
TSP	Team Software Process
UCP	Use Case Points
UML	Unified Modelling Language
USA	United States of America
V	Version
VAR	Variance Account For
VARE	Variance Absolute Relative Error
WP	Work Product

Definitions

Affirmations	"Oral or written statement confirming or supporting implementation (or lack of implementation) of a model practice provided by the implementers of the practice, provided via an interactive forum in which the appraisal team has control over the interaction (CMU/SEI, 2011)."
Artefacts	"Tangible forms of objective evidence indicative of work being performed that represents either the primary output of a model practice or a consequence of implementing a model practice (CMU/SEI, 2011)."
Benchmark	To take a measurement against a reference point. Benchmarking is a process of comparing and measuring an organisation with the business leaders located anywhere (Kasunic, 2006). The acquired information helps the organisation to improve its performance.
Constellation	VBVBBB
Self-directed Teams	Teams whose members sense the project needs without being told, help whenever is necessary and "do whatever is needed to get the job done" (Humphrey, 2006).
Data Sufficiency Rules	Coverage rules that determine how much evidence (Affirmations and Artefacts) needs to be provided in the SCAMPI A (Byrnes, 2011).
Fully Implemented	Sufficient artefacts and or/affirmations are present and judged to be adequate to demonstrate practice implementation (CMU/SEI, 2011). No weaknesses are noted.
Largely Implemented	Sufficient artefacts and or/affirmations are present and judged to be adequate to demonstrate practice implementation (CMU/SEI, 2011). One or more weaknesses are noted.
Not Implemented	Some or all data required are absent or judged to be inadequate (CMU/SEI, 2011). Data supplied does not support the conclusion that the practice is implemented. One or more weaknesses are noted.
Not Yet	"The basic unit or support function has not yet reached the stage in the sequence of work, or point in time to have implemented the practice. (CMU/SEI, 2011)"
Organisational Scope	A subset of the organisational unit that is determined by selecting support functions and basic units to supply data for the SCAMPI appraisal (CMU/SEI, 2011).
Organisational Unit	The part of the organisation that is subject of a SCAMPI appraisal and to which results will be generalised (CMU/SEI, 2011).
Partially Implemented	Some or all data required are absent or judged to be inadequate (CMU/SEI, 2011). Some data are present to suggest some aspects of the practice are implemented. One or more weaknesses are noticed. OR Data supplied to the team conflict. One or more weaknesses are noted.
Sampling Factors	Rule to select the organisation Basic Units into subgroups that determine the organisational scope to be target of the SCAMPI A (Byrnes, 2011). Are used to ensure adequate representation of the organisational unit.
Standard Processes	Processes that the organisation statistically controls to assure that the organisation and projects quantitative objectives are achieved.
Subgroups	Subset of the organisational unit defined by sampling factors, that are basic units with common attributes (CMU/SEI, 2011).

Chapter 1

Introduction

When we open the CMMI(Capability Maturity Model Integration) for development (Chrissis et al., 2011) book and read the preface the models are presented as "collections of best practices that help organisations improve their processes" and the CMMI for development (DEV) "provides a comprehensive integrated set of guidelines to develop products and services". For years several successful stories have been presented to the world, showing the benefits organisations achieved when using the model, going from improving the quality of the products and processes, to reducing schedule, costs and amount of rework. Consequently, processes become more predictable and customer satisfaction increases (Goldenson et al., 2004). The model is an improvement tool that can be implemented step by step, used to improve a process area, evaluate capability or maturity, or simply improve selected practices. CMMI also guides organisations in building measurement capability to provide the information necessary to support management needs, as stated in the Measurement and Analysis process area (Chrissis et al., 2011). For adequate use, it is necessary to understand the model as a whole. In the staged representation the CMMI model is composed by 5 maturity levels, each of them achieved with the implementation of the specific and generic goals prescribed in the model in a current maturity level and all the precedent ones. To satisfy a goal the generic and specific practices, or acceptable alternatives to them, need to be fulfilled. The levels 4 and 5 are called high maturity levels. In these levels the organisations need to have knowledge on simulation, modelling and statistical analysis that support building process performance models that are relevant to indicate the status of objective and measurable organisation goals, and have process performance baselines to quantitatively control process/product execution. In maturity level 5 the organisations use their knowledge and capability of anticipating the behaviour of their standard sub-processes to support decisions regarding performance improvements or resolution of problems. This implies that decisions are made based on evidence of the success of the solutions.

There is plenty information, tips and cases of what makes CMMI work to help organisations improve, but it is still their choice how they shape their processes to respond to their business needs and reflect their culture. Besides, a process may be defined but the real process is the one actually being executed. Even on strict sets of rules the real process may still differ from the documented desired process. As each organisation has a choice on how to implement the model, use the

practices and perform their work, there is high variability when comparing performance results. Therefore, there are unsuccessful cases and organisations achieving a maturity model but not performing accordingly. The Software Engineering Institute (SEI) and the United States Department of Defence (DoD) (Schaeffer, 2004) expressed concern with high maturity implementation as not all organisations understood it well, which reflected on their performance and the release of V1.3 was intended to fix this problem (Campo, 2012).

CMMI Version 1.3 emphasises improvements on the organisations' performance, i.e. it clarifies that organisations need to focus processes on their business goals and do performance improvements to achieve those goals that are continuously improving. The Standard Appraisal Method for Process Improvement^{SM} (SCAMPI^{SM}) appraises the alignment of the organisation's processes, activities and results with the CMMI model but its objective is not to measure performance. And to the best of our knowledge there is no tool to measure organisations' performance and evaluate it as a result of the quality of implementation of the CMMI practices and/or goals.

1.1 Conducted Research

This research has the following main objectives:

Objective 1 – Identify problems and difficulties in the implementation of CMMI to help to define the problem to tackle.

Objective 2 – Develop and validate a framework to evaluate the quality of implementation of the CMMI practices.

Objective 3 – Demonstrate the evaluation of quality of implementation by building the performance indicator model to evaluate the particular case of the Project Planning process's Specific Practice 1.4 "Estimate Effort and Cost".

1.1.1 Problem Definition

The problem is composed by three main aspects. One is that CMMI has a **high variability of performance**. Schreb (2010) compiled the problems of CMMI: implementations that do not impact projects, wide range of solutions for the same practice not all leading to high performance, and highly variable approaches to implementation that may not lead to performance improvement. The model is not prescriptive, it only provides guidance and therefore the performance of the organisations implementing it depends on factors related to the business and teams, but also on the methods used to perform the work and quality of implementation of the model. In fact, as Peterson stated, the big issue is CMMI implementation (Schreb, 2010). Furthermore, we state, that the quality of the implementation of the practices has an impact in performance indicators, related to the organisations' objectives.

Another aspect of the problem is the **quality of implementation** of the model. Some organisations have difficulties in the selection of the implementation methods others simply copy the

model as if it was a standard which leads to bad implementations. We consider that if the quality of implementation is good, the performance of the organisation using CMMI is improved.

Lastly, we consider that there is a need for a **performance evaluation method** that can help organisations to assess the quality of implementation of the practices and if they are actually improving their results or not. Even though the version 1.3 of the CMMI model is more focused in the organisation performance, the objective of the SCAMPI is not to measure performance but appraise organisations' compliance with the model. Regarding this problem we consider that it should be possible to define metrics that measure the quality of the implementation of the CMMI model and measure the effects of process improvements. Having this capability would help to avoid implementation problems by early recognition of failures.

We synthesise the problem that we tackled in the following statement:

Not all organisations using CMMI achieve the best performance results, which could be achieved using a good implementation of the model. If a relationship can be established between methods used to implement a practice and the performance results of that practice, such relationship can be used in a framework to evaluate the quality of implementation of that practice.

1.1.2 Research Questions and Hypothesis

Our research questions are the following:

- Why some organisations do not achieve the expected benefits when implementing CMMI?
- Why SCAMPI does not detect implementation problems, or does not address performance evaluation in all maturity levels?
- What additional recommendations can we provide to organisations to help them avoid problems when implementing CMMI?
- How can we evaluate the quality of implementation of the CMMI practices ensuring that organisations fully get their benefits and perform as expected?
- Is it possible to define metrics to evaluate the quality of implementation of CMMI practices focused on their effectiveness?
- Can we determine the effects, expressed in a percentage, of non-controllable factors in an evaluation metric?

Based on the problem statement and the theory that there is a relationship between the quality of implementation of a CMMI practice and the quality of the outcome of the application of that practice, we theorise that *it is possible to objectively measure the quality of implementation of the CMMI practices by applying statistical methods, in the analysis of organisations' data, in order to evaluate process improvement initiatives and predict their impact on organisational performance*.

To demonstrate our theory we endeavoured on a quest to model a quality indicator to measure the quality of implementation of the CMMI Project Planning Specific Practice 1.4 "Estimate Effort and Cost".

1.1.3 Beneficiaries

With the development and demonstration of our framework we will provide organisations a tool that can help them to:

- Implement CMMI, by providing a pool of methods, that can be adapted to implement the practices, and performance indicators to monitor them;
- Choose methods not only for their adequacy to the context but for their performance, in terms of efficacy and efficiency, when compared to others;
- Monitor process performance in order to act before problems occur;
- Anticipate impact of process changes on the performance indicators;
- Prioritise performance improvements;
- More accurately understand the origin of certain results.

The CMMI Institute will be able to assess that there were actual performance improvements in a given organisation from one appraisal to the next.

1.2 Thesis Organisation

This dissertation is organised in five chapters. This first one, 1 Introduction presents the the motivation and problem that drove the research work done on this PhD thesis; states the objectives of the research, the research questions and hypothesis and indicates the beneficiaries of our research work.

In chapter 2 Fundamental Concepts, we present the concepts necessary to understand this research and the remainder chapters of the thesis.

Chapter 3 State of the Art gives the necessary information to delimit the problem and the contributes of other researchers to help solve some of the problem components.

We present our contribution to solve open points identified on prior research, on chapter 4 The EQualPI Framework, the core of our research, where we detail the framework to evaluate the quality of implementation of the CMMI practices and how organisations can use it.

On chapter 5 EQualPI Validation, we validate the framework we detailed on the previous chapter.

In chapter 6 Conclusions we guide the reader in how the framework is extended to other practices, indicate our achievements and their impact in the problem resolution and define the boundaries of this research. We leave the research open and point directions for future work that needs to be done in this area.

Chapter 2

Fundamental Concepts

Software Engineering is a discipline that appeared from the necessity of producing software applying engineering principles van Vliet (2007). IEEE defines it as "the application of systematic, discipline, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software" (IEEE, 1990). This concept is aligned with Humphrey (1988), who defined a **Software Engineering Process** as being the "total set of software engineering activities needed to transform user requirements into software". The process may include requirements specification, design, implementation, verification, installation, operational support, and documentation. Fuggetta (2000) extends the definition by stating that a software process is defined as a coherent set of policies, organisational structures, technologies and artefacts necessary to develop, deploy and maintain a software product.

If software engineering is a "quantifiable approach" it needs to be measurable, hence apply measurement. In figure 2.1 we represent the measurement components and the relations between them. We clarify these concepts in the next section.

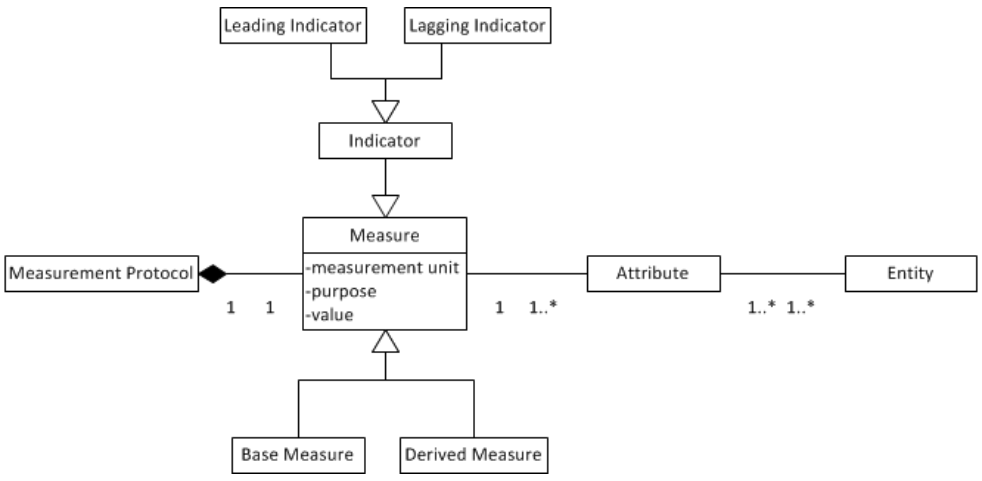


Figure 2.1: Measurement components.

2.1 Measurement

There are several definitions for the adequate terms to use in software engineering with respect to expressions such as measures, metrics, metrication, etc. (Zuse, 1997). Ragland (1995) clarifies the definitions of the terms measure, metric and indicator, based on the definitions of the Institute of Electrical and Electronics Engineers, Inc. (IEEE) and the Software Engineering Institute (SEI), and provides illustrative examples.

To **measure** (verb) is to ascertain or appraise to a standard that may be universal or local. It is considered an act or process of measuring; the result of measurement. The **measure** (noun) is the result of the act of measuring. An example of a measure is a single data point, for instance "today I produced 10 pages of my thesis". The data point to register would be *10*, i.e. the **value**. However, that information would be insufficient to analyse the measure. It is necessary to define the **measurement unit**, that in this case is *number of pages*, and the **purpose** of the measure, in this case *to know how long it took me to produce my thesis*. In ISO (2001), a **metric** is defined as "a measurement scale and the method used for measurement". Such term is often used to designate the data point and all information that allows collecting and analysing it, and that is the definition we are following when we refer to it in this thesis. So, the definition of metric (ISO, 2001) is more suited to the **measurement protocol** that we define later in this sub-section.

An **indicator** (Ragland, 1995) is a device or variable that is set to describe the state of a process, based on its results, or occurrence of a predefined condition. The indicator provides insight into software development processes and improvements concerning attaining a goal. The indicator compares a metric with a baseline or expected result.

In the Practical Software Measurement (PSM) (McGarry et al., 2002) definition a **base measure** measures a single property or characteristic of an **attribute** that can be a product, process or resource. Base measures are used to calculate **derived measures** or, using the previous definition, metrics. A **performance indicator** is a measure that provides an estimate or evaluation of an attribute. The performance indicator is derived from a base measure or other derived measures. This means that even a performance indicator can be derived from other performance indicators. A **leading indicator** anticipates quality, as it is a measure that allows forecasting and diagnosis (Ferguson, 2010; Investopedia, 2007). On the other hand, a **lagging indicator** follows an event or tendency, therefore allows appraising (Investopedia, 2007).

Regardless the scientific field, **measurement** (Pfleeger et al., 1997) generates quantitative descriptions of key processes, products and resources, they are part of their outputs and are useful to understand their behaviour. The enhanced understanding of processes, products and resources is useful to better select techniques and tools to control and improve them. Pfleeger et al. (1997) consider that software measurement exists since the first compiler counted the number of lines in a program listing. In 1974, Donald Knuth reported on using measurement data, instead of theory, to optimise FORTRAN compilers, based on natural language. In the CMMI for development model constellation (CMMI-DEV), the **process measurement** is considered to be a set of definitions, methods and activities used to take measurements of a process and the corresponding products for

the purpose of characterising and understanding it (Chrissis et al., 2011).

One of the requirements for establishing a process measurement program is to put in place a measurement system. Kueng (2000) mentions two important characteristics of a measurement system: it shall focus on the processes and not in the entire organisation or in organisation units, and the measurement system shall evaluate performance holistically, by measuring quantitative and qualitative aspects. Kitchenham et al. (1995) enunciate some of the necessary concepts to develop a validation framework to help researchers and practitioners to understand:

- How to validate a measure;
- How to assess the validation work of other people;
- When it is appropriate to apply a measure according with the situation.

In the same work, Kitchenham et al. (1995) define **measurement protocols** as necessary elements to allow the measurement of an attribute repeatedly and consistently. These characteristics contribute to the independence of the measures from the measurer and the environment. The measurement protocol depends on how the measured value is obtained and on the use that will be given to the measure. A measure is therefore applied to a specific **attribute** on a specific **entity** using a specific measurement unit for a specific purpose.

Doing proper measurement and analysis is fundamental to evaluate processes and products development performance, and to improve them. Process measurement allows inferring the performance of the processes.

2.2 Process Performance Measurement and Improvement

CMMI-DEV defines **process performance** as a measure of the actual results achieved by following a process (Chrissis et al., 2011). It is characterised for both process measures and product measures. The process performance models depend on historical data of the processes performance and on which data is collected by the measurement system in place.

According with CMMI-DEV (Chrissis et al., 2011), the **process performance model** (PPM) describes the relationships among the attributes and the work products of a process. The relationships are established from the historical data of the process performance and the calibration of the model is done using data collected from the product, process and metrics of a project. Consequently, the process performance models are used to predict the results achieved by following the process that the model represents. Kitchenham et al. (1995) indicate that in predictive models, such as COCOMO (COnstructive COst MOdel), variability of the predicted values may occur, caused by the incompleteness of the model, as there are factors that affect what is being predicted that may not be considered in the model. The model error is the sum of the model incompleteness and the measurement error.

When the data collected on measures is stable and the process performance model is adequately supporting the prediction of the behaviour of the projects it is possible to understand the

normal behaviour of the process, i.e., under known circumstances. The **process performance baseline** (PPB) characterises the behaviour of the process, by establishing the maximum and minimum values where the process behaves under the expected causes of variation. If a project or process behaves out of the boundaries, by a certain threshold, the model shall allow the anticipation of that occurrence. In that case the team needs to identify the special causes of the variation. If the variation brings negative consequences then the team needs to act in order to prevent the deviation of the project or process. CMMI-DEV (Chrissis et al., 2011) defines process performance baseline as a documented characterisation of the actual results achieved by following a process. The baseline is used as a benchmark[1] to compare the actual performance of the process with its expected performance.

In his doctoral thesis, Dybå (2001) indicates that a broad definition of **Software Process Improvement** (SPI) would include the following activities:

- Define and model a software process;
- Assess the process;
- Refine the process;
- Innovate by introducing a new process.

Our perception is that to achieve process improvement it is necessary to measure the initial performance to compare it with the final performance. The objective of the process improvement is to improve the process performance. For that reason we introduce the term **Software Process Performance Improvement**. So Dybå's activities are here updated to include the term performance:

- Define and model a software process **performance**;
- Assess the process **performance**;
- Refine the process;
- Innovate by introducing a new process **or new process version**.

Considering that a process improvement must be measured and valuable, it has to result in performance gains, hence, a process improvement should not be done without considering that the process performance must be improved. Some may argue that a process improvement *per se* implies a performance improvement. Nonetheless, many organisations do "improvements" without measuring the performance of the process *as is* and the final performance. Moreover, even if a particular process improvement leads to its better performance it may have a negative impact in other processes that cannot be perceived if the organisation does not do an overall control. In fact, an improvement of a process may coincide with a process improvement project but may result from another process change that may have been planned or not. For these reasons we consider

[1] For a definition of benchmark please refer to Acronyms and Definitions.

that it is important to align process improvements with the organisation goals and focus in the right outcome.

The task of identifying performance indicators that can show how the process is performing is not trivial. First of all, performance indicators *per se* do not necessarily show if the organisation is doing better or worse. Those indicators need to be related with the organisation business objectives and that is what it makes metrics implementation a challenge. To make the task easier organisations can use tools such as the Goal Question (Indicator) Metric, Balanced Score Card or the Goal-driven Measurement (Park et al., 1996).

Process performance improvements result in updates in Process Performance Baselines and eventually in Process Performance Models. The Process Performance Baselines help defining Process Performance Models and there are bidirectional relations between what needs measurement and what builds measurement (processes, performance, models, baselines and improvements). CMMI has practices of Measurement and Analysis, at maturity level 2, and practices to build process performance models and establish process performance baselines in the the Organisational Process Performance process area (PA), at maturity level 4.

2.3 CMMI Architecture and Appraisal Method

CMMI has two representations designated continuous and staged (Chrissis et al., 2011). The **continuous** representation is organised in Capability Levels (CL), going from 0 to 3, while the **staged** representation is organised in Maturity Levels (ML) that range from 1 to 5. In our research we refer to the staged representation, because CLs are just applied to individual process areas, whereas MLs are applied across **Process Areas**. However, the framework we developed is usable in both representations as one organisation may select just a process area to evaluate or do a broader evaluation.

In figure 2.2 we present the CMMI maturity levels. To achieve a ML it is necessary to accomplish the Specific and Generic Goals of that ML and the precedent ones.

In the **Initial** level (ML 1) there are no formal processes (Chrissis et al., 2011). In ML 2, **Managed**, the processes are planned and executed according with the organisation's policy. The projects have documented plans necessary for their management and execution. At this ML, the process description and procedures can be specific of a project. The statuses of the projects are visible to management and commitments with relevant stakeholders are established and revised as needed.

In ML 3, **Defined**, the standard processes are used to establish consistency across the organisation and are continuously established and improved. The procedures used in a project are tailored from the organisation set of standard processes. The interrelationships of process activities and detailed measures of processes, work products and services are used to manage processes.

At ML 4, **Quantitatively Managed**, the organisation establishes quantitative objectives for quality and process performance. The projects have quantitative objectives, based on the goals of the organisation, customers, end-users and process implementers expectations. The projects

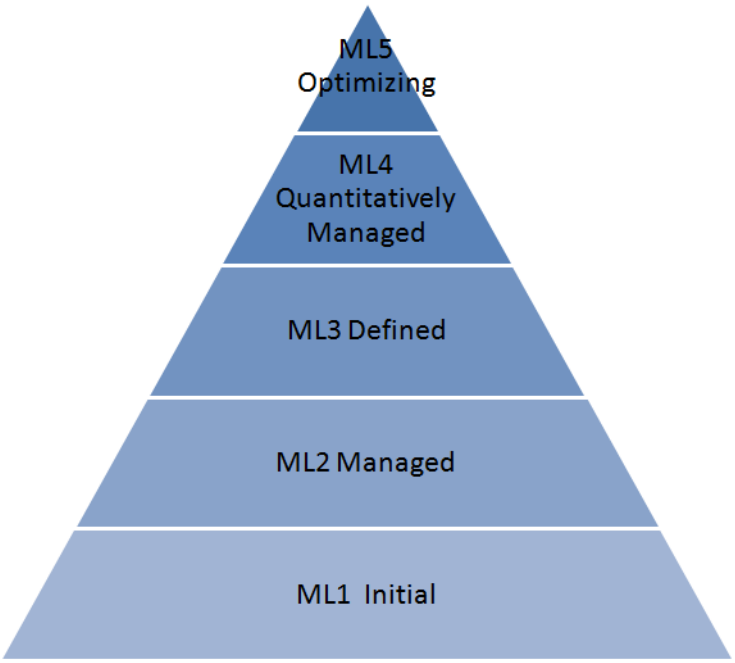


Figure 2.2: CMMI maturity levels in the staged representation.

selected sub-processes are quantitatively managed, i.e., the data of specific measures of the process performance are collected and analysed. The process performance baselines and models are developed by setting the process performance objectives necessary to achieve the business goals. The processes performance becomes predictable, based on the projects and processes historical data.

At ML 5, **Optimizing**, the quantitative understanding of the business objectives and performance supports the organisation improvement decisions. The defined and standard processes performance, the supporting technology, innovations and business objectives are continuously improved based on the revision of the organisational performance and business objectives. The improvements are quantitatively managed. Maturity Levels 4 and 5 are known as High Maturity Levels (HMLs).

In CMMI the process areas are organised in categories, namely **Process Management**, **Project Management**, **Engineering** and **Support** (Chrissis et al., 2011). The Process Areas include **Specific Goals** to accomplish, each of them presenting **Specific Practices** that help achieve those goals. Besides, at levels 2 and 3 the model includes **Generic Goals**, with the respective **Generic Practices**, applicable across process areas. When organisations are appraised at a ML or CL, the analysis is focused on what the organisations practices are to achieve the Specific Goals within that level.

SCAMPI is the method used to benchmark the maturity of a company in terms of the CMMI model (Davis and McHale, 2003). This method is used to identify strengths and weaknesses of the processes and determine the company's capability and maturity level. There are three SCAMPI

classes: A, B, and C. Class A is the most formal one, and is required to achieve a rating for public record. The other two apply when companies are implementing internal improvements at lower costs. In the remainder of this section we present two of the SCAMPI rules that we believe should be considered in the evaluation of CMMI implementations, i.e., not the rules focused on planning the SCAMPI but the **sampling factors** and **data sufficiency** rules[2].

It is not cost and effort effective to appraise an entire organisation and all its projects, therefore, it is important to have sampling rules that ensure that the subset of the organisation and projects that are appraised are representative of the overall organisation. Sampling organisation units is done by following the steps:

Sample Rule 1 - Understand the organisation unit and how it is organised. The organisation unit is composed by basic units and support functions;

Sample Rule 2 - Determine the organisation unit process drivers that influence how the processes are implemented;

Sample Rule 3 - Organise basic units and support functions into subgroups by applying sampling factors (e.g. location, customer, size, organisational structure and type of work);

Sample Rule 4 - Use equation 2.1 to determine the minimum representative sample that is collected from the subgroups and are included in the organisational scope.

A good principle is to evaluate elements of the organisation in proportion to their contribution:

$$MinimumNumberofBasicUnits: \quad MinBU = \frac{SubgroupsxBU}{TotalBU} \tag{2.1}$$

where **MinBU** is the minimum number of basic units to be selected from a given subgroup, **Subgroups** is the number of subgroups, **BU** is the number of basic units in the given subgroup and **Total BU** is the total number of basic units. When the computed value is less than 1 the required number of BU is 1, when is greater than one the number of BU is given by rounding the number to 0 decimal places.

The coverage rules (CMU/SEI, 2011; Byrnes, 2011) determine how much should be collected in the appraisal:

Coverage Rule 1 – Each basic unit or support function sampled must address all practices in the process areas for which they supply data.

Coverage Rule 2 – For each subgroup at least one basic unit shall provide both artefacts and affirmations. The sampled basic unit shall provide data for all process areas.

Coverage Rule 3 – For at least 50 percent of the basic units within each subgroup, both artefacts and affirmations shall be provided for at least one process area.

[2]The definitions of these terms can be found on Acronyms and Definitions.

Coverage Rule 4 – For all sampled basic units in each subgroup either artefacts or affirmations shall be provided for at least one process area.

Coverage Rule 5 – Both artefacts and affirmations shall be provided for each support function for all process areas relating to the work performed by that support function.

Coverage Rule 6 – The artefacts and affirmations provided by a support function shall demonstrate the work performed for at least one basic unit in each subgroup.

Coverage Rule 7 – In cases where multiple support functions exist within the organizational unit, all instances of the support function shall be included in the appraisal scope.

After all evidence are collected the appraisal team characterises the implementation of CMMI practices, for each model practice and each basic unit or support function. The practices implementation is classified as a **weakness** or a **strength**. Based on this classification each practice in each basic unit or support function is characterised as **Fully Implemented** (FI), **Largely Implemented** (LI), **Partially Implemented** (PIm), **Not Implemented** (NI) or **Not Yet** (NY)[3]. The rules for aggregating implementation-level characterisations to derive organisational unit-level characterisation are summarized in Table 2.1.

Table 2.1: Rules to aggregate implementation-level characterisations (CMU/SEI, 2011).

Characterisation	**Implementation**
Fully Implemented (FI)	All FI or NY, with at least one FI
Largely Implemented (LI)	All LI or FI or NY, with at least one LI
Largely or Partially Implemented (LI or PIm)	At least one LI or FI and at least one PIm or NI
Partially Implemented (PIm)	All PIm or NI or NY, with at least one PIm
Not Implemented (NI)	All NI or NY, with at least one NI
Not Yet (NY)	All NY

If any practice is not characterised as FI it is necessary to explain the gap between the organisation practice and what the model expects (CMU/SEI, 2011). A weakness, "ineffective, or lack of, implementation of one or more reference model practices", is only documented if it has impact on the goal. A goal is rated **Satisfied** if and only if all associated practices at organisational unit level are characterised as LI or FI and the aggregation of weaknesses of the goal do not have negative impact on its achievement.

CMMI establishes quality principles - what to do. It presents guidelines in a set of good practices to achieve goals that also concern the organisation as a whole, but does not define the processes. The Team Software Process (TSP) is used together with the Personal Software Process (PSP) providing organisations disciplined processes to be used by individuals and teams (Davis

[3]The definitions of these terms can be found on Acronyms and Definitions.

and Mullaney, 2003). PSP was defined by Humphrey himself while developing 60 software programs applying all SW-CMM (Software Capability Maturity Model) practices up to level 5. While PSP is used by individuals TSP helps them work together as self-managed teams. TSP shows how to do things - steps to do the work and forms to register plans and work execution information.

2.4 TSP Architecture and Certification

TSP includes process scripts to achieve high maturity performance that are followed by self-directed teams, i.e., teams where decisions are made together. In such teams anyone can assume the roles necessary for project completion, there is a team leader, who is part of the team, and a coach, who is an observer helping individuals and team to be on track. "All team members participate in planning, managing and tracking their own work" which is "key for high motivation and high performance in knowledge-based work" Faria (2009). The team members shall receive training in Personal Software Process (PSP), so that individual team members skills are built.

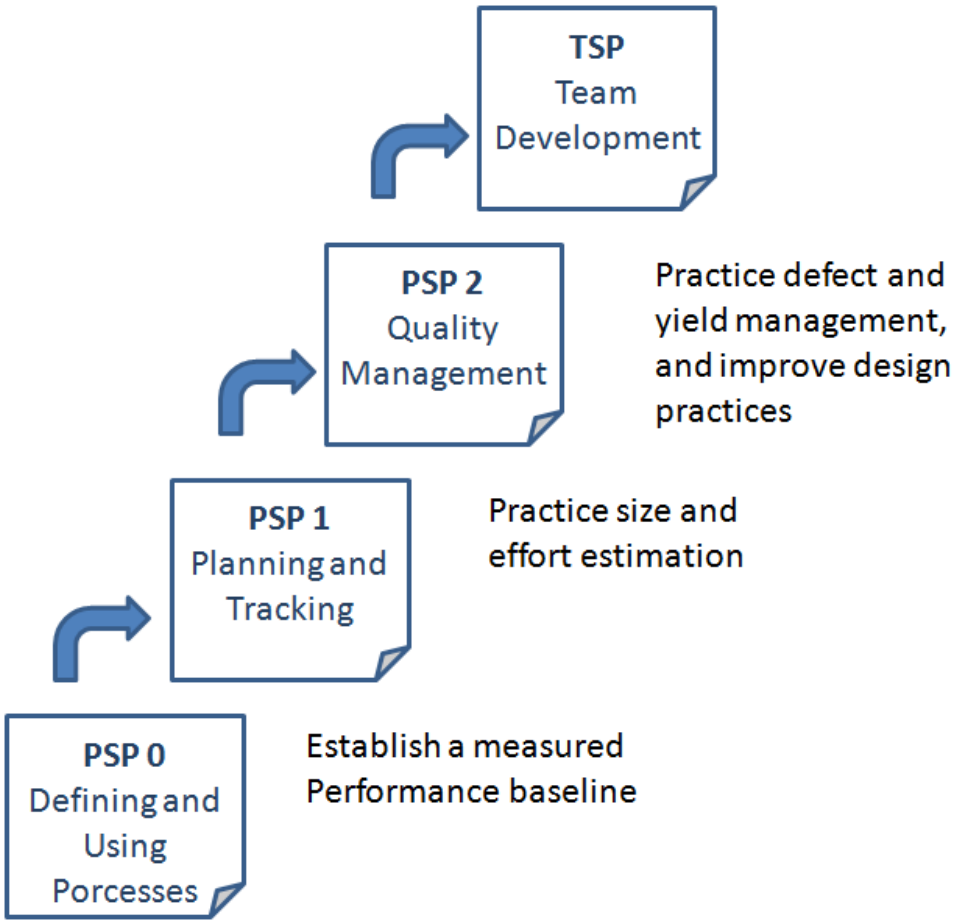


Figure 2.3: Personal Software Process (PSP) training: introduced stepwise in a sequence of small projects; people get convinced by seeing their performance improved with practice (Faria, 2009). The last step is Team Software Process (TSP^{SM}) training.

"PSP training is a path for self-improvement and discipline, consisting of developing several products and improving the process by adding quality practices. The training is done in steps (figure 2.3) beginning with **PSP0**, where developers write their development process and follow it. During development, programmers record information about the program: size, development time, and number of defects (Davis and Mullaney, 2003). At the beginning of **PSP1** programmers use the data collected in the previous phases to plan their work, estimate the necessary effort to develop the program and its size. Such data is used in **PSP2** to alert programmers for their mistakes, when they are made and the quality practices to avoid them. They plan expected number of defects inserted and removed in each development phase, hence learning to manage defects and yield. PSP training towards expertise goes from simple and unplanned to complex and predictable. "I find that PSP is the exact application of engineering to software and, as van Vliet (2007) stated "discipline is one of the keys" to successfully complete a development project. The benefit of PSP is that programmers see their results improving during training; at the end their skills and programs are unequivocally better. The processes are embraced not imposed. Furthermore, programmers become aware of their personal data, allowing them to set individual goals for improving their own development process towards making better products faster." (Lopes Margarido, 2013)

The introduction of TSP in an organisation is done gradually in two phases, the **Pilot** and the **Roll-out**. The pilot phase includes training a controlled small number of project teams and their managers, launch the teams with TSP followed by a TSP coach, execute the project using TSP and gather data that will support results evaluations at the end of the project (Faria, 2009). The roll-out phase requires the train and certification of internal TSP trainers and coaches to gradually launch additional teams at a sustainable pace. As people integrate new teams they bring in their knowledge and experience using TSP.

The basis of TSP teams is individual team members that have built their skills and completed PSP training. The team is built in the project launch week, by setting goals, assigning roles, tailoring the team's processes and designing detailed balanced plans with the active participation of all team members. Team management is done by managing communication and coordination, tracking the project and analysing risks. The project is developed and planned iteratively, it has a launch and a post-mortem meeting and is done in cycles, each of them beginning with a launch/relaunch[4] meeting and finishing with a post-mortem meeting. This iterative development should be based on the most adequate development model, which is determined by the technical and business context of the project (Faria, 2009). It can be done in small iterations, using spiral with increasing functionalities and/or complexity, or it can be sequential, following a waterfall model. The status of the project is reviewed weekly.

A TSP project includes different phases:

- **Planning**, done in the launch and relaunch meetings, in which all team members participate, so everyone's commitment is assured. Roles are assigned, the processes are selected, size of work products and effort to do tasks are estimated;

[4] In case of not being the first launch meeting

- **Development**, not only of the Code itself but eliciting Requirements, doing High-Level Design and Detailed Design;
- **Defect Removal**, which includes PSP processes of personal review, unit testing and compile, and also inspections, peer-reviews, integration and system testing.

TSP and PSP require collecting "four core measures, which are the basis for quantitative project and quality management and project improvement, at personal, team and organisation levels Faria (2009)". The actual and planned of size, effort, quality and schedule are recorded and controlled.

TSP-PACE (or just PACE), TSP Performance and Capability Evaluation, is the process to evaluate the TSP data of software development organisations and programs towards the TSP certification of organisations (Nichols et al., 2013). The program certification involves assessing the quality of the following TSP elements:

- Data;
- Training;
- Coaching;
- Launches and relaunches;
- Post-launch coaching;
- Project cycle postmortem reports;
- Project postmortem reports.

For organisational certification the teams under the certification scope are assessed in the aforementioned dimensions and the organisation is also assessed in **scope**, quality and quantity of the gathered data, and **customer satisfaction data**, regarding the work done by the TSP teams. All data are analysed in five dimensions, that constitute the profile: coverage, process fidelity, performance, costumer satisfaction and overall. Each profile has multiple variables that are evaluated.

2.5 Effort Estimation in CMMI and TSP

The CMMI Project Management category includes the Project Planning (PP) Process Area (Chrissis et al., 2011). Project Planning is among the first activities necessary to start a software development project and plays an important role in the course of the project. The plan can be revisited whenever necessary, being it because of the process used is done in cycles at the beginning of which detailed estimates are provided for the necessary tasks to execute them, as in Scrum or TSP, or just because the scope changed and it is necessary to re-plan to accommodate the necessary activities. Therefore, project planing plays an important role not only on the execution of the project but also on the quality of the outcomes. As a CMMI process area Project Planning is part of ML 2

with the purpose of establishing and maintaining plans that define the project activities (Chrissis et al., 2011). We include the specific goal and practice summary of this on table 2.2 to show what goals are expected to be achieved when planning projects following CMMI, in order to be able to develop the project plan, involve relevant stakeholders, have the team commitment to the plan and maintain the plan.

Table 2.2: Specific Goals and Practices of Project Planning (Chrissis et al., 2011).

SG 1	Establish Estimates
SP 1.1	Estimate the Scope of the Project
SP 1.2	Establish Estimates of Work Product (WP) and Task Attributes
SP 1.3	Define Project Lifecycle Phases
SP 1.4	Estimate Effort and Cost
SG 2	Develop a Project Plan
SP 2.1	Establish the Budget and Schedule
SP 2.2	Identify Project Risks
SP 2.3	Plan Data Management
SP 2.4	Plan the Project's Resources
SP 2.5	Plan Needed Knowledge and Skills
SP 2.6	Plan Stakeholder Involvement
SP 2.7	Establish the Project Plan
SG 3	Obtain Commitment to the Plan
SP 3.1	Review Plans That Affect the Project
SP 3.2	Reconcile Work and Resource Levels
SP 3.3	Obtain Plan Commitment

Focusing of the estimation process, to establish the estimates (SG1) of project planning parameters, that need to be identified, have sound basis, consider project requirements from relevant stakeholders, be documented along with the information sustaining the estimates and have team commitment. The goal can be accomplished by following SP1.1. to SP1.4. All the estimation is done under the project scope (SP1.1), that is broke down, and depends on the estimates of the WP and task attributes (SP1.2) and on the definition of the project lifecycle phases. Therefore, SP1.4"Estimate Effort and Cost" is done using the prior SPs as inputs.

TSP provides a project planning framework compliant with the CMMI and a set of planning guidelines to support planning (Humphrey, 2006). Ideally, the teams use their historical TSP data but if unavailable the values in the guidelines can be initially used. In case the data are not adequate for that project/team case it is recommended they use their best estimate. TSP teams will rapidly get their own data to estimate next because they will be gathering data as they work and feeding their historical database.

There are several TSP plans produced. The **overall plan** is produced by the team and is later adjusted once the next phase balanced plan is done. The **bottom-up plan** is done considering task decomposition for the next phase and is done by each team member individually. That plan is then load balanced to consider the individual plans and produce the team's **balanced plan** for the next phase.

TSP does not provide initial values to estimate Requirements, Requirements Inspections and High-Level Design. However, guidelines are provided in the Quality Plan (Humphrey, 2006, pages 148, 149) to define the time balance between inspection and development of requirements and high level design, for example, the guideline for the ratio between detailed design and coding time higher than 1.

For the implementation phase, TSP provides guidelines for the development rate per hour of the total code, which requires estimating the product/component size; percentage of time spent in phase (Humphrey, 2006, page 131); and ratio between defect removal and corresponding defect insertion phase. The implementation phases are Detailed Design, Detailed Design Review, Detailed Design Inspection, Coding, Code Review, Compiling, Code Inspection and Unit Test.

In TSP the method used to produce the estimates of size and time are is the one used on PSP, the PROxy-Based Estimation (PROBE) (Humphrey, 2005). The method is based on the definition of any item that can be used as a proxy. It requires that the team has the capability to provide an initial draft of the detailed design that allows them to estimate the size (added, modified, removed and actual) of the parts that compose the solution to develop. When there is no historical data, expert judgement is used to estimate size and time, PROBE D. When there is some data that can be used size is estimated based on it and so is time, PROBE D. PROBE C uses historical data to estimate size and time and in a regression model with correlation higher than 50%. While PROBE A, the desired method, uses historical data of the proxy to do the estimation using a linear regression model when the correlation is higher than 50%.

Chapter 3

State of the Art

This chapter presents the results of the literature review done to define the problem, understand the previous contributions of other researchers, and identify the open ends of the problem that still need to be addressed. This is needed to understand the ground basis of our work and the areas to which other researchers already contributed.

3.1 Related Work on Process Improvement

In this section we discuss the evolution of metrics programs and the CMMI model pointing the problems that persist. We then analyse which frameworks exist to help solve the problem and indicate the limitations that remain. By the end of the section the motivation for our research and the problem we tackled shall be clear.

3.1.1 Historical Perspective on CMMI and Metrics Programs

In 1978 a group of Hewlett-Packard (HP) engineers went to Japan to study the techniques used in manufacturing that led to significant improvements (Grady, 1992). Those techniques were analysed to realise how they could be applied to the HP's software development processes. The HP experience is a good example of the implementation of high maturity practices even before the idealisation of the CMMI level 5 [consultant personal statement, 2009].

According to Jones (1991), since 1979 it has been possible to have stable metrics and accurate applied measurement of software. Organisations such as IBM, Hewlett-Packard, Tandem, UNISYS, Wang and DEC, measured productivity and quality and used data to make planned improvements. Du Pont, General Electric and Motorola were innovative in software measurements. ITT, AT&T, GTE and Northern Telecom were pioneers in quality and reliability measures. Several management companies such as Software Productivity Research; DMR Group, Peat, Marwick & Mitchell, Nolan, Norton & Company, and Ernst and Young, were more effective than universities in using metrics and in transferring the technologies of measurement throughout their client base.

Phil Crosby assembled a maturity framework in 1979 and at the beginning of the 1980's IBM begun the assessment of the capabilities of many of its development laboratories. Later, the SEI

incorporated Deming[1] principles and Shewhart[2] concepts of process management, published by Deming in 1982 (Humphrey, 1992). Crosby's framework originated an assessment process and generalised maturity framework that was published by Radice in 1985. In April 1986, Watts S. Humphrey stated in the IEEE Spectrum that complex systems could be programmed with high quality and reliability if were done by strong technical teams using a highly disciplined software process (Callison and MacDonald, 2009). The concepts of the SEI and MITRE Corporation were compiled in a technical report by Humphrey that was published in 1987 (Humphrey, 1992). In 1989, Humphrey added that the problems in software engineering are not technological but have a managerial nature (Dybå, 2001). The Personal/Team Software Process were released in 1990 (Callison and MacDonald, 2009) and in 1991 Mark Paulk clarified the content of the maturity framework with the publication of the Capability Maturity Model by the SEI (Humphrey, 1992). Since its creation until our days the CMM evolved giving place to the CMMI, which is now in its version (V) 1.3. The evolution of the model is represented in the chronological diagram in Figure 3.1.

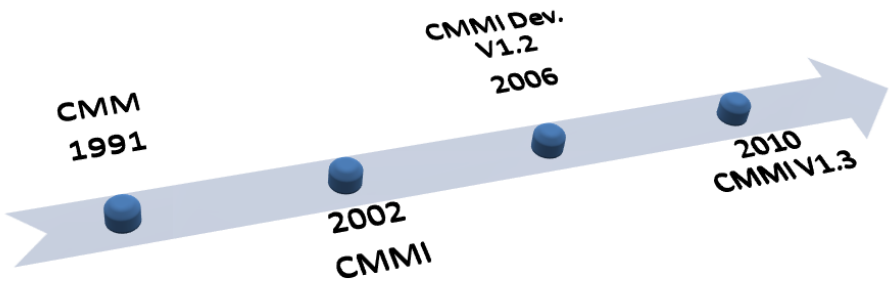


Figure 3.1: Subset of the CMM(I) releases.

3.1.2 CMMI and TSP

The implementation of the CMMI maturity level 5 includes a set of demonstrated benefits for the organisations (Goldenson et al., 2004):

- Improve the quality of the products and processes;
- Reduce the development cost and the schedule;
- Increase the customer satisfaction;
- Add predictability to the processes;
- Reduce rework by reducing the quantity of defects detected later in the life-cycle and that imply spending time in finding and correcting them.

[1] Deming espouses the Shewhart concepts (Humphrey, 1992).
[2] Plan, Do, Check, Act cycle, is the foundation of process improvement work (Humphrey, 1992).

However, not all organisations achieve the same results. The 2003 TSP report indicates that the defect density (number of defects per a thousand lines of code) in products delivered is lower in organisations using TSP than in CMM level 5 organisations. The evidence is graphically represented in Figure 3.2 that shows the number of defects per a thousand lines of code (KLOC) delivered to the customer.

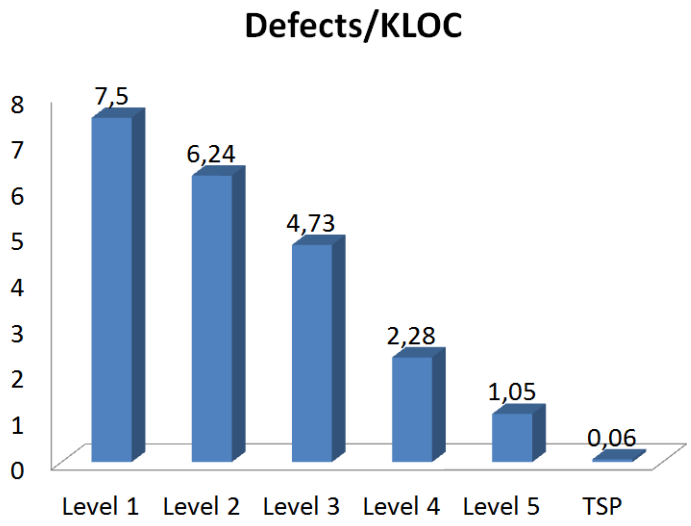


Figure 3.2: Average defects per thousand lines of code of delivered software in TSP and CMM different maturity levels (Davis and Mullaney, 2003).

From these results we could consider that since TSP performs better in quality than CMM, it would be preferable to use TSP rather than CMMI. However, the question is whether they are even comparable. In fact, regarding CMMI, the model is not prescriptive and for that reason the organisations using it present different performance results ()such variance depends on the methods used to implement it). Furthermore, TSP only covers part of the CMMI practices.

The Accelerated Improvement Method (AIM) implementation guidance (McHale et al., 2010) provides information on TSP evidences that allow achieving CMMI ML3, but ML4 and 5 are not yet covered. The AIM is a process improvement initiative that combines the best of CMMI, TSP and Six Sigma measurement and analysis techniques. The new TSP version implements 70% of the specific practices up to maturity level 3 (SEI, 2010).

Webb et al. (2007) extended TSP to directly address the CMMI process areas that are not completely addressed by TSP practices by publishing additional process scripts, items that were added to existent processes, metrics and requirements.

PSP and TSP are the application of high maturity to teams (Davis and Mullaney, 2003), while CMMI defines the capability of an entire organisations. They have different natures and purposes: CMMI gives guidance and organisations are free to select the most adequate methods

to implement the practices; TSP provides all necessary tools to have a mature process in place and improve it. To help us understand why organisations using CMMI can have different performance, and sometimes even an unacceptable one, we analyse in the next section the problems that can arise while implementing process improvements, metrics programs and CMMI.

3.1.3 Problems in Process Improvements, Metrics Programs and CMMI

A survey was conducted to understand what CMMI level 4 and 5 companies used in the CMMI implementation, which showed that practices were not clearly institutionalised (Radice, 2000). The SEI concluded that some companies did not understand the statistical nature of the CMMI level 4 and certified CMMI HML companies did not have a consensus on the necessary characteristics of level 4 (Hollenbach and Smith, 2002). Even more recently, the performance of high maturity organisations is being questioned (Bollinger and McGowan, 2009).

In 2004 the Department of Defence (DoD) of the United States of America (USA), pointed out problems resultant from having CMMI levels (Schaeffer, 2004). Among other problems, it was recognised that not all programs of the organisations are appraised. Therefore, practices were not implemented organisation wide and organisations let the baselines erode once they achieved a certain maturity level. In response to the DoD problem, Pyster (2006) proposed a set solutions, of which the following are examples (the ones related to acquisition are identified with the word in parentheses):

- Guaranteeing that when contracting an organisation with a certain maturity level, the performing team uses the maturity processes being referenced (acquisition);
- Doing periodical appraisals after the contract to ensure that the tailoring of adequate processes for the specific program include adequate high maturity processes (acquisition);
- Recognising that CMMI is relatively new and will take time to "fully permeate companies";
- Improving the appraisals by providing guidance on how to select representative samples and aggregate results from subordinate organisations, when appraising large organisations.

The last suggestion that Pyster made may improve the SCAMPI and avoid the certification of organisations where the practices are not institutionalised. The other suggestions are important for the contractors that demand that their suppliers have a specific CMMI maturity level, but do not tackle the root problem that may exist either in CMMI or in SCAMPI, or perhaps in both.

Many companies face problems when implementing CMMI HML that arise from complex practices such as measurement and quantitative management or the use of effective performance models for predicting the future course of controlled processes. In fact, part of the difficulties found in the processes evolution and new Process Areas implementation are related to the need to move towards a statistical thinking and quantitative management (Takara et al., 2007).

Kitchenham et al. (2006) analysed a CMMI level 5 corporation's database to find that data were collected but metrics could not be correlated and did not have meaning for upper management. According with Monteiro et al. (2010), some authors (Hall et al. 1997; Berander and

Jönsson 2006; Agresti 2006) argue that several measurement programs in organisations fail because they define too many measures that are not actually implemented and analysed in decision making. Kitchenham et al. (2006) disclosed important concerns that shall be taken into consideration when storing data and what needs to be considered when designing the database, so data analysers and decision makers can actually make use of them. They also proposed the use of the M3P framework that extends the GQM (Goal Question Metric) by providing links between the collected metrics, the development environment and the business context.

Regarding the metrics definition, it is important to understand how the data is collected and analysed, what are the common and special causes of variation. As already mentioned many metrics exist, expressed in natural language or the values that allow their calculation are expressed in natural language (Goulão, 2008). Breuker et al. (2009) mention there are different definitions of the same software metrics in the literature (books and papers), tools for metrics collection's specification and the tools that actually collect the metrics. Literature needs to clearly define software metrics and practitioners shall be aware of this problem when implementing the measurement and analysis system.

Leeson (2009) added more problems that can occur in CMMI implementation, that we compile in table 3.1. Those problems are classified as Program Management to refer to the management of the implementation of CMMI, Maturity Level 2 and Maturity Level 3, to refer to problems that occur in the implementation of those ML.

In her doctoral thesis, Barcellos (2009) states that metrics programs are failing because they are producing metrics that do not allow the analysis of the performance and capabilities of their processes (Goh et al. 1998; Fenton and Neil, 1999; Niessink and Vliet, 2001; Gopal et al., 2002; Wang and Li, 2005; Kitchenham et al., 2006; Sargut and Demirors, 2006; Curtis et al., 2002; Rackzinski and Curtis, 2008). In her literature review, she also states that there are problems of metrics adequacy (Wheeler and Poling, 1998; Kitchenham et al., 2006; Tarhan and Demoirs, 2006; Boria, 2007; Kitchenham et al., 2007; Curtis et al., Barcellos, 2009).

The SEI and the Systems and Software Consortium, Inc. (SSCI) published a report in 2009 reporting reasons contributing to the success of programs. The authors concluded that people acting as individuals or as teams contribute to program success or failure (SEI and SSCI, 2009). The main points were decision making, communication, teams experience, adequate coaching and understanding the program purpose and goals. It is recognised that a good process is not enough for a program to succeed and the factors that are considered as overriding the above mentioned are "effective leadership and objective governance for the program" and "willingness and ability of program personnel to think through problems and tailor the prescribed process to the needs of the program".

Even though the SEI, and now the CMMI Institute, publish good performance results from organisations that implement CMMI process improvement programs to ensure that CMMI users are benefiting from using the model, all the problems mentioned in this section are the result of

Table 3.1: Some of the problems identified in the implementation of CMMI (Leeson, 2009).

Program Management Problems
Senior management is not involved in establishing the objectives, policies and the need for processes.
Sponsor does not play its role and delegates authority.
Software Engineering Performance Group is not managed.
Organisations are focused in achieving a maturity level more than improving the quality of their products or services.
Maturity Level 2 Problems
Organisations lack a global view of the model. Organisations do not understand the relationship and differences between measurement and project monitoring, GP (Generic Practice) 2.8, 2.9 and 2.10, process areas and practices, maturity levels and capability levels.
Some organisations misinterpret ML 2 and 3, which causes the failure of many programmes.
Assume that ML2 is only for project managers. Developers and engineers need to be involved.
Measurements are not related to customer and business objectives.
Quality Assurance is focused on product compliance instead of assuring the quality of the processes.
Maturity Level 3 Problems
Some organisations define theoretical processes that do not correspond to actual activities to achieve ML 3.
A communication process is not established. So experiences and suggestions from people who know how to do their job are not gathered and consequently are not fed back.
Organisations do not consider HML, when implementing ML 3, and fail to understand the end-picture not seeing the direction they are taking at lower levels before moving to HML.

a deficient implementation of CMMI. Such problems become more nefarious when they are not detected in the appraisal.

3.1.4 SCAMPI Limitations

The SCAMPI appraisal method has already missed implementation problems. After analysing the description of the method, some of its features can be regarded as limitations that may be in the origin of this problem.

Appraisal Team Quality

Armstrong et al. (2002), in their presentation of the changes and features of the SCAMPI V1.1, stated that the appraisal method is focused on practices implementation. That is, the SCAMPI appraisal team is focused in verifying whether the practices are implemented or not. The objective of the appraisal is not to verify how people are actually doing the work or the quality of their results. With such an orientation malpractices may be missed by the appraisal team. In fact, the appraisal results reflect the knowledge, experience and the skill of the appraisal team (CMU/SEI, 2011).

Organisation Honesty

SCAMPI relies on the organisation honesty, which provides evidences and supports the choice of the projects that are going to be appraised (Armstrong et al., 2002). Either the lead appraiser is very rigorous in the choice of the projects and critique about the evidences or the outcome of the appraisal may be biased by the organisation.

Limited Number of Affirmations

In the appraisal only a small number of affirmations sustain the practices. In version 1.2 to classify a practice as fully implement, and therefore contributing to the level achievement, a direct and indirect artefact could suffice (CMU/SEI, 2011). Back in 2003, Radice described a 50% : 50% rule that stated that there should be an affirmation in 50% of the practices and 50% of the projects covering one row per one column (Radice, 2003). The SCAMPI V1.3 coverage rules, that we previously described, also limit the number of affirmations. We consider that an affirmation from a single BU, that does not come out on an interview because the coverage rules do not demand it, could suffice to demonstrate that the BU was not following one of the practices.

Coverage of the Organisation

As mentioned by the DoD, not all programs of the organisation are analysed in the appraisal (Schaeffer, 2004). During the appraisal only a small percentage of the organisation projects and business units may be appraised, therefore it is difficult to have guarantees that the entire organisation is working in the same way in all projects or programs – that means that the practices may not be institutionalised. In face of the limitations of SCAMPI V1.1 description in providing guidance to the selection of the projects for the appraisal, Moore and Hayes (2005) proposed the application of Design of Experiments (DOE) to construct a representative sample of the organisational units being appraised. DOE is a statistical technique that helps understanding the influence of the different experimental factors on the response of the system. When applied to the SCAMPI the method allows an accurate appraisal planning and execution, as it supports the construction of a representative sample of the organisational unit and the selection of the personnel to interview and questions to be asked when collecting affirmations. The projects are selected from the analysis of the influential factors of the organisation unit. In order to reduce the number of projects the authors propose the use of fractional factorial design of instantiations and apply the method in sequence (early SCAMPI C's and later SCAMPI A's or B's) in order to eliminate factors based on results. Later, Moore and Hayes (2006) presented useful information on how to use the previously mentioned method, DOE, to select the most appropriate projects for an appraisal, fulfilling the SCAMPI V1.2 description. SCAMPI V1.3 clearly defines the sampling rules.

Evidence Collection

Pricope and Horst (2009) indicated that the SCAMPI is described in natural language and does not provide activity-oriented graphical description of the appraisal process. For those reasons the authors proposed a method to measure SCAMPI appraisals by using the Unified Modelling Language (UML) to represent the metamodel of the SCAMPI. The metamodel includes all the

SCAMPI elements, such as types of evidences, activities performed in the appraisal, roles, etc. Besides the model, the authors proposed quality metrics to evaluate the appraisal, introducing a quality metric for activities. The metrics allow determining the level of weakness or strength of the appraisal elements. The method proposed by Pricope and Horst introduced the quantitative novelty in the SCAMPI and is useful for quantifying the appraisal that was conducted. However, it does not evaluate how practices are being actually done nor evaluates the organisation performance.

Sunetnanta et al. (2009) proposed a model that constitutes a Configuration Items (CI) repository, where all projects configuration items are pooled together. The authors' idea was to use this repository in organisations working with different offshore units but we consider that the model is applicable to any organisation. The CMMI appraiser needs to set up the rules to identify the projects' CI that constitute evidence of the sub practices of the model. The repository allows the collection of evidences as the projects are ongoing, as well as analysing the projects and appraisal results. The quantitative assessment of the projects is done by score, i.e. number of times an activity is executed, and by compliance, i.e. by checking if the activity was executed or not. By the time of the appraisal all the evidences are already available. There is a limitation on the method that the authors do not mention, though. The evidences still need to be evaluated and analysed by the appraisal team. A CI may be generated but if it is empty it shows that the expected activity was not performed, and even when generated it is necessary to assess whether people actually did the practice or just produced an artefact that is mandatory.

Pilots on Results-based Appraisals

McCarthy (2009) reported results of pilot projects being assessed with results-based appraisals, including the Telecommunication Quality Management System – TL 9000 standard, to identify and validate an appraisal method that would assess performance measures. The information collected on SCAMPI would be useful to trigger the appraisal team for further investigation in face of unexpected performance, have results-oriented findings, have records for posterior assessments and recommendations related to performance and benchmarking. From the identified challenges the appraisals took longer (more 5% to 10% when compared with a regular appraisal) and became more expensive. The industry benchmarks varied in value which raised doubts of their applicability. Besides, the measurement repository built in the pilot environment had no documented linkage to processes and practices in standard process or in the CMMI.

CMMI has been facing the paradigm of how well high maturity organisations are performing since quite a long time (Bollinger and McGowan, 2009; Campo, 2012). SEI released version 1.3 that included improvements in the model, focusing in the performance of the organisations (Phillips and Shrum, 2010), but the new SCAMPI version still does not measure their performance.

3.1.5 CMMI V1.3 Changes

The CMMI product team worked on the definition of the version 1.3 of the CMMI constellations and the SCAMPI. The new version of the CMMI model was designed to be more compatible with the multi-model tendency that has been occurring. Figure 3.3 depicts the relationships between different models. To implement good practices organisations follow quality principles, such as CMMI constellations, ISO standards and the Project Management Institute (PMI) documentation. To know how to follow the quality principles organisations use operational practices, for example TSP, Agile and the Information Technology Infrastructure Library (ITIL). We consider that methods also include techniques and procedures used to generate the products and/or services. Organisations also use improvement techniques that help them evolve and shape their processes, such as Lean, Six Sigma and Theory of Constraints. We consider that those improvement techniques can also be used to define organisations processes.

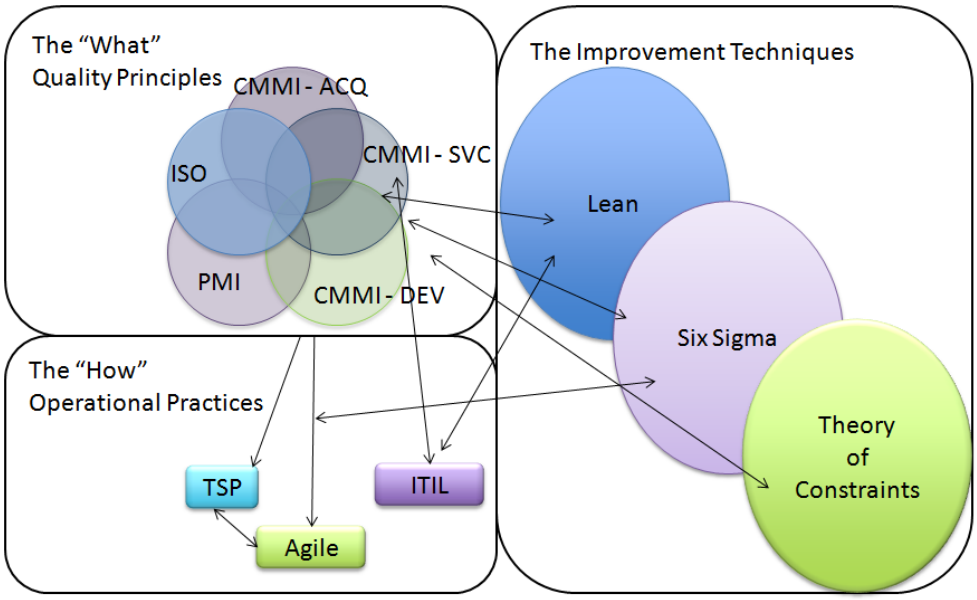


Figure 3.3: Multi-model representation (Phillips, 2010).

With the new version of the CMMI, the SEI intended to do the following improvements (Phillips, 2010):

- Simplify/clarify the terminology;
- Update selected process areas to provide interpretation of practices for organisations with respect to Agile methods, quality attributes, product lines, systems of systems, customer satisfaction, among others;
- Simplify Generic Goals and remove the ones of the high maturity levels;

- Clarify High Maturity concepts, such as process models and modelling, business objectives thread to high maturity, common causes, high maturity expectations in individual process areas, among other changes;
- Add a new process area in ML 5, called OPM (Organisational Performance Management), that substitutes OID (Organisational Innovation and Deployment);
- Revise QPM (Quantitative Project Management) SP to reflect the connection between CAR (Causal Analysis and Resolution) and QPM;
- Lighten the model in number of pages, generic goals and practices, and specific goals and practices.

Figure 3.4 represents the combination of OPM and OID that gave origin to OPM. With OPM the improvements are driven by the intention to achieve quantitative objectives of quality and process performance. The drivers of the improvements will not only be the organisation but also the customer.

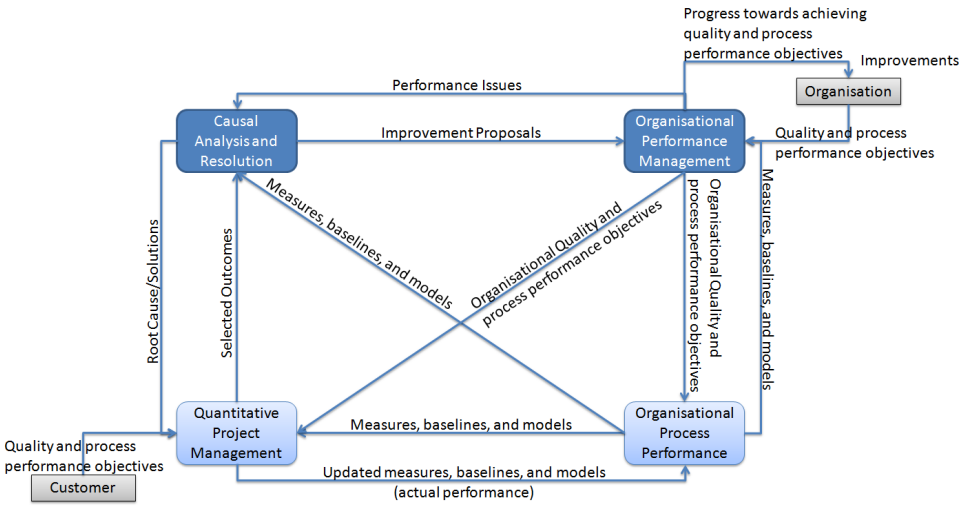


Figure 3.4: Representation of OPM and OID (Phillips, 2010).

The version 1.3 of the SCAMPI has the following changes (Phillips, 2010):

- Includes details on the definition of the appraisal scope and on how to sample the organisation units and projects;
- Clarifies doubts per example about direct and indirect artefacts;
- Improves the appraisal efficiency.

Regardless the improvements in defining the scope and collecting enough evidences SCAMPI appraisers raised some limitations on the coverage rules. One example was given by Heather

Oppenheimer, on August 2011[3], stating that for some support functions some activities of a PA are assumed by one group and others are used by basic units because they support their work. This means that the support function cannot provide all the evidences for the PA. We also consider that there is another problem with the coverage rules. It is possible to leave lack of institutionalisation undetected, because not all basic units need to provide artefacts/affirmations. Some of them may not be doing a PA that regards their work at all.

The CMMI model evolved to focus on the improvements in the performance of the organisations, paying more attention to results. This may help to avoid implementations more oriented to achieving a maturity level rather than improving organisation's performance. The model was also updated to cope with multi-model environments. Having guidance for different models may prevent poor practice's implementation.

SCAMPI is addressing part of its problems, in particular by better defining the scope of the appraisal, i.e. how to sample from the organisational units and how much evidences are necessary. However, we consider that the implementation problems may remain undetected and the performance problems persist, because the SCAMPI is not evaluating the implementation performance, which is out of its scope. For this reason it is necessary to do further research in this area in order to have a framework that measures organisations performance and evaluates the quality of implementation of the CMMI practices.

3.1.6 Methods and Models for Process Measurement and Evaluation

Next we present other research contributions that may help understanding and solving the problem addressed in this work. With this analysis we clearly define what was is still needed to tackle the problem and motivate our approach.

Metrics Definition

In 2009 the SEI and the Object Management Group (OMG) announced the creation of the Consortium of IT Software Quality (CISQ). The CISQ is sponsored by OMG and the SEI is working with them in the development of software-related standards and appraiser licensing programs. The metrics would be unambiguously defined contributing to the possibility of automate the measurement and analysis process (Curtis, 2010). The consortium published code quality standards to be consistently applicable to any organisation: the Automated Function Points (AFP) (CISQ, 2014), as a standard measure of size, with rules to measure different software code files, and Automated Quality Characteristic Measures (CISQ, 2016) conformant with ISO/IEC 25010 quality characteristics of security, reliability, performance efficiency and maintainability, and providing "measures of internal quality at the source code level". CISQ is currently developing Automated

[3] http://www.linkedin.com/groupAnswers?viewQuestionAndAnswers=&discussionID=64637798&gid=54046&commentID=54099629&trk=view_disc&ut=0b8g1zukIS5R01 – last accessed on 17-11-2011.

Enhancement Points, a measure of size to be used in productivity analysis, Technical Debt, measuring the cost, effort and risk of the remaining defects in code at release, and Quality-Adjusted Productivity to consider the quality in the measurement of productivity.

An anonymous research group from Switzerland worked on the definition of metrics to assess the quality of the CMMI implementation. The results of their work were not published since they concluded that such task would be impossible to execute. Their justification for that is that it is difficult to define metrics applicable to all organisations, because each organisation has its own business objectives. A member of the group shared with us the unpublished documentation of their work (Group, 2007). They did an exhaustive work identifying metrics that would allow the control of 11 process areas that the researchers referred to as processes. One of the problems in the research work is that, instead of first reviewing the literature in search of the metrics for the processes that they intended to monitor, they opted to introduce new ones. There is a large number of metrics identified in software engineering and most of them is never used. We noticed that the metrics are described but the goal that would support each metric is ignored. In our opinion, if the metrics defined by the research group are of use they need to be mapped with the questions that the organisation would want to have answered in order to verify that a certain objective would be achieved. The way the metrics document was built reveals that the usage of the BSC and GQM was not considered.

With the existence of a size measurement standard, already released by CISQ, and future releases of productivity standards that take the quality of the developed code into account, organisations can collect data, the same way, making it comparable. Moreover, the fact that productivity will be based on the quality of the produced work makes the metric more useful and relevant. This will facilitate the task of benchmarking organisations performance. Which will help overcoming some of the aforementioned limitations of process performance assessments.

Metrics Analysis

The SEI Software Engineering Measurement and Analysis (SEMA) group (SEI, 2016) publishes the results of the state of measurement and analysis practices and conducts research of the most effective ways to implement measurement and analysis processes from the Process Area MA (Measurement and Analysis) at ML 2 to the high maturity techniques that are necessary to implement levels 4 and 5. SEMA developed the Quantified Uncertainty in Early Lifecycle Cost Estimation (Ferguson et al., 2011) that considers "program change driver uncertainties common to program execution in a DoD Major Defense Acquisition Program lifecycle". The method includes the use of Bayesian Belief Networks,to generate likely scenarios and Monte Carlo Simulation to estimate the distribution of the program cost.

The Performance Benchmarking Consortium (PBC) (Kasunic, 2006) was created in 2006 with the objective of providing tools and credible data for goal-setting and performance improvement and combining data from different provenances to create a superset of information for benchmarking and performance comparison. The benefits of the initiative would be to establish the

specifications for the collection and comparison of data of different source vendors and provide existing data to organisations to help them establishing and achieving their business goals. PBC members would contribute with their assets to a repository, and PBC would specify the measurements, in order to make the members data comparable. The subscriber organisations would be able to access the repository, have access to performance reports and submit performance data adherent to the measurements specifications. A large database was built with performance data of several organisations, but the analysis of the data allowed to conclude that they were not comparable because the organisations had their own definitions of each measure. They tried to use the TL9000 standard to define metrics but the standard has too many metrics, which would not be acceptable in the software industry [Mike Philips, 2010 personal communication].

From the results of the SEI experiments on creating the PBC and running pilot projects of SCAMPI appraisals oriented to results (see subsection 3.1.4 SCAMPI Limitations) it becomes clear that it is necessary to create a standard of software engineering metrics.

The Software Productivity Research (SPR) is a consulting services provider, created by Capers Jones. Capers Jones analyses data related to software processes performance, gathered by several organisations in USA. In his work he classifies software development projects in categories and supports the choice of the adequate quality decisions according with the characteristics of the projects, based on data (Jones, 2010).

The Software Benchmarking Organisation (SBO) is using the data of Capers Jones (2008) to apply benchmarking in the comparison of the behaviour of European projects with USA data. The results showed that projects of European organisations behave similarly to the USA's projects. Sassenburg and Voinea (2010) identified Key Performance Indicators (KPI) that would:

- Support project management in analysing, planning and monitoring projects;
- Inform top management of the status of the project and the direction that it is heading;
- Support business units in measuring their capability improvements;
- Support organisations in comparing/benchmarking business units.

They have a set of questions that allow to identifying KPI of different categories: project performance, process efficiency, product scope and product quality. We summarise them in table 3.2:

SBO's results indicate that it is possible to apply benchmarking techniques to the processes performance data and organisations data becomes comparable by these means. The benchmarking structure that is applied in SBO is done in three phases, each one of them involving a certain number of processes that are characterised by process definition elements (Sassenburg, 2009). The description of the method to perform benchmarking in organisations does not correspond to the expectations when asking for the benchmarking process. One of the outcomes of benchmarking is the definition of a process that is applicable to any objects that we intend to compare. In this particular case, the objects would be organisations.

Table 3.2: KPI categories - based on (Sassenburg and Voinea, 2010)

Question	KPI Category	KPI
How predictable is the project?	Project Performance	Cost, schedule, staffing rate, productivity.
How fast is my process?	Project Efficiency	Effort distribution (cost of the quality model).
How much of the product?	Product Scope	Features, deferral rate, size, re-use.
How well are we doing?	Product Quality	Complexity, test coverage, removal efficiency, defect density.

Evaluate Factors of Success

From a literature review, Jeffery and Berry (1993) extended a framework to evaluate and compare reasons for the success and failure of metrics programs. They analysed several authors' recommendations for the success of metrics programs and used Fenton's categories, namely context, inputs, process and products, to classify them. Based on that, they built a questionnaire to conduct a case study to analyse organisations in those perspectives. They provided the organisation's context and goals for the metrics program. The framework also includes a scoring scheme, to classify the extent to which the criteria were met. Later on, Wilson et al. (2001) adapted the Jeffery and Mike framework to be used in SPI.

Niazi et al. (2005) created a maturity model to assess and improve the implementation process of SPI. They related critical factors with maturity stage based on their occurrence in the literature and considering the inputs from interviews that they conducted. Those factors are related to the way SPI is conducted and not to improving processes outputs.

Metrics Repositories

Palza et al. (2003) designed an object-oriented model named Multidimensional Measurement Repository (MMR) to collect, store, analyse and report measurement data in order to facilitate the implementation of CMMI. MMR is based on PSM and the Software Measurement Process (ISO 15939).

The Alarcos research group proposed a measurement model (García et al., 2006) and an ontology for software measurement (Canfora et al., 2006). In 2007 they presented a proposal to support consistent and integrated measurement of software by providing the following elements: the Generic measurement metamodel, to represent the data related to the measurement process and the GenMETRIC, a tool that allows the specification of software measurement (García et al., 2007).

The Quantitative Approaches on Software Engineering and Reengineering (QUASAR) research group worked on the unambiguous definition of metrics. They also applied metamodel based approaches, in particular by using Ontology Driven Measurement (ODM), as defined by Goulão (2008). This model was an evolution of the MetaModel Driven Measurement (M2DM) for the evaluation of object-oriented designs of software engineering metrics (Abreu, 2001). In 2009

the ODM method was being applied to SQL databases in a Master Science thesis supervised by Goulão. Metrics metamodels can be used in the unambiguous definition of our framework's performance indicators.

Software Process Measurement

The Alarcos research group applied a metamodel approach to the management of software process performance measurement. They developed an integrated framework to model and measure the software processes based on number of activities, steps, dependencies between activities of the process, activity coupling, number of work products, number of process roles, and so on (García et al., 2003). In our research, process measurements are not restricted to the process itself but include the performance (efficiency and effectiveness) of its outcomes.

The Integrated Software Acquisition Metrics (ISAM) is an SEI project that consists in developing a common measurement framework for acquirers and developers based on TSP and PSP practices. Our framework not only measures CMMI practices but also evaluates the quality of their implementation.

Process Modelling or Simulation

Hsueh et al. (2008) showed that UML and software simulation can be used in the design, verification and validation of processes. They designed a static process metamodel that establishes the relations between the elements of CMMI and process components. Figure 3.5 presents the metamodel of CMMI. In their work, static processes are modelled with class diagrams and include the relationships between process elements and processes in CMMI. Process elements behaviour is modelled using state-chart diagrams. Finally, dynamic processes sequences are modelled with activity diagrams. The process verification is done by the definition of process rules and CMMI verification rules, using the Object Constraint Language.

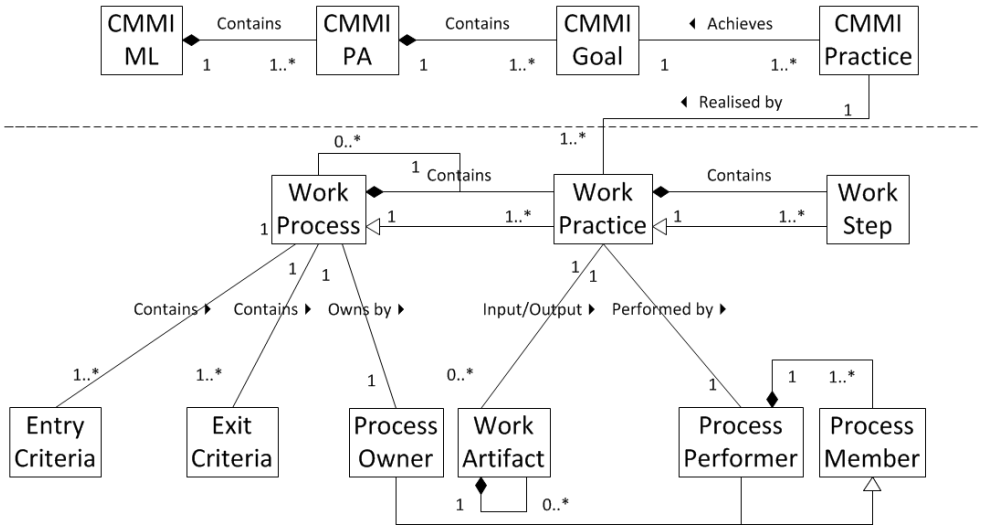


Figure 3.5: CMMI static process metamodel (Hsueh et al., 2008)

Mishra and Schlingloff (2008) proposed a formal specification based product development model that integrates product and process quality in the implementation of processes. They demonstrated the model in process compliance with CMMI, without considering performance.

Process Modelling and Measurement

Colombo et al. (2008) designed a metamodel to support multi-project process measurement to calculate across-process-multi-projects metrics aligned with CMMI. They developed an open source tool named Spago4Q that supports different development models, such as waterfall and agile. The CMMI assessment framework is supported by the Assessment component of Spago4Q. The tool is available online and is announced as being "a platform to measure, analyse and monitor quality of processes, products and services". Such tool may be useful to support the implementation of our framework.

In table 3.3 we summarise the existent frameworks and give some comments about them. The authors are identified by their last name initial or research group.

Table 3.3: Related Frameworks

Framework Type	Description	Ref.	Comments
Metrics Definition	Standard of code size metric that allows automation.	CISQ	Useful to have common and unambiguous metrics definition. To be released a productivity metric that considers quality
	Metrics for CMMI Process Areas.	Anon	Too many metrics unrelated to goals.
Metrics Analysis	Performance Benchmark Consortium.	K	Not comparable metrics due to different definitions.
	Publication of projects data.	J	Too many metrics unrelated to goals.
	Using projects data for benchmarking KPI.	SV	KPI that can be used to evaluate practices at higher level and characterise organisations performance.
Evaluate Factors of Success	Based on questionnaire and score. Used to evaluate: - Metrics Programs; - Software Process Improvements.	JB WHB NWZ	Based on how programs (SPI, Metrics) are conducted.

Continued on next page

Table 3.3 – *Continued from previous page*

Framework Type	Description	Ref.	Comments
Metrics Repository	Data model of software development.	KKJC	Model for a metrics database considering context.
	Multidimensional Measurement Repository. Metamodel design for software engineering metrics. Ontology for measurement and a measurement tool.	PFA GBC CGPRV GCLRP G	Formal specification of metrics and building repositories.
Software Process Measurement	Model and measure software processes. Measurement framework for TSP and PSP practices.	GRPC SEI	Based on process structure characteristics.
Process Modelling or Simulation	Models/metamodels of CMMI based on UML.	HSYY MS	Focused on compliance.
Process Modelling and Measurement	Metamodel to support multi-project process measurement aligned with CMMI.	CDFORR	Support CMMI assessment, not particularly focused on the quality implementation of the practices.
SCAMPI Modelling	Graph representation of the SCAMPI and quantitative evaluation. Repository for distributed projects to collect evidences as they are being executed.	PH SNG	Focused on compliance.
Performance Measurement	Performance measurement on SCAMPI for CMMI organisations benchmarking.	M	Introduced overhead on SCAMPI.

The frameworks mentioned so far can tackle some of the issues regarding the identified problem. However, they are not directly tackling the performance of process improvements. Also, one of the limitations verified in software engineering is there are several data repositories fed by different organisations, but not all are reliable as organisations should provide the same metrics under the same definition, collected systematically. The AFP standard allows organisations to automate their data collection on code size, if organisations use it and make their data available in public repositories will be a excellent source for this area of research. In the mean time, organisations using TSP already collect data using a similar measurement protocol and metrics definitions, as

TSP includes a set of forms to collect relevant metrics that are used to evaluate the quality of the process. Therefore, we validated our framework using TSP data (see details in section 5.1 Evaluation of the Estimation Process).

3.2 Survey on MA Performance in HML Organisations

What can organisations do to ensure that they properly achieve HML? This question is indirectly answered in the reports of two surveys conducted by the SEI: TR2008 (Goldenson et al., 2008) and TR2010 (McCurley and Goldenson, 2010). The surveys, regarding the use and effects of measurement and analysis in HML organisations, were focused on the value added by PPM and the results were considered comparable. In 2008 the respondents were the sponsors (assisted their delegates), or their delegates, from organisations appraised at CMMI HMLs, and in 2009 similar questions were asked to lead appraisers of organisations pursuing HMLs.

The SEI analysis is relevant to organisations pursuing any ML of CMMI, including problems and good practices that may have helped the organisations to achieve HML. In one question organisations had to indicate their routine while using PPM. The 2009 respondents indicated their most common problem was the long time it takes to accumulate historical data, which some organisations remediated by doing real time sampling of processes when they had no prior data available. In both surveys, the respondents gave different importance to obstacles found in the implementation of PPM (Figure 3.6).

To measure the strength of the relationship between two variables and the accuracy of predicting the rank of the response the authors used the Goodman and Kruskal's gamma. In 2008 the gamma value between the quality of the managers training and their capability to understand PPM results was not very strong. However, that relation was stronger when the training was more formal. 80% of respondents considered that the builders and maintainers of PPM understood CMMI's definition of PPM and PPB very well or even better, but their perception of the circumstances under which PPM and PPB are useful was lower. The results improved in 2009, over 50% of appraisers considered that the builders and maintainers of PPM understood all concepts very well or extremely well.

Organisations reported difficulties in collecting data manually. Regarding the automated support for MA activities responses to the 2008 survey showed that the organisations used spreadsheets, automated data collection and management software and, less frequently, statistical packages, workflow automation or report preparation software. Automation was considered to have a moderately strong relation with the overall value of PPM. There was also a strong relationship between the overall value of PPM and the following variables:

- Models with emphasis on "healthy ingredients" (listed next), and models for purposes consistent with those ingredients;

- Diversity of models used to predict product quality and process performance;

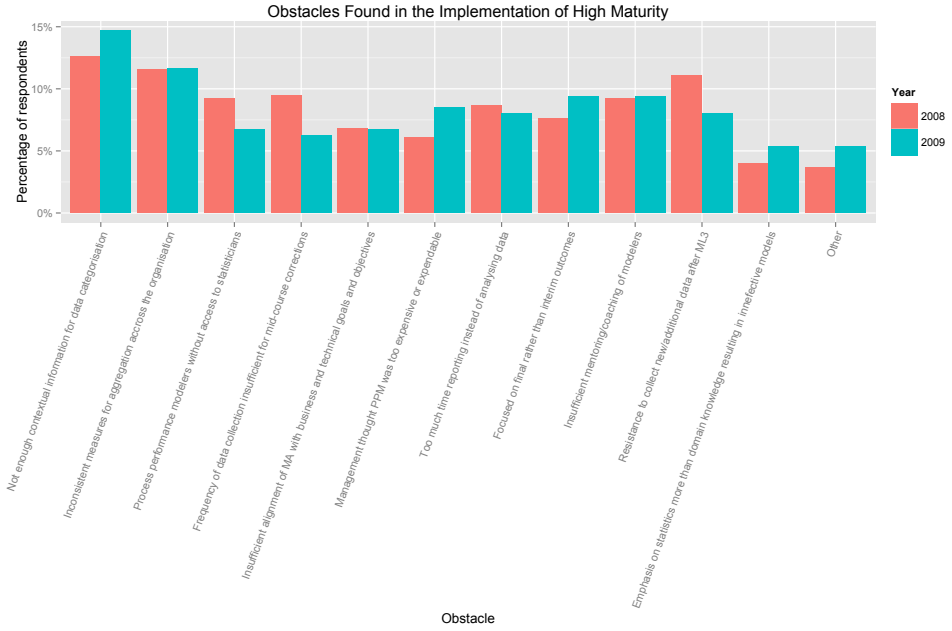


Figure 3.6: Obstacles identified by the organisations respondents found in the implementation of HML (TR2010).

- Use of statistical methods: regression analysis for prediction, analysis of variance, SPC charts, designs of experiments;
- Data quality and integrity checks;
- Use of simulation or optimisation methods: Monte Carlo simulation, discrete event simulation, Markov or Petri-net models, probabilistic models, neural networks, optimisation.

The SEI compiled a set of "healthy ingredients" to be considered in process performance modelling (Goldenson et al., 2008):

- Modelling uncertainty in the model's predictive factors;
- Ensuring models have controllable factors and possible non-controllable factors;
- Identify factors directly associated with sub-processes to construct the models;
- Predicting final and interim project outcomes;
- Using confidence intervals of the expected outcome to enable "what if" analysis;
- Enable identifying and implementing mid-course corrections during projects execution towards successful completion.

When analysing the relations between the achievement of HML and certain practices some revealed differences between achieving and not achieving HML:

- All organisations with poor or fair documentation relative to process performance and quality measurement results failed to achieve high maturity, whilst most of the organisations with excellent and good documentation achieved HML;

- Using simulation/optimisation techniques had a strong relation with HML achievement. The relation with the number of such methods used and achieving HML was very high. Particularly, all organisations using two of those methods achieved HML;

- There was a very strong relation between achieving HML and, respectively: having models with emphasis on healthy ingredients, and the models for purposes consistent with those ingredients;

- All organisations that used statistical techniques substantially, achieved HML;

- The frequency of using PPM predictions in status and milestones reviews had a quite strong relation with the achievement of the target HML.

In general the gamma between variables was higher in the 2009 survey. McCurley and Goldenson (2010) justify the improvements from the 2008 to the 2009 surveys results: "There may be a trend over time" and/or "The perspectives of the sponsors or the appraisers are more accurate". The results of both surveys indicate several improvements that HML organisations may consider to get full advantage of having PPM in place, but also show the obstacles that organisations that intend to implement HML practices may find.

3.3 Defect Classification Taxonomies

To define and validate the improvement component of the framework we developed in our research, we conducted an experiment of improving the requirements review process, by introducing a defect type classification specific to requirements (details on 3.3 Defect Classification Taxonomies). The literature reviewed that supported us on the definition of the classification is discussed in this subsection.

In 2009, Chen and Huang performed an e-mail survey with several software projects, and presented the top 10 higher-severity problem factors affecting software maintainability, as summarised in table 3.4. The authors indicated the following causes of software defects:

- a significant percentage of defects is caused by incorrect specifications and translation of requirements, or incomplete ones (Apfelbaum and Doyle, 1997; Monkevich, 1999);

- half of the problems rooted in requirements are due to ambiguous, poorly written, unclear and incorrect requirements, the other half result of omitted requirements (Mogyorodi, 2001).

Table 3.4: Top 10 Higher-severity problem factors impacting software maintainability (Chen and Huang, 2009).

#	Software Development Factors	Problem Dimension
1	Inadequacy of source code comments	Programming Quality
2	Documentation obscure/untrustworthy	Documentation Quality
3	Changes not adequately documented	Documentation Quality
4	Lack of traceability	Documentation Quality
5	Lack of adherence to standards	Programming Quality
6	Lack of integrity/consistency	Documentation Quality
7	Continually changing requirements	System Requirements
8	Frequent turnover within the project team	Personnel Resources
9	Improper usage of techniques	Programming Quality
10	Lack of consideration for software quality requirements	System Requirements

In 2003, Lutz and Mikulski analysed the impact and causes of requirements defects discovered in the testing phase, resulting from non documented changes or defects in the requirements, and proposed guidelines to distinguish and respond to each situation. Their work emphasises the importance of requirements management. Considering the problems that occur in the requirements specifications we present next, work that is related with or includes a requirements defects classification.

Code Defects Classifications, 1992

ODC is applicable in all the development phases except the requirements phase. The defect types used are: function, interface, checking, assignment, timing/serialisation, build/package/merge, documentation and algorithm. For each defect it is necessary to indicate if the feature is incorrect or missing (Chillarege et al., 1992). Such classifiers do not seem completely adequate to classify requirements defects, and Documentation is too generic to give further information on the defect. Hewlett-Packard (HP) (Grady, 1992) categorises the defects by mode, type and origin, (see figure 3.7). From the types of defects with origin in the requirements specification phase, the requirements specifications seem to be vague and the interfaces ones are too detailed and more adequate to design specification defects.

Quality Based Classifiers, 1976 – 2010

In 1976, Bell and Thayer conducted a research to verify the impact of defects in software requirements. Not surprisingly, they concluded that software systems meeting defective requirements will not effectively solve basic needs. They aggregated the defects in categories, as presented on table in figure 3.8. In 1981, Basili and Weiss categorised defects found in requirements documents and gathered a set of questions to be asked while reviewing them (as a review checklist). The table in figure 3.8 shows the distribution of the 79 errors by different categories. Later, in 1989, Ackerman et al. analysed the effectiveness of software inspections as a verification

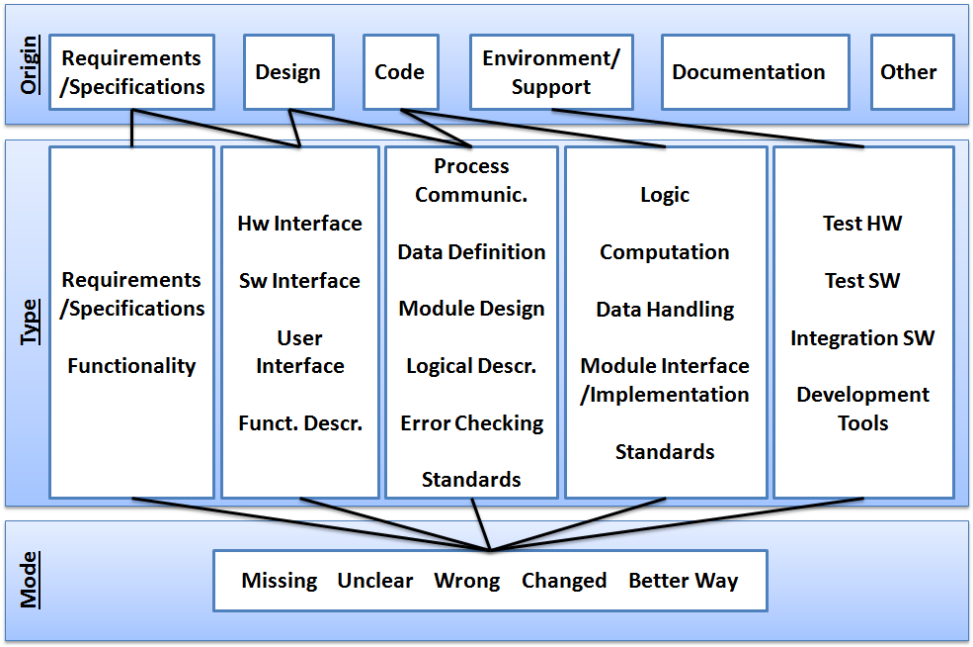


Figure 3.7: HP defects classification scheme (Freimut et al., 2005)

process. They presented a sample requirements checklist to use in inspections of requirements documents, containing questions organised by defect categories: completeness, consistency and ambiguity. And in 1991, Sakthivel performed a survey about requirement verification techniques and presented a requirements defects taxonomy based on a literature review (Walia and Carver, 2007). The classes that the author proposed are: incomplete, inconsistent, infeasible, untestable, redundant and incorrect. For each class, Sakthivel presented different defects and an example.

Hayes (2003), developed a requirements fault taxonomy for NASA's critical/catastrophic high-risk systems. Hayes stated that ODC refers to design and code while their approach emphasised requirements, so they adapted the Nuclear Regulatory Commission (NRC) requirement fault taxonomy from NUREG/CR-6316 (1995). Afterwards, in 2006, Hayes et al. analysed a software product related with the previous to build a common cause tree. In both works unachievable was reserved for future. In 2006, the same was also done with infeasible and non verifiable (table in figure 3.8 shows their results).

Defects classification is important to support the analysis of the root causes of defects. In 2010, Kalinowski et al. were aware that Defect Causal Analysis (DCA) could reduce defect rates by over 50%, reducing rework, and improving quality and performance. To enhance DCA, they improved their framework named Defect Prevention Based Process Improvement (DPPI) used to conduct, measure and control DCA. The authors mentioned the necessity of collecting metrics for DCA and the importance of considering:

1. Context when collecting metrics;

2. Stability of the inspection;

		Bell&Thayer	Basili&Weiss	Ackerman'al	Sakthivel	Chillarege'al	Grady	Schneider'al	Porter'al	Hayes'al	Walia&Carver	Kalinowski'al	Sum
1	Not in current baseline	1.50%											1
2	Out of scope	7.20%											1
3	Missing/Omission	21.00%	24.00%							10.80%		23.50%	4
4	Incomplete	merged		Yes	Yes					23.30%			4
5	Inadequate	merged											1
6	Incorrect	34.80%	37.00%		Yes					30.11%		35.30%	5
7	Inconsistent	9.10%	10.00%	Yes	Yes			23	Yes	13.07%	Yes	5.90%	9
8	Incompatible	merged											1
9	New	7.20%											1
10	Changed Requirement	merged											1
11	Typos/Clerical	9.90%	23.00%										2
12	Unclear	9.30%											1
13	Ambiguity		4.00%	Yes				15	Yes	13.07%	Yes	11.80%	7
14	Wrong Section/Misplaced		1.00%						Yes	1.14%	Yes		4
15	Other		1.00%								Yes	5.90%	3
16	Infeasible				Yes					0.00%			2
17	Untestable/Non-verifiable				Yes					0.00%			2
18	Redundant/Duplicate				Yes					2.27%			3
19	Missing Functionality/Feature						/u/w/c/b	34	Yes		Yes		4
20	Missing Interface					/incorrect		11	Yes		Yes		4
21	Missing Performance							7	Yes		Yes		3
22	Missing Environment							9	Yes		Yes		3
23	Missing Software Interface						/u/w/c/b						1
24	Missing Hardware Interface						/u/w/c/b						1
25	Missing User Interface						/u/w/c/b						1
26	Missing Function/Description					/incorrect	/u/w/c/b						2
27	Missing Requirement/Specification						Inadequate						0
28	Missing/Incorrect Checking					Yes							1
29	Missing/Incorrect Assignment					Yes							1
30	Missing/Incorrect Timing/Serialization					Inadequate							0
31	Missing/Incorrect Build/Package/Merge					Inadequate							0
32	Missing/Incorrect Documentation					Inadequate							0
33	Missing/Incorrect Algorithm					Formal Spec							0
34	Incorrect or Extra Functionality								Yes		Yes		2
35	Data Type Consistency								Yes				1
36	Over-specification									1.14%			1
37	Not Traceable									2.27%			1
38	Unachievable									0.57%			1
39	Intentional Deviation									2.27%			1
40	General										Yes		1
41	Extraneous Information											17.60%	

For each defect classifier we indicate the authors who used it. The following information appears: **Yes** if we have no further information; the percentage of occurrence of a defect using the data of the experiment done with more data points; the quantity of defects; merged when the author used it merged with the classifier that is above that one; Inadequate when we consider that the classifier is not useful for requirements defects; /incorrect, indicating that the authors also used the 'incorrect' prefix; /u/w/c/b indicating the authors also used the prefixes 'Unclear', 'Wrong', 'Changed' and 'Better Way; Formal Spec. (Formal Specification) when we consider that such defect classifier would only be applicable if the requirements were specified with formal language.

Figure 3.8: Defect classifier per authors by chronological order from left to right.

3. Technology/similarity of projects in inspections.

When demonstrating their approach they reported the requirements defects distribution, classified by nature (see on figure 3.8).

Functional and Quality Based Classifiers, 1992 – 2009

Next we present defect classification taxonomies that are functional and quality based. In our research we consider that the functional classifiers represent the function of the requirement in the product (e.g. interface, performance, environment, functional).

Schneider et al. (1992), identified two classes of requirements defects to use when reviewing user requirements documents: Missing Information and Wrong Information(figure 3.8). In 1995, Porter et al. compared requirements inspection methods. They performed an experiment where two Software Requirements Specification (SRS) documents were inspected with a combination of *ad hoc*, checklist and scenario inspection methods. The checklist was organised in

categories, resembling a defect classification: omission (missing functionality, performance, environment or interface) and commission (ambiguous or inconsistent information, incorrect or extra functionality, wrong section). The scenarios also included categories: data type consistency, incorrect functionality, ambiguity, and missing functionality. The authors concluded from their results that the scenario inspection method was the most effective for requirements.

Later, in 2007, Walia and Carver repeated an experiment to show the importance of requirements defects taxonomy. They involved software engineering students in a SRS document review using a defect checklist. The students repeated the review, after being trained in the error abstraction process. The results of the experiment showed that error abstraction leads to more defects found without losses of efficiency and the abstraction is harder when people are not involved in the elaboration of the SRS and have no contact with developers. Requirements defects were classified as: general, missing functionality, missing performance, missing interface, missing environment, ambiguous information, inconsistent information, incorrect or extra functionality, wrong section, other faults. This experiment was applied to error abstraction; we consider that a similar experiment is useful to validate defects classification.

Along the years researchers introduced classifiers to fulfil the specificities of requirements defects. Some reused existent classifications and conducted experiments to analyse the impact of different methodologies in SRS inspections. The table in figure 3.8 summarises the relation between authors and classifiers.

3.4 Related Research on Effort Estimation

We did a literature review, designed to gather information to answer the following research questions:

- Which effort estimation methods exist?
- How can we define the effort estimation accuracy?
- Which factors are considered on effort estimation?
- Which factors affect effort estimation accuracy?

Part of our strategy was to analyse errors in schedule, effort and duration of projects and find the causes of those deviations, already identified by other researchers.

Jørgensen and Shepperd (2007), defines estimation approach to name the method used to estimate, which includes regression, analogy, expert judgement, work break-down, function points, classification and regression trees, simulation, neural networks, theory, Bayesian and combination of estimates. We found several effort estimation methods that other researchers classified. We merged overlapping classifications: expert based (Moløkken and Jørgensen, 2004), expert judgement/expert estimation (Lopez-Martin, 2011) and Knowledge-based (Jun and Lee, 2001); Model

Based (Moløkken and Jørgensen, 2004), Statistical Model (Jun and Lee, 2001) and Algorithmic Model (Lopez-Martin, 2011); Artificial Intelligence (AI) (Jun and Lee, 2001) and Machine Learning (Lopez-Martin, 2011). The following classifications were considered:

- **Expert Based/Expert judgement/Expert Estimation/Knowledge-Based** – intuitive processes that aimed at deriving estimates based on the experience of experts on similar projects, expert consultation (Lopez-Martin, 2011). Include Intuition and experience, and Analogy by comparing completed similar tasks (Moløkken and Jørgensen, 2004);

- **Model Based/Statistical Model** – are software cost models, including formal estimation models and algorithm driven methods (Moløkken and Jørgensen, 2004); based on mathematical functions between causing factors and resulting efforts, where the estimating parameters are based on historical data (Jun and Lee, 2001), and linear and non-linear regression (Lopez-Martin, 2011). For example, COCOMO, Use-Case based, FPA metrics, Putnam's SLIM, Doty, TRW, Bailey&Basili;

- **Artificial Intelligence/Machine learning** – includes fuzzy logic models, neural networks, genetic programming, regression trees and case-based reasoning (Jun and Lee, 2001; Lopez-Martin, 2011);

- **Parametric models** – use historical data (system size, complexity, skills and experience of the project personnel, hardware limitations, software development tools, stability of user requirements, and reuse of SW routines) (Morgenshtern et al., 2007). An example is Function Points estimation that uses the number of entities and their complexity to determine size;

- **Other** – Price-to-win, Capacity Related, Top-down, Bottom-up, Other (Moløkken and Jørgensen, 2004). Although, the authors consider that Top-down and Bottom-up can also be interpreted as expert judgement methods.

We compile the summary of methods we found on table A.1 and how they were classified, on Appendix A. From all the methods in use, the one that seems to have more accurate results is fuzzy logic, even better than particle swarm optimisation (Morgenshtern et al., 2007). These methods are good to gather estimates but explaining and varying the factors can be more complex when compared to regression models with variables without transformations. People experience should not be neglected, in particular because in some situations expert estimates can be expected to be more accurate than formal estimation models (Morgenshtern et al., 2007). Even TSP recommendeds teams to use expert judgement when there is no prior TSP data available and the guidelines are not applicable to the project (Humphrey, 2006).

We also analysed the methods used to evaluate and compare the accuracy of the estimation methods. We present their equations on the next paragraphs.

$$MeanMagnitudeRelativeError: \quad MMRE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Est_i - Act_i}{Act_i}\right| \tag{3.1}$$

In 3.1 MMRE is the Mean MRE, in a given project i, where n is the number of samples, Est is the estimated and Act is the actual. The metric is better when lower. The MRE penalises overestimation more than underestimation (Jørgensen, 2004). The Relative Error RE is the fraction of 3.1, without the module, showing the direction of the estimate.

$$Percentage of Predictors(r): \quad PRED = \frac{k}{n} \tag{3.2}$$

In the PRED(r) equation (3.2) k is the number of projects in a set of n that whose MRE <= r, that is, fall within r of the actual value. A cost model is considered accurate when MMRE is at most 0.25 and PRED(25) is at least 75% (Braga et al., 2008).

Jørgensen (2007) used 3.3 to determine estimation accuracy as follows:

$$EstimationAccuracy: \quad z = \frac{Est}{Act} \tag{3.3}$$

$$MagnitudeofErrorRelative: \quad MER_i = \frac{|Act_i - Est_i|}{Est_i} \tag{3.4}$$

MER (3.4) is calculated for each observation i whose effort is predicted, and its aggregation over multiple observations gives the Mean MER (MMER) (Smith et al., 2001; Lopez-Martin, 2011; Hari and Prasad Reddy, 2011).

Other equations referenced by Hari and Prasad Reddy (2011) are the Variance Accounted-For, equation 3.5, Mean Absolute Relative Error, equation 3.6, and Variance Absolute Relative Error, equation 3.7.

$$VarianceAccountedFor: \quad VAR = 1 - \frac{var(Act - Est)}{var(Act)} \tag{3.5}$$

$$MeanAbsoluteRelativeError: \quad MARE = mean\frac{abs(Act - Est)}{Act} \tag{3.6}$$

$$VarianceAbsoluteRelativeError: \quad VARE = var\frac{abs(Act - Est)}{(Act)} \tag{3.7}$$

Statistic analysis of the models residuals was also used by Grimstad and Jorgensen (2006), including the Standard Deviation of residual error (3.8).

$$StandardDeviation: \quad SD = \sum \frac{(Act - Est)^2}{n-1} \tag{3.8}$$

From the analysis of all the variables used by other researchers, when analysing effort estimation accuracy, we found that statistic methods would be more adequate, however other authors compared them to MMRE and PRED concluding that they did not lead to a significant improvement (Jørgensen, 2004). From the variables more commonly used in effort estimation MMER is preferable to MMRE, because MER measures the inaccuracy relative to the estimate (Foss et al., 2003). Therefore we used it in our model to evaluate the quality of implementation of the practice "Estimate Effort" (see 5.1 Evaluation of the Estimation Process).

We also analysed the factors influencing the effort estimation accuracy, grouped as factors to be considered on the:

- **Estimation Process** - including experience and skills of estimators and executors; complexity of products, processes, tools; schedule, time constraints; level of detail; uncertainty of project, technology; personnel availability, turnover; support tools; people localisation; cumulative complexity; methodology, development phases; customer involvement; internal and external communication;
- **Project Execution** - including project execution, priorities, information completeness; personnel availability and turnover; progress and status control; risk management; reporting.

We include the tables detailing the factors of the effort estimation process and the factors of the development process on appendix A. The raised factor were considered when building the data model and dictionary (4.3.1 Data Model). This literature reviewed served as input to base our research when designing the experiment, analysed data and implemented the Effort Estimation Accuracy model (5.1 Evaluation of the Estimation Process).

Chapter 4

The EQualPI Framework

In this chapter we present the EQualPI Framework (Framework to Evaluate the Quality of Implementation of Process Improvements). To design its metamodel we based ourselves in the SPEM 2, CMMI architecture, measurement principles and the necessary alignment that must exist between the organisation (quantitative) business goals and the performance indicators. EQualPI is composed by a **Metamodel** that defines the rules and relations between its elements, a **Repository** that includes the mathematical models and data to perform a quantitative evaluation of the implemented practices and a set of **Procedures**, including the steps to prepare EQualPI to be used in an organisation.

4.1 Framework Overview

When organisations use CMMI their processes are aligned with the process areas they use and they implement those practices using operational practices, of which TSP is an example Phillips (2010). In our research we call **methods** to the definitions of the operational practices. We evaluate quality of implementation by determining the degree of support of the methods used by the organisation in the definition of their processes to the CMMI practices and use performance indicators to both measure the quality of implementation and the organisation performance. In this context we define the following concepts:

- **Method** - good practices, procedures, techniques, etc., that define how the work is done and support doing it. Methods are used to achieve a certain work objective.
- **Process** - includes a set of reusable elements, the methods, which can be optional, alternative or mandatory, and are used to perform work. The processes are tailorable, meaning that some methods or activities are optional and others can have alternative ones, decided according with the implied needs of the work to perform.
- **Quality of Implementation** (of a practice) - refers to the way that the work related to a practice is performed. One way of working is better than another if it consistently produces better results in a more effective way. We characterise the quality of implementation of a

CMMI practice by a combination of **efficiency** and **effectiveness** of implementation, on one hand, and **compliance** of implementation on the other (i.e., alignment with CMMI recommendations or with what is prescribed by the concrete implementation method used), all measured by appropriate **performance indicators** (PI), possibly dependent on the practice and implementation method used. By considering these three quality characteristics, we are looking both at **how** the work is done and **what** its performance results are.

- **Performance Indicators** - derived metrics that measure the organisation performance and/or the quality of implementation. Their aggregation indicates the degree of institutionalisation of the practices that is necessary to achieve generic goals and high maturity, and consequently to allow practices' evaluation.
- **Controllable factors** - methods, that are distinguishable, enumerable and reusable and in some cases quantifiable. They can be related to each other or not.

To build the EQualPI we followed a bottom-up approach, as represented in figure 4.1, based in the principle that the quality of the practices implemented in lower maturity levels is reflected on the practices of higher maturity levels. The practices used individually, in each team, project, organisation unit and in the organisation give the implementation quality as a whole. The result of the evaluation is signalled in a colour scheme (red, yellow and green), defined by the threshold of the performance indicator.

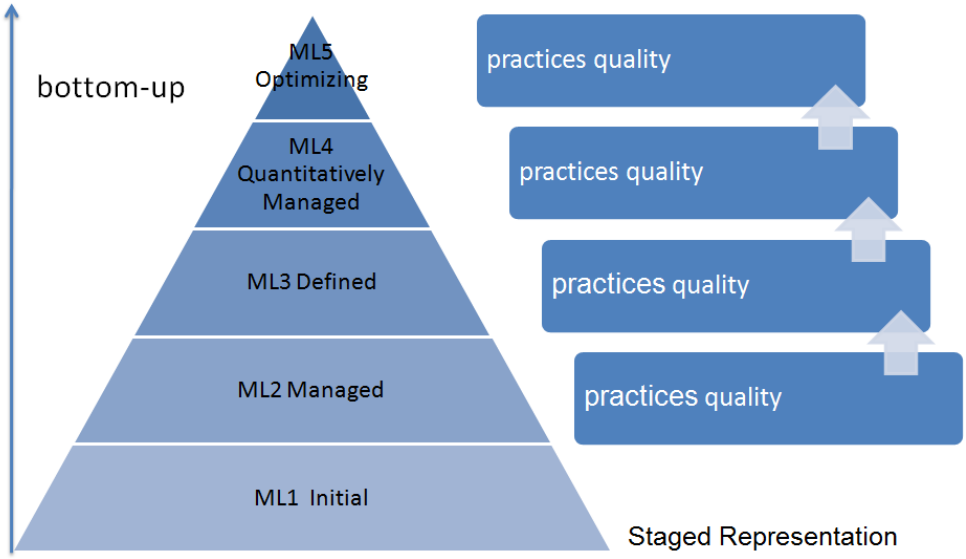


Figure 4.1: Bottom-up evaluation of practices implementation.

To be compliant with the CMMI model the organisation needs to implement the Specific Goals (SG) and Generic Goals (GG) goals enounced in the model, in the current maturity level and all the precedent ones (Chrissis et al., 2011). To satisfy a goal the Generic Practices and Specific Practices or acceptable alternatives to them need to be fulfilled.

Our approach to the problem is represented in figure 4.2. For each Specific Goal and/or Specific Practice we identified the methods that are documented in the literature and the ones that are used in the organisations to implement the CMMI goal or practice. The organisation's processes and tools are documented in the Quality Management System.

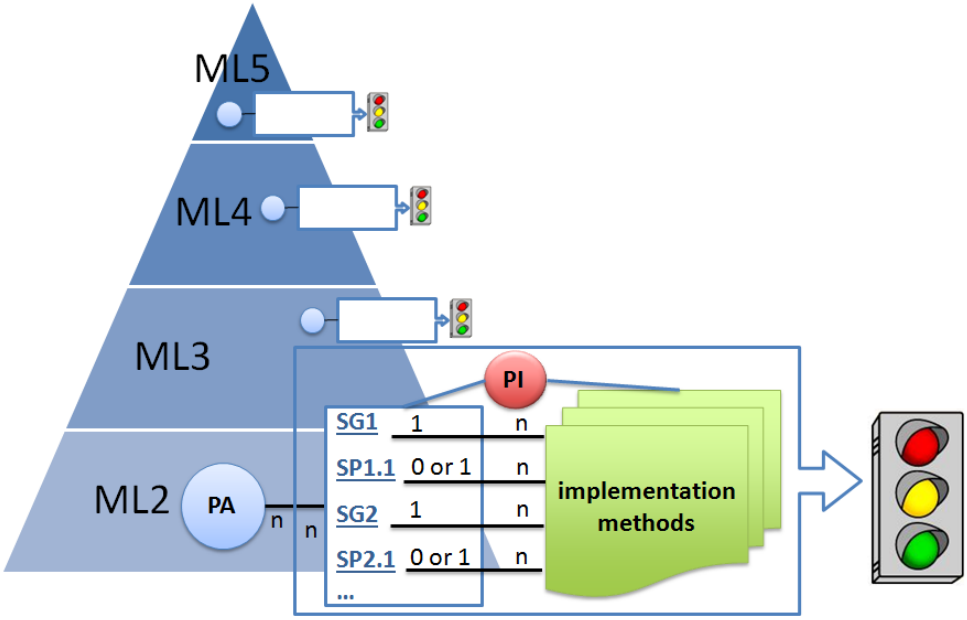


Figure 4.2: Building the evaluation framework (Lopes Margarido et al., 2011b). Legend: ML – Maturity Level, PA – Process Area, SG – Specific Goal, SP – Specific Practice, n – one or more, PI – performance indicator.

The practices implementation is reflected in the implementation of the generic goals. We map the methods with the CMMI practices and/or goals, and evaluate the quality of implementation. That quality is related to the percentage of the CMMI practice that the method covers and the way it is implemented. The SCAMPI evaluates if a practice is fully implemented, partially implemented or not implement and in certain cases the evidence of a practice is a documented process and records of its usage, and collateral evidences of its application (refer to 2.3 CMMI Architecture and Appraisal Method). The effectiveness of the methods application is not considered. To analyse the practice implementation we based the evaluation on the recommendations about the method that can be found in the literature.

The percentage of usage of a practice is reflected in the generic practices. For example, if 100% of the projects use the process, tailored from the organisation's set of standard processes, then the process is institutionalised as a standard process and "GG3 – Institutionalize a Defined Process" is fulfilled. We consider that the percentage of usage of a method in the organisation can be used to evaluate GG and higher maturity practices.

The purpose of the research is to develop a framework that allows organisations to select and adapt methods that improve the quality of implementation of CMMI practices, allowing them to

monitor practices performance and anticipate the impact of changing their processes in the performance of practices. Our goal is to help improve and replicate success by identifying the elements in the way of doing work (controllable factors) that have most impact in the efficacy and efficiency of the processes. Those elements are related to the methods used in the execution of the process. The framework is composed of a metamodel, shaping a repository of performance indicators, to evaluate the quality of implementation of CMMI practices, dependent on the methods used to implement those practices. The performance indicators are tailorable, defined as mandatory or optional, and mapped with profiles according to maturity level and the methods of the organisation. Additionally, the framework includes procedures for setup (tailoring), use in practice to evaluate quality of implementation and do process improvements, and supporting choice of indicators. However, CMMI has 22 Process Areas, with the respective Specific Goals and several Specific Practices, and maturity levels 2 and 3 have Generic Goals, with their corresponding Specific Goals. Developing a framework to include the entire model in a single Ph.D. research would be infeasible. Consequently, we decided to demonstrate our theory by analysing the factors that influence the quality of implementation of the CMMI Project Planning SP 1.4 "Estimate Effort and Cost".

We consider that the results of project estimation are determinant in projects' success. For example, with good effort estimation the uncertainty in the plan is lower, and the team is less stressed and subject to extra unconsidered tasks that should have been considered as part of the scope of the project, but forgotten. We also believe that it would be easier to determine how the framework can be extended by demonstrating it in this particular area because we think that PP SP1.4 depends on the quality of other CMMI practices. We focused on effort estimation and left cost out of the research because effort estimates are a dominant factor considered in cost estimation.

The quality of a project plan depends on the quality of estimates and some of the factors that influence those estimates can and need to be controlled. We focused our research on the factors that we can control, to do better Effort Estimation, i.e. in indicators that are drivers of the results. Those indicators are the ones related to the Effort Estimation process and they show how well the process is defined and executed. Other factors, in particular the ones related to the project execution itself were monitored with two purposes: understand the percentage of the effect that they have on the Effort Estimation Accuracy and to characterise our datasets, so that the research can be reproducible.

Our goal was to demonstrate that it is possible to evaluate the quality of implementation of the CMMI practices based on the performance resultant from the methods used in their implementation. Such evaluation shall support organisations to achieve the benefits of the model and anticipate the impact of changes in their processes.

We consider that a performance indicator that shows the quality of effort estimates is the Effort Estimation Accuracy, based on the deviation between the estimated and actual effort. The effort estimation deviation depends on several factors, such as:

- Definition of the estimation process;

- Execution of the estimation process;
- Execution of the project.

These factors contribute to the quality of implementation of the Effort Estimation Accuracy, also designated as construct or dependent variable. We determined the percentage of the estimation accuracy that is affected by the outputs of the estimation process – quality of the process and its execution – and focus on the analysis of factors that contribute to that percentage. Therefore other external factors related to the execution of the project and the team had to be controlled and recorded but are out of the scope of the research. In practice, we formulate the problem as follows: the dependent variable Effort Estimation Accuracy, Y, is a function of n controllable factors X_c and i non-controllable factors X_{nc} (see equation 4.1).

$$DependentVariableDefinition: \quad Y = f(X_{c1}, X_{c2}, ...X_{cn}, X_{nc1}, X_{nc2}, ...X_{nci}) \tag{4.1}$$

Regarding the performance indicators we understand that some of them depend on the implementation methods. However we wanted them to be good predictors of the performance of the organisation regarding **Effort Estimation Accuracy** and to be usable. We consider that usable performance indicators are the ones that bring added value and can be collected in a cost effective manner (without high costs and overhead). They allow the organisations to improve their performance by following the guidelines of implementation of the methods, knowing current and desired performance and knowing impact of process changes on performance indicators.

4.2 EQualPI Architecture

In this section we present the EQualPI Framwork complemented with technical description.

In figure 4.3 we present the architecture level 0, the deployment perspective, of the EQualPI framework. This perspective is used to give the physical context in which EQualPI will operate, within an organisation context. We also identify external agents that communicate with the framework.

In the company work environment and while people do their work, metrics are collected from **Workstations** to the **Company Database**, where the data on those metrics are stored. The EQualPI framework is composed of a **EQualPI Client**, that presents the User Interface (UI), which allows the user to access the **EQualPI Server** where evaluations are performed. The server includes the **Repository** and **Procedures**, both shaped by the **Metamodel**. The performance indicators to evaluate the quality of implementation of the CMMI practices are stored in the Repository. They may depend on the methods used to implement those practices and on the business goals. For that reason the performance indicators are tailorable, defined as mandatory or optional, and mapped with profiles according with maturity level and methods used by the organisation. Furthermore, organisations can add metrics that are not yet in the system, but they must be aligned

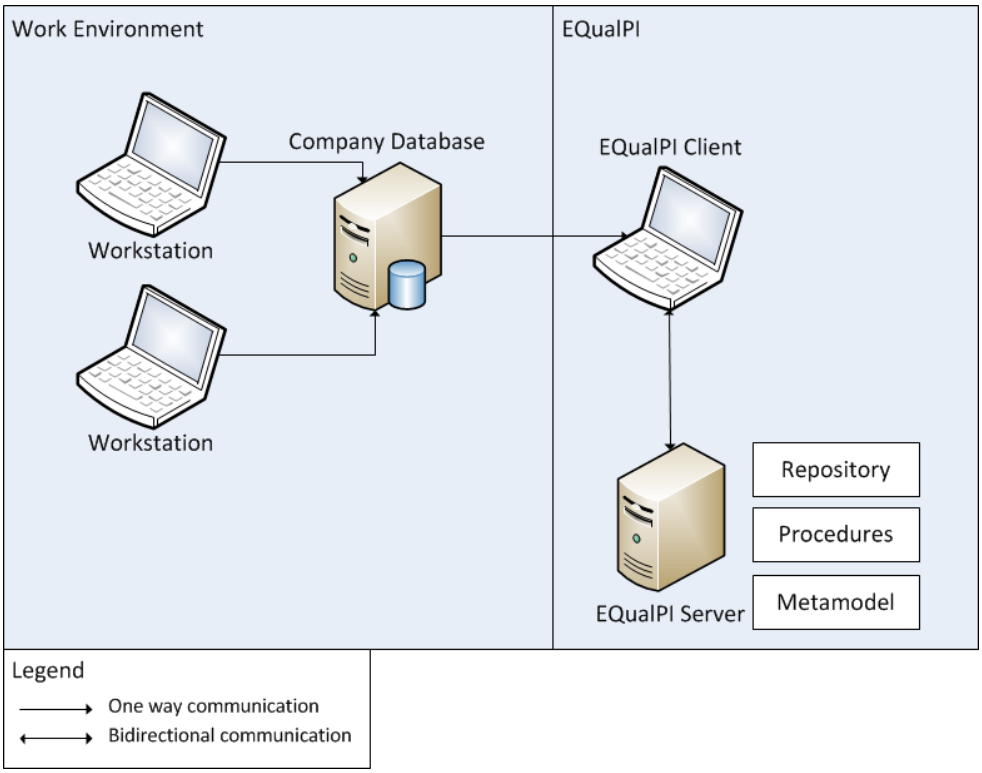


Figure 4.3: EQualPI architecture level 0 - deployment perspective.

with business goals, methods and CMMI practices and respect the data dictionary rules. Additionally, EQualPI includes procedures for setup (tailoring), use in practice and support the choice of indicators.

We used a metamodel to define our framework. According with Álvarez et al. (2001), in a model architecture a model at one layer specifies the models in the layer below and it is viewed as an instance of some model in the layer above: "The four layers are the meta-metamodel layer (M3), the metamodel layer (M2), the user model layer (M1) and the user object layer (M0)." In our case we represent the metamodel layer M2 and the user model layer M1. Figure 4.4 presents the level 1 of EQualPI's architecture, the static perspective, including the layers of the framework and the metamodel. This perspective represents the modules that are part of EQualPI, which is implemented in a three-tier architecture.

EQualPI has three layers: **Presentation**, **Business** and **Data**. In the Data Layer all data are stored, including the data of the organisation and the data that comes with EQualPI, belonging to the Repository. The Repository has the **Data Dictionary**, which describes all variables properties and all base and derived measures that allow to evaluate CMMI practices. The **Domain Model** explains the relation between variables that are specified in the Data Dictionary. Finally, the Repository also includes the **Performance Indicators Models** that are the ones that actually evaluate the quality of implementation of a given practice. The organisation data is composed of **Organisation Metrics**, **Evaluation**, which includes all evaluations, and **Organisation Settings**

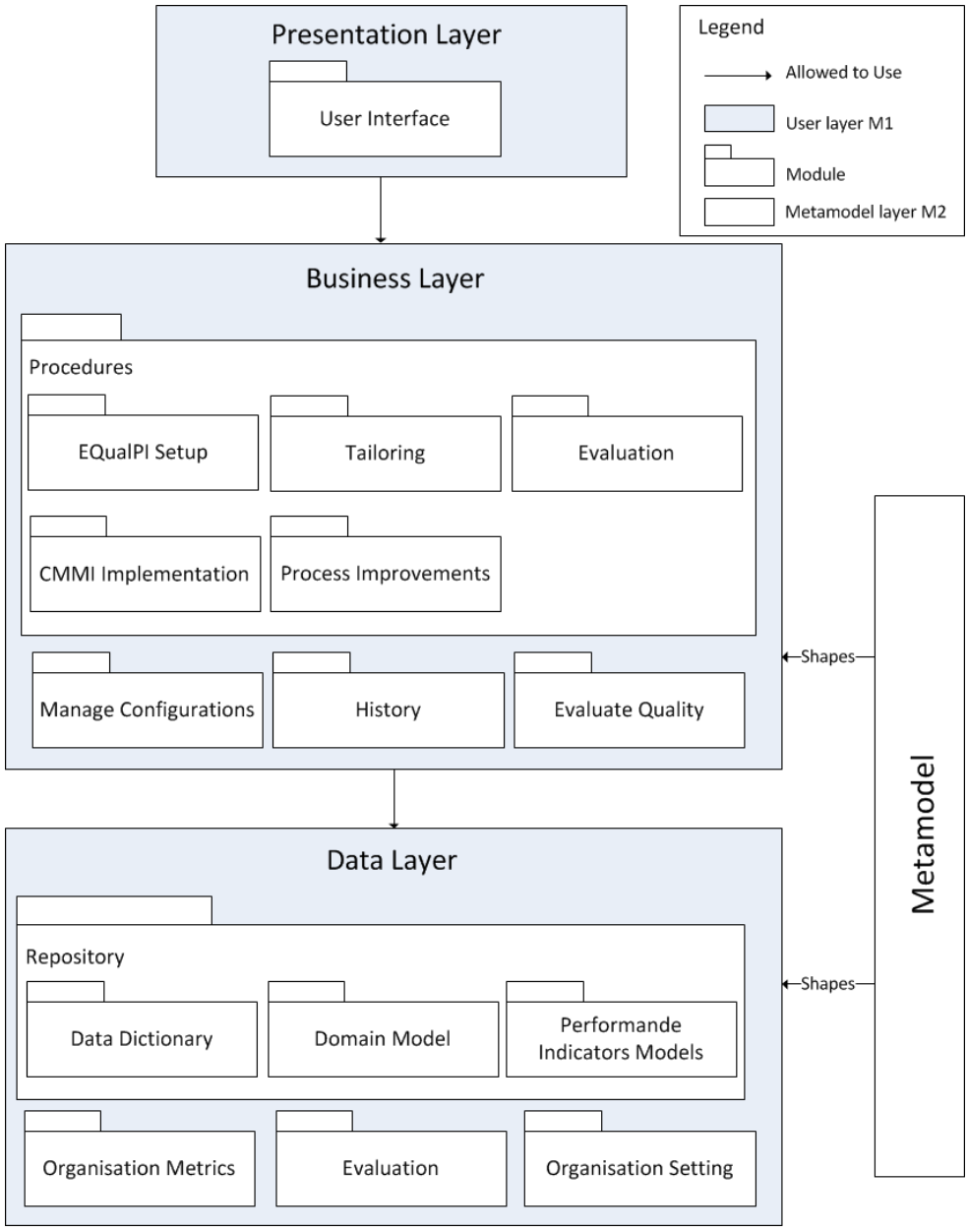


Figure 4.4: EQualPI architecture level 1 - static perspective.

that determine the practices and methods in used, mapped with the organisation goals. The settings history is also stored, so if an organisation needs to see an old evaluation it can also see the settings that were used by then.

The Business Layer includes all Procedures that are necessary to use EQualPI and include:

- **EQualPI Setup**, how to align business goals with performance indicators and practices. The information that needs to be provided to start using the framework.
- **Tailoring**, with instructions to select the practices, methods and performance indicators the organisation will use.

- **Evaluation**, that explains how the evaluation and aggregation is done.
- **CMMI Implementation**, that includes guidelines and a checklist to help implement CMMI avoiding problems that often occur when implementing the model.
- **Process Improvements**, the steps to do process improvements and how to use the framework to evaluate them.

The Business Layer allows to **Setup (the) Framework**, receiving the data of the organisation and preparing EQualPI to work with those data. It is possible to consult the **History** of the organisation, such as previous evaluations and settings, and actually evaluate the quality of implementation or of improvements using the **Evaluate Quality** module. The metamodel shapes the Business and Data Layers, as they follow the definitions it contains.

The Presentation Layer includes the **User Interface** that can only communicate with the **Business Layer**, the UI formats and presents the information to the end user. Through it the user sends the information that is necessary to setup the framework and the data of the organisation, formatted according with the data dictionary. To evaluate quality the user inserts the parameters necessary to do it; EQUalPI presents the results of the evaluation in the UI. The user can also send parameters to consult previous evaluations, whose results are presented in the UI as well.

A metamodel can support the design and implementation of a framework and establishes dependencies and rules necessary to use it in practice. In practice, it describes and analyses relations between concepts. We represent the metamodel of the repository in figure 4.5 and of the evaluation in figure 4.6.

Regarding 4.5, the organisation selects the constellation of the CMMI framework to be implemented, in the figure designated as **Reference Model**. According with the organisation goals the practices to be implemented are selected, which all together they may allow achieving a maturity level. The organisation goals are aligned with performance indicators and methods that support them and the CMMI practices that are being implemented. The **Performance Indicators** allow the evaluation of practices and are derived measures, calculated through **Base Measures** that the organisation collects through time. There are **Process PI** evaluating the process and **Product PI** that are related to the product. **Leading** indicators, helping to predict the outcome of the work done following a given method, and **Lagging** indicators, used to appraise the process and product implementation performance. A lagging indicator in a phase of the project lifecycle may be leading indicator in the next phase.

For instance, assume that we want to evaluate the quality of implementation of practice "SP2.2 Conduct Peer Reviews" of the Verification process area and that reviewing follows two TSP guidelines: use checklists derived from historical data, and review at a moderate pace. Here, one can measure **efficacy** by *review yield* (percentage of defects detected), **efficiency** by *defect detection rate* (defects detected per hour), and **compliance** by *checklist usage* (a qualitative PI with values, *not used*, *ad-hoc check-list*, and *checklist derived from historical data*) and *review rate* (size reviewed per hour), compared with some recommended values.

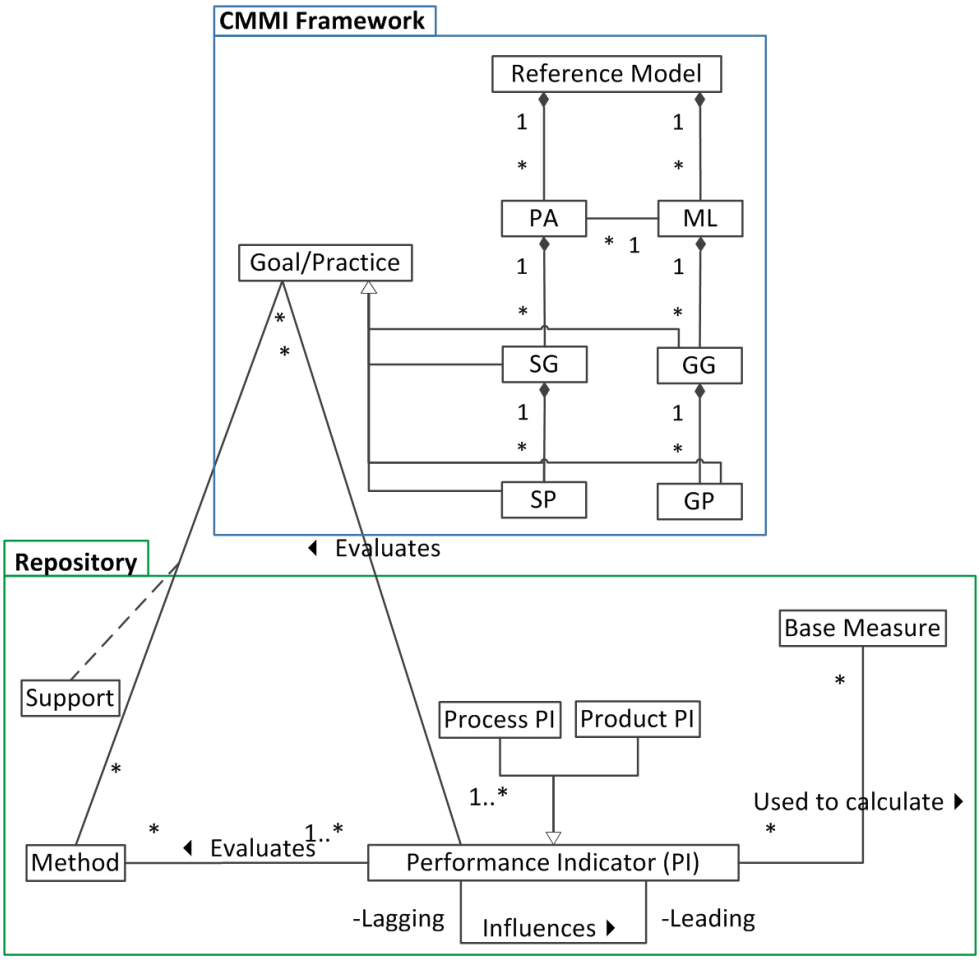


Figure 4.5: EQualPI repository metamodel. Legend: PA- Process Area, ML - Maturity Level, SG- Specific Goal, SP- Specific Practice, GG- Generic Goal, GP- Generic Practice, PI- Performance Indicator

A rich set of PI usually combines process and product indicators, and leading and lagging indicators. In the given example, *review yield* is a lagging performance indicator, as the remaining defects can only be known a posteriori. Compliance indicators are often leading indicators; they influence and can be used to predict and control the values of lagging indicators. In the example, *review rate* is commonly considered a leading indicator of the *review yield* in TSP literature. The *density of defects* found in a review is a product performance indicator, whilst the *review rate* is clearly a process performance indicator.

To conduct an evaluation (see figure 4.6) the **Goals/Practices** are mapped with one or more **Methods** that implement them and the **PIs** used to evaluate those methods and, consequently, the Practices. A PI has a **Threshold** that at a given **time** has a particular **value** and is established by the organisation from the analysis of what is considered to be the limit of the normal behaviour for that indicator. Thresholds have different levels, used to determine the PI **semaphore** colour (red, yellow, green), established according with the organisation quantitative business goals and

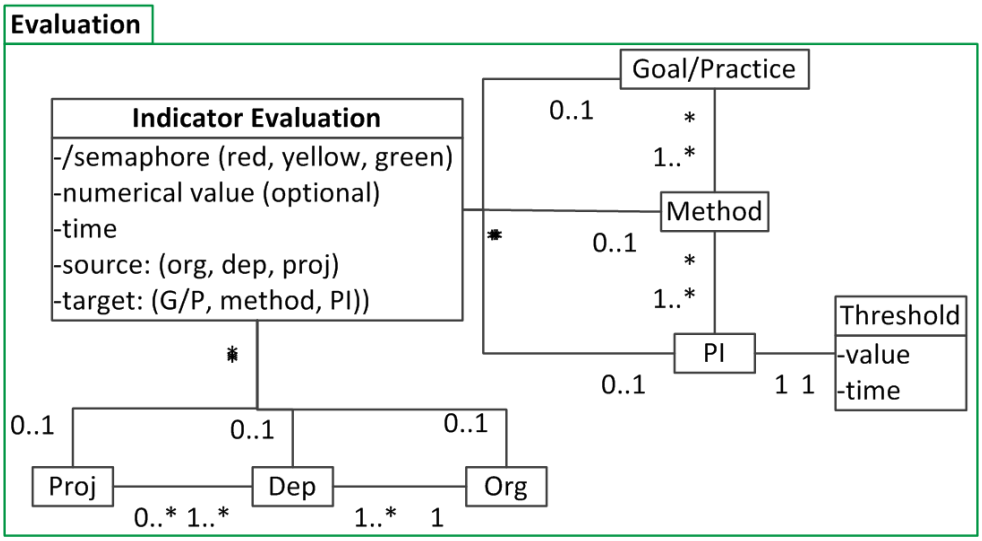


Figure 4.6: EQualPI repository metamodel. Legend: Proj- Project, Dep- Department, Org- Organisation, PI- Performance Indicator, G/P- Goal or Practice.

processes baselines, and define its normal behaviour regarding a PI. The evaluation of a method can be done by aggregating the result of the evaluation of one or more PIs; similarly the evaluation of a practice is given by the aggregation of the evaluation results of their implementing methods. Notice that a practice can be implemented by a group of methods, therefore, a method can be **mandatory**, **alternative** or **optional**.

The PI can be collected at a given **Source**, a **Project**, a **Department** or in the **Organisation**. Moreover, the evaluation of the several projects can be aggregated to evaluate a department, and the results of a department can be aggregated to evaluate the entire organisation. In any of these cases what is evaluated may be a PI, a method or a practice. This approach also helps monitoring if the organisation quantitative goals are achieved.

There are three dimensions of aggregation of evaluation results, the already mentioned **target** (practice, method, PI) and **source** (project, department and organisation), to which we add **time**. Aggregation in time is done by analysing the organisation's data in a selected period, given the methods and thresholds at that moment. Aggregation at organisation level indicates the degree of institutionalisation of the practices necessary to achieve generic goals and high maturity, and consequently, allow their evaluation. A project, department or organisation can also use target aggregation to evaluate a method or a CMMI goal/practice. The evaluation by aggregation of colours is done as follows:

- Green – all green;
- Yellow – at least one yellow and no reds;
- Red – at least one red.

We are aware that results aggregation can be complex and not simple sum or median of PIs' evaluations, but the aggregation at source level is out of the scope of this research.

4.3 Repository

Looking at the Data Layer of EQualPI, in particular to the Repository and its modules (see figure 4.7), the repository contains a **Data Dictionary** of the base and derived measures that are used to evaluate the quality of implementation of the practices. Organisation's performance data is stored in this database in the variables of the data dictionary and the way they relate with each other is given by the **Domain Model**. The repository also includes the **Performance Indicator Models**, that are used to evaluate CMMI practices.

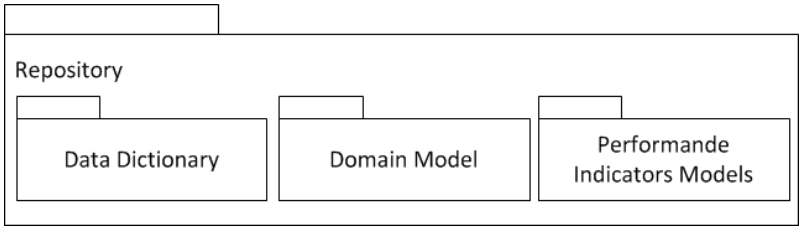


Figure 4.7: Contents of the repository

4.3.1 Data Model

We now describe the data dictionary and its variables and include the schemas of the domain model, to help better understand them. The data dictionary is defined in a matrix where the rows are variables and the columns are elements used to define each variable. Even though the data model was designed focused on getting data from a TSP repository it can be adapted to get data from other development processes.

Figure 4.8 defines the elements of the data dictionary **Data Entry**. A Data Entry is a row of the data dictionary, that is a **Variable**, and has several properties that are represented in the columns: Type of Value, Level, Name, Data Element Name, Definition, Valid Values, Limits or Data Validation, Missing Value, Primary Source, Secondary Source, Role in Research, Type of Indicator, Process, Input or Output.

The Variable includes a **Name**, which is a logical name, a **Data Element Name** that is the unique name to designate that variable in the data dictionary, and a **Description** – altogether these columns are **Definition** elements. The **Constraint** elements that characterise the variable are the **Valid Values**, **Limits or Data Validation** and **Missing Value**. The column **Level** is **Meta-information** used to characterise where the data is collected. Finally, the Data Entry includes information about the **Source** of the variable, which is where the data is normally stored. There are two columns for that purpose, the **Primary Source** that is the most common source for that variable, and the **Secondary Source** that is an alternative place to look for the data if it cannot

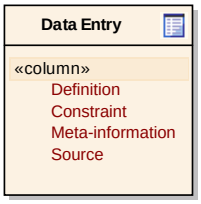


Figure 4.8: Data Entry elements.

be found in the primary source. We first defined the data dictionary based on projects and organisation's data that are collected during projects execution and we complemented it with the data that is collected in TSP projects. Hence, the source of information is present in the data dictionary because we used TSP data to demonstrate the framework and we had to know where organisations could get them.

We represent the structure of the data dictionary in figure 4.9. Each **variable** will have a **time stamp**, a measurement **unit** and a measured **value**. The **valid values** of the variables are defined in the data dictionary. As they are necessary for the performance models the variables are related to a process. The cases we have developed so far relate variables with the **estimation** or **development** process, of which the variables may be an **input** or **output**. Notice that the input of a process can be the output of another. As previously explained, the construct of a performance indicator, that is the **performance indicator model factors**, can be characterised by a set of **controllable** and **non-controllable** factors. The variables play a **role in the model**, they can be **context** variables, hence, non-controllable factors, and may also be **grouping** variables to allow aggregation. Context variables help characterising the scenario in which the data collection is done. Another role that variables play is of **quality indicator**, consequently controllable factors. The quality indicators may be of **performance** (**efficacy** or **efficiency**) and of **compliance** with the process.

There are three dimensions to consider when using the data dictionary:

- **Time:** this dimension has different variables, depending on the level where it is being analysed.
- **Level:** this dimension considers where the data is collected/analysed.
- **Type of Value:** some variables have a suffix that indicates if the value is planned, baseline, actual or benchmark (_<Type of Value>). These suffixes are explained later in this section.

DataCollection_Period is a time dimension that exists at the **Level** *Organisation*, therefore it mainly characterises **Organisation_ID**. It may have more than one value and corresponds to the time interval when the projects/cycles, from which the data was gathered, occurred. This variable can have intervals of time when characterised by **CMMI_ML** and/or **TSP_Partner**, i.e. we can characterise periods of the organisation and expect that the results in projects are different if those periods have an influence on the results of the projects. Even if the organisation is not rated with

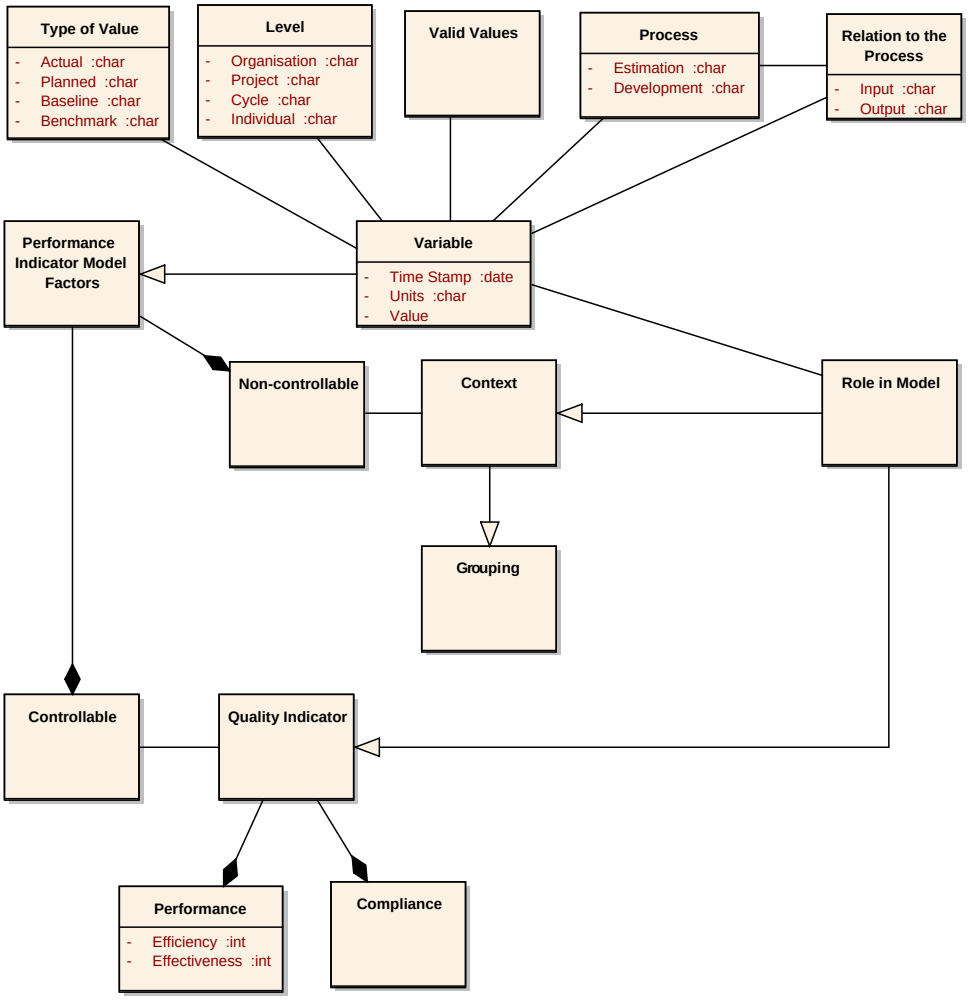


Figure 4.9: Structure of the data dictionary

a CMMI level nor is a TSP partner the time interval is still relevant to characterise the usage of a technology, for example.

Start Date and End Date are related with several time variables such as the TSP launch (**TSP_Launch_Start**, **TSP_Launch_End**), the project or cycle itself, and in that case there are planned, baseline and actual dates: (**Start_Date_Planned**, **End_Date_Planned**, **Start_Date_Actual**, **End_Date_Actual**). **Baseline** also includes a start date and end date and refers to the period of time in which the baseline was applicable. This time interval is different from **Start_Date_Baseline** and **End_Date_Baseline**, which refer to the plan itself.

In figure 4.10 we represent the metamodel of an organisation project, which is based on the SPEM2 (OMG, 2008) and the SPAGO4Q (Colombo et al., 2008) metamodel, and was adapted to

better describe iterative projects of which spiral, prototyping, TSP or agile development models are examples.

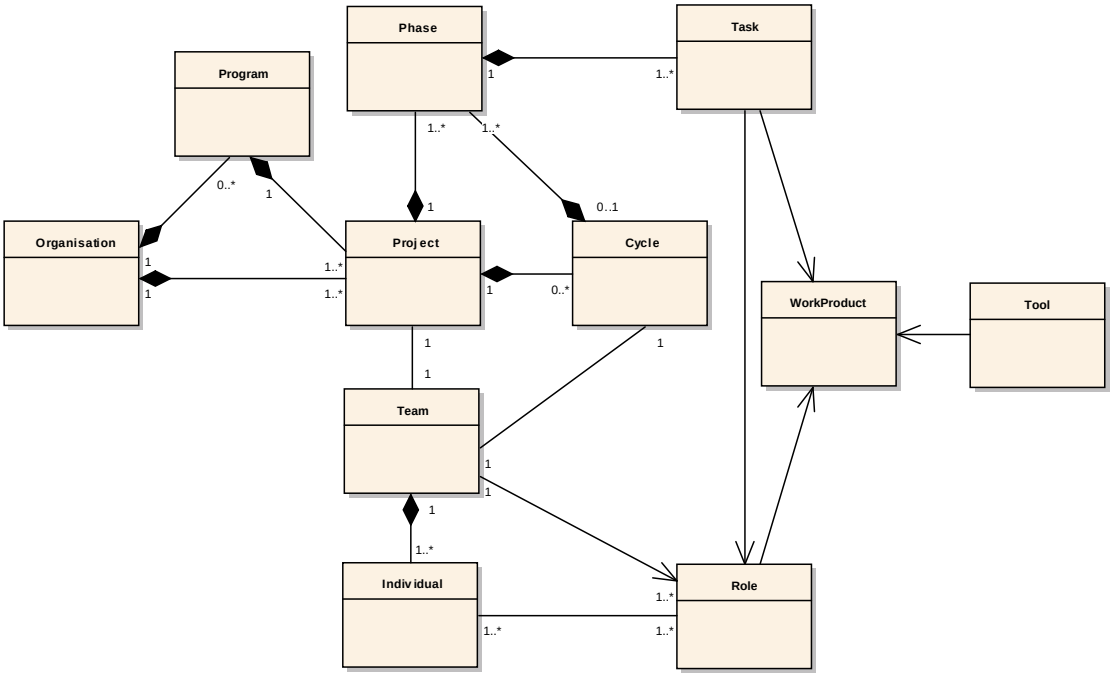


Figure 4.10: Iterative projects overview.

In a high level view an **Organisation** has one or more **Projects**, whose development is comprised by one or more development phases. In certain cases more than a product can be developed through time in different projects, which are related to the same **Program**. In the case of applying TSP, for example, projects are done iteratively in cycles and each **Cycle** comprises one or more development phases. Each **Phase** includes a set of **Tasks** that have a **Role** associated, performed by one or more team members. Those tasks are necessary to develop the **Work Product** and a **Tools** may be necessary to do the task. If the project was developed using Scrum (Schwaber and Sutherland, 2013) the cycle would be a *sprint* and the tasks to execute in a given cycle would be *items* in the *sprint backlog*.

Regarding the analysis of TSP metrics the data can be analysed at different levels (column Level of the TSP Data Dictionary): *Organisation*, *Project*, *Cycle*, *Team* or *Individual*. The individual data are aggregated in team data, team data are aggregated at the project level and the aggregation of projects' data helps to characterise the organisation. On the other hand, the team data can be used to characterise a project cycle. These are the levels where data can be collected and analysed. When using the data dictionary it is possible to filter the variables per Level and only analyse the ones that are at each one of the described levels.

The diagram in figure 4.11 represents the variables that are exclusive of the organisation level.

The Organisation has a *primary_key*, which is **Organisation_ID**. The D**ataCollection_Period** represents the interval of time in which the organization collected the data, however there may be more than one intervals of time if there are different organisation characteristics for that period of time, namely regarding a CMM or CMMI maturity level and/or a period of time in which the organisation becomes **TSP_Partner** (yes or no). The **Business Goal**s may also vary through time or just remain unchanged in the time interval that the data were collected.

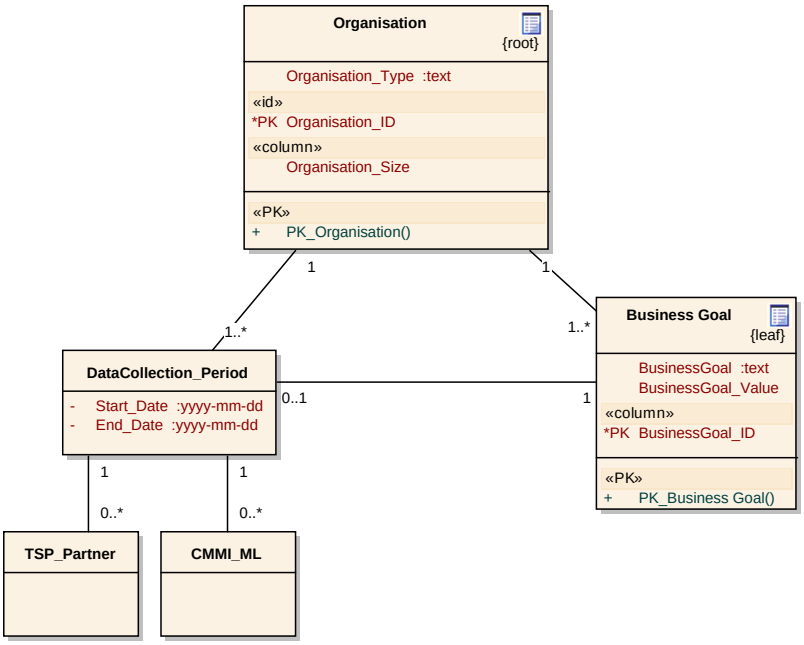


Figure 4.11: Organisation elements.

The variables in figure 4.12 are related to the Project; however, they can also can be defined at Cycle level. A **Project** is developed for a **Client** or several clients and it includes a set of **Stakeholder Goals** that can be of the *Clien*t, different *organisation departments* or *upper managers*, and the *Team*. Any goal is characterised by a description **StakeholderGoal** and can have a target value **StakeholderGoal_Value** that the team tries to achieve. The **StakeholderGoal_Status** gives information of whether the team met the goal or not, and if it has a target value it gives the current value. **Project Goals** include the goals that were already specific of the project but also the goals that the team accepted and with which it is committed; they can include the goals of the remainder stakeholders but can vary in coverage or target value.

All projects have a set of attributes that characterise them. In the particular case of the **ProgrammingLanguage**, it has a meaning when related with the **DataCollection_Period**, the programming language can already be set in the market and well documented and discussed by the community, or it can be in its early stage, where information is scarce and limitations are unknown.

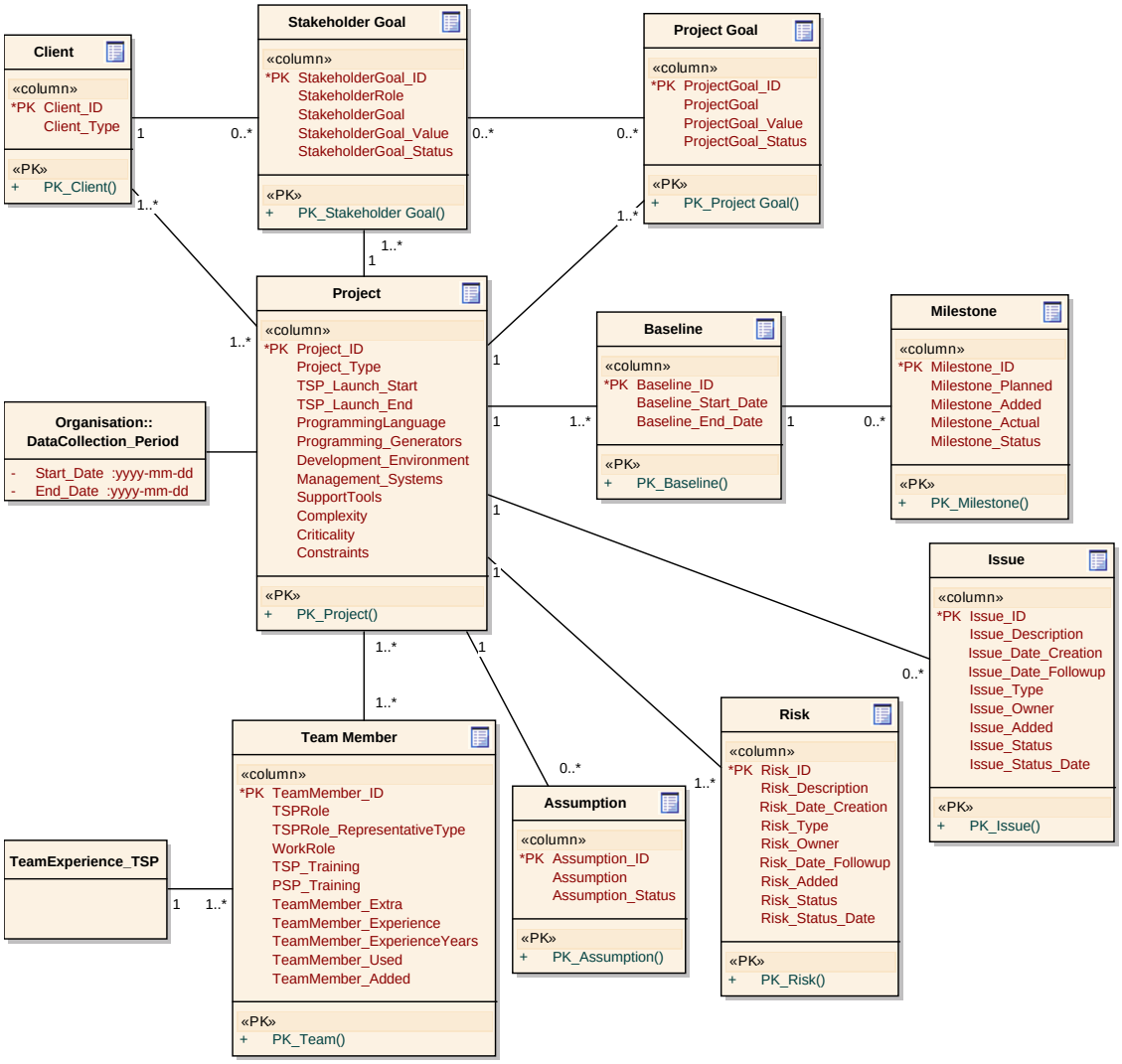


Figure 4.12: Project elements.

A **Project** has a team assigned who executes it. In the particular case of TSP projects there are roles specific of TSP, **TSPRole**, that are assumed by team members, and have Primary and Secondary representatives, **TSPRole_RepresentativeType**. More generically, each member as a **WorkRole**, that determines what the team member does in the project (e.g. *Developer*, *Tester*, *Project Manager*). Not all TSP team members have TSP and PSP training and that is also identified. The difference between a **TeamMember_Extra** and a **TeamMember_Added** is that the extra team member only participates in the project in "at peak of work" circumstances and its participation is short in time. Added team members become part of the team either because their need was not foreseen at the beginning or a change in the scope demands a bigger or more specialised team.

Often, **Assumptions** made by the team regarding the project are documented at the beginning of the project, **Risks** are identified and managed throughout the project and **Issues** are also recorded. They are all updated and managed over the project duration. Project plans have **Baselines** and may have **Milestones**, although in TSP many teams do not identify milestones. In the level *cycle* we identified several variables that are often compared with the baseline.

As mentioned before, in some software development lifecycles projects are developed in cycles that we represent in figure 4.13. In case of TSP, each cycle begins with a launch meeting, where it is planned by the team and a post-mortem meeting, where the cycle is analysed by the team and processes may be updated if necessary. The plan of a cycle is organised in weeks, where tasks are executed by team members in order to produce work products, here designated as **Program Elements**. Those program elements can be *Requirements*, *Detailed Design*, *Test Cases*, etc. or the *Code* itself; in this last case they are part of a **Component** which can be part of a **Module**. In the same way **Task**s are assigned to **Team Members**, so are **Program Elements**. The development of a program element includes different **Phases** and it is only complete when particular phases finish successfully. The **Phase** can refer to an introduction of defects phase, for instance *Coding*, *Detailed Design*, or defects removal phase, such as *Inspection* or *Unit Testing*.

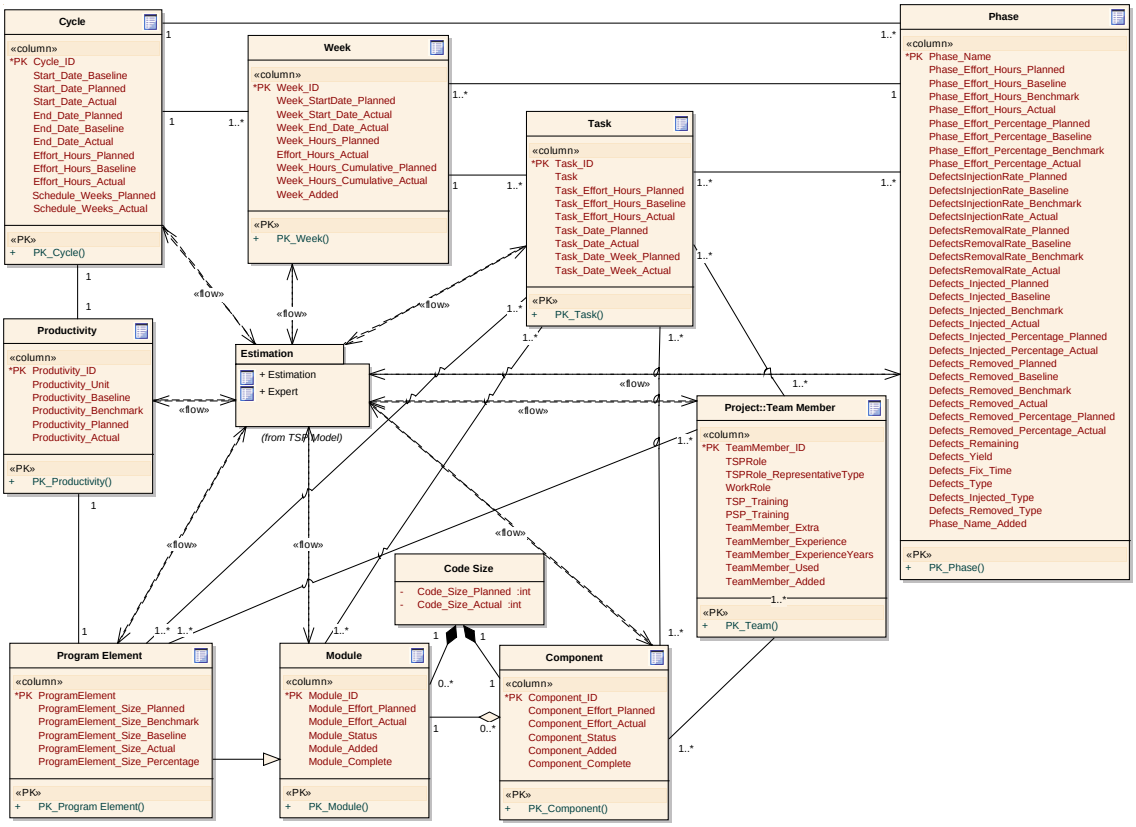


Figure 4.13: Cycle elements.

Certain variables in the data dictionary include **_Planned**, **_Baseline**, **_Benchmark** or **_Actual** extensions in their name. The baseline, as previously mentioned, refers to the variable value in

the current baseline. The planned is the value that is planned for the current cycle. The benchmark value is the value considered when planning the project to support planning and estimating, and the value can come from industry benchmarks, TSP Quality Guidelines Humphrey (2006), TSP recommended values for pilot teams, organisation historical data, expert judgement or changes to any of these benchmarks.

4.3.2 Effort Estimation Evaluation Model

The model of effort estimation uses the data dictionary variables. In this section we focus in the fields **Type of Variable**, **Process** (*Effort Estimation Process* or *Development Process*) and **Input** or **Output**. The following paragraphs describe how the Estimation and Development processes are related when considering the dependent variable (Y) Effort Estimation Accuracy.

The diagram in Figure 4.14 represents the estimation process variables. At the launch of the cycle all the components of the cycle are planned and the estimates of several variables are produced. There is a bidirectional information flow between the Estimation package and all the components of the software development cycle, because the estimates produced affect all planned values in development and the actual values at the end of the development cycle are considered for the estimation of the next cycle.

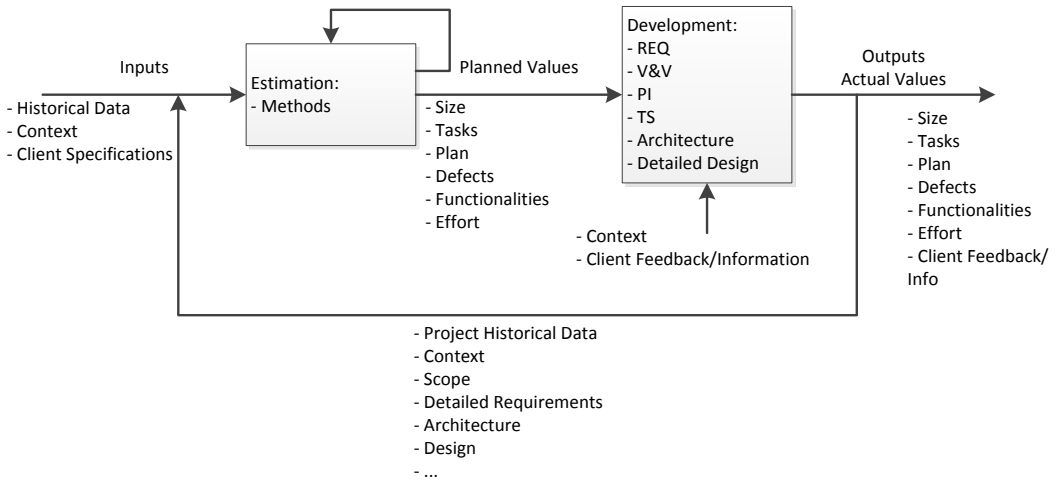


Figure 4.14: Estimation and Development processes feedback loop. The planned values, that are outcomes of the Estimation process, feed the development process, which is also affected by external elements of context and client information, for example. One of the outcomes of the development process is the actual data of how the process was executed, which can be used for appraisal and to feed the organisation database of historical data. Legend: REQ - requirements, V&V - verification and validation, PI - product integration, TS - technical solution.

First Instantiation

When the effort estimation process is first instantiated there are a set of controllable and non-controllable factors that need to be considered. Controllable factors (X_c) (variables on which we can act on) are:

- Choice of what is being estimated and considered to estimate effort;
- Choice of estimation methods.

Non-controllable factors (X_{nc}) (variables that we cannot yet control):

- Context;
- Scope;
- What will happen during the development cycle.

To evaluate the quality of implementation of PP SP1.4 these two parcels (controllable and non-controllable factors) need to be considered. If we determine the percentage of the effort estimation accuracy that each of these parcels represent we will know what is the percentage that we can control while estimating. To determine the effort estimation accuracy at a given point in time $t = i$ we can consider the deviation as being a partial variation (equation 4.2).

$$ParcialEffortEstimationAccuracy: \quad EEA_t = \frac{Actual_t - Planned_t}{Planned_t} \tag{4.2}$$

The value of EEA_t is determined considering only the effort estimated for the tasks that were planned to the next moment of estimation and executed up to that moment.

The global deviation considers the project from start to end (see equation 4.3). The planned values are the ones from the first plan or proposal.

$$GloablEffortEstimationAccuracy: \quad EEA_{global} = \frac{Actual - Planned}{Planned} \tag{4.3}$$

The estimation process is instantiated throughout the project, as the uncertainty about it diminishes over time, the new knowledge must be considered in the project plan. It would be a mistake not to consider the new information to better define what needs to be done, how it will be done, the sequence of tasks and their duration. Ignoring such information to re-plan would be as ignoring information to build what the client truly expects that can be obtained from the technical knowledge that is gradually obtained.

N^{th} Instantiation

The outputs and actual values of the Development Process at the moment of a n^{th} instantiation of the Estimation Process become a valuable input for it and include what part of the plan was already done. In other words, it is necessary to consider the actual values up to the execution phase at which the project is. Tasks that were finished earlier than expected leave slots of time that

can be used to execute other tasks, people that become available earlier can start following tasks or help other team members that are late to finish theirs.

Every output of the development process that adds detail to what is to be done and how it needs to be done (such as detailed requirements, architecture, design, scope), change requests to what have been done already and requirements that are added, changed or eliminated by the client – all imply re-planning. Another aspect is the number of defects that are actually being detected, which can trigger changes in the verification strategy, for example. By the analysis of complexity and execution of the task itself it can be concluded if it is necessary to add or remove effort in order to finish the task, or even add people or extra tasks, such as training.

Part of the non-controllable factors of the first instantiation of the estimation process are considered in the n^{th} instantiation becoming controllable. The non-controllable parcel only becomes more controllable if the agents that contribute to diminish uncertainty are considered when re-planning. Then, there are re-planning cycles as there are development cycles. The objective of this approach is to benefit from the reduction of uncertainty to better and more accurately plan the next steps of the project. It is necessary to calculate a variation that is a sum of the parcels deviation and compare it with the global deviation previously defined (see equation 4.4).

$$TotalEffortEstimationAccuracy: \quad EEA_{total} = \sum_{t=1}^{k} EEA_t = \sum_{t=1}^{k} \frac{Actual_t - Planned_t}{Planned_t} \tag{4.4}$$

4.4 Manage Configurations

To have a CMMI implementation and analyse its performance through time it is necessary to manage the configurations. The module **Manage Configurations** in the business layer, is where the configurations of the processes are stored, so they can be used when necessary. Those include the following implementations:

- **Current** - the implementation that is loaded since the last setup of the framework;
- **Previous** - configurations of previous implementations are saved in the database. They are loaded when the user intends to see the history of evaluations, as an evaluation is based on the organisation data and the configuration of the framework;
- **Pilot** - configurations of process improvements in pilot projects. It may be possible that they never leave the pilot state, if the organisation realises that they do not benefit the organisation;
- **Deployment** - configurations that result of successful pilots and are gradually being deployed in the organisation. The deployment configuration co-exists for a period of time with the current implementation, and when a selected percentage of projects is already following it with the desired performance results, the deployment implementation replaces the current implementation that is saved in the history.

- **Updated** - used to establish the new baseline once the deployment of new configurations is completed.

Current and Previous Implementation

The implementation configuration is the one reflecting the current processes configuration that exist in the organisation. To evaluate that implementation the framework uses all data of the methods that are currently in place. If a pilot is in place or a process improvement is being deployed the data of those configurations are not used to evaluate the current implementation. Otherwise it would not be possible to compare a pilot or deployment configuration with the current implementation configuration. The previous implementation is also stored.

Pilot

The organisation may start one or more pilots to do a process improvement. The pilots must be isolated from the organisation's current implementation and from other pilots. Based on the current configuration the organisation selects the change to introduce in the process improvement: changing a method, using a different tool or changing a procedure. For that pilot one or more projects will be done and the related data will be collected, eventually new performance indicators will be monitored. The pilots will be executed for a period of time. The data collected to monitor them will not be loaded when evaluating the current implementation of the organisation, are loaded only to evaluate the pilot. To verify if the changes improved organisation performance, that is to evaluate the pilot, for a period of time the pilot data are compared with the data of the current configuration in the same period. If the pilot shows better performance than the organisation, the deployment can be done. Even if the organisation does several different process improvements in parallel only one can be deployed at a time.

Deployment

To prepare the deployment configuration the current configuration is loaded with the change done by the pilot. The organisation begins gradually deploying the changes done and collects the data on the projects that use the changes. In parallel the organisation keeps evaluating the current implementation to ensure that the performance of the deployed improvement is better than the one of the current configuration. However, the deployment may be not static. As people use a new tool or functionality and/or a new methodology the process may need to be adjusted. So, after the change, new data is collected and compared with the current implementation. When the deployed improvement affects a significant part of the organisation (and that is a decision for the organisation to make) and the process and tools are not updated for a given period of time, the improvement is compared with the current implementation. If the performance is considered to be better and other process areas are not negatively affected the deployed improvement may become the current implementation. The organisation may keep the improvement and involve the entire organisation, updating the current implementation accordingly. That only happens after training and having tools in place so everyone is ready to use them. Furthermore, if the organisation has a process performance model and baseline, they need time to become stable to be able to compare

the new performance. This may be one criterion for the organisation to keep the changes in deployment longer.

Updated Implementation

The new configuration can be loaded from the stable deployment configuration. A new baseline is established. If the data since the final deployment is stable it may be loaded in the current implementation and the baseline may be set from there. From that point on, the whole organisation operates with the new process and that is the one that is evaluated.

4.5 Procedures

The Business Layer's package **Procedures** (see figure 4.15) and its modules give organisations the necessary information to use the framework in practice.Procedures include:

- Instructions to deploy the framework and align it with the organisations practices and goals;
- Checklist to support the CMMI implementation;
- Instructions to populate the EQualPI database and calibrate the models;
- Instructions to evaluated the implemented practices;
- Instructions to conduct processes improvements pilots and deploy them.

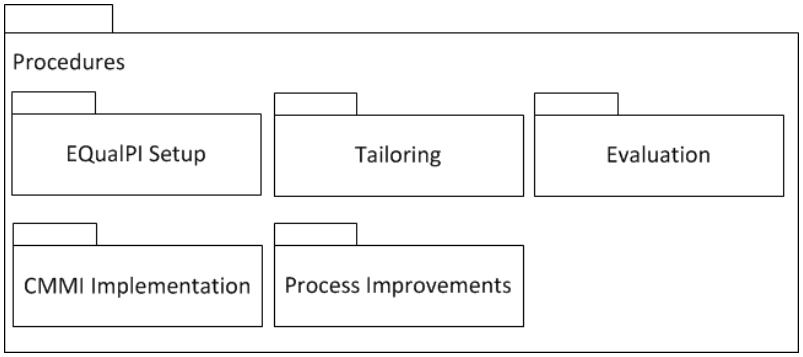


Figure 4.15: Contents of the repository

4.5.1 EQualPI Setup, Tailoring and Evaluation

The preparation of the framework by an organisation, or setup, is done following the repository metamodel that we presented in figure 4.5. The organisation must choose a reference model in the CMMI framework, it may wish to implement a process area or a maturity level, or it may implement just a subset of process areas that do not allow achieving a capability or maturity level

but are relevant for the organisation. The organisation may choose to implement all specific practices to achieve a goal or it may indicate alternative practices. Similar choices can be made in the case of implementing a generic goal, by following the generic practices or following alternative ones. After this, the organisation must indicate the methods used to implement the goal or practice, and the performance indicators to evaluate them. The methods are selected from a pool of methods existent in the EQualPI Repository along with the corresponding performance indicators, but they may also be added by the organisation. EQualPI allows the implementation according with the CMMI model, and the user just indicates the level and representation and all information is automatically loaded. The steps of the setup are represented in figure 4.16.

After EQualPI is configured, and practices are mapped with methods and performance indicators, the organisation needs to setup thresholds and quantitative goals per Performance Indicator. The data used for that purpose, in case of not having historical data, can be a benchmark from the industry. Depending on the indicator, the thresholds define performance intervals of acceptable (green), alarming (yellow) or unacceptable (red) behaviour, triggered by reaching a given value or going out of an interval of values.

To populate EQualPI's database with **Organisation Metrics**, the organisation exports information of the company database in the format of the data dictionary. When an organisation does an evaluation the database needs be updated until the end date that the evaluation refers to. The organisation may also choose the source (organisation, departement, project) and target (performance indicator, method, practice) of the evaluation, as we explained in 4.2 EQualPI Architecture.

In practice, when an organisation uses the framework it tailors the CMMI process areas/practices, methods, and corresponding performance indicators, that will be used. The organisation executes the evaluation process, after defining the thresholds for the performance indicators according with its business goals. From executing the evaluation framework the organisation will get a colour for the quality of implementation of the CMMI practices, organisation performance and impact of performance improvements (in case it changed its processes). The performance indicators are based on the business goals, which may be financial, marketing, quality and related to the customer. Those goals are drilled down to departments and projects goals. Having goals related to better planning, faster development, customer satisfaction, etc., we can map them with performance indicators that show us the predictability, productivity, re-work, defects delivered, etc.

To setup the framework an organisation must follow a set of steps according with the flowchart in figure 4.16:

1. Identify business goals;
2. Select the Reference Model (CMMI constelation(s)) to use;
3. Select the model representation (staged, continuous or none);
4. If staged was selected, select the Capability Level or the Process Areas to implement;
5. If continuous was selected, select the Maturity Level or the Process Area;
6. If none was selected select the Process Areas and Generic Goals to implement;
7. Select the Specific Goals;
8. The framework suggests the use of Specific Practices or the user may use Alternative Practices;
9. If Alternative Practices were selected the user must indicate them and map them with the Specific Goal;
10. If Specif Practices were selected they will be mapped under the corresponding Specific Goals;
11. Per practice or alternative practice select the method(s) and indicate which ones are mandatory, alternative and optional;
12. If a method is not in the system add it and map it with the practice it implements;
13. Per method EQualPI suggests a list of Performance Indicators that are in the pool, according with the maturity profile (in case the PAs selected do not allow to achieve a level, then the Performance Indicators suggested indicate the maturity profile). Select the performance indicators;
14. If the desired performance indicator does not exist specify it according with the metamodel and data dictionary, and map it with the method.

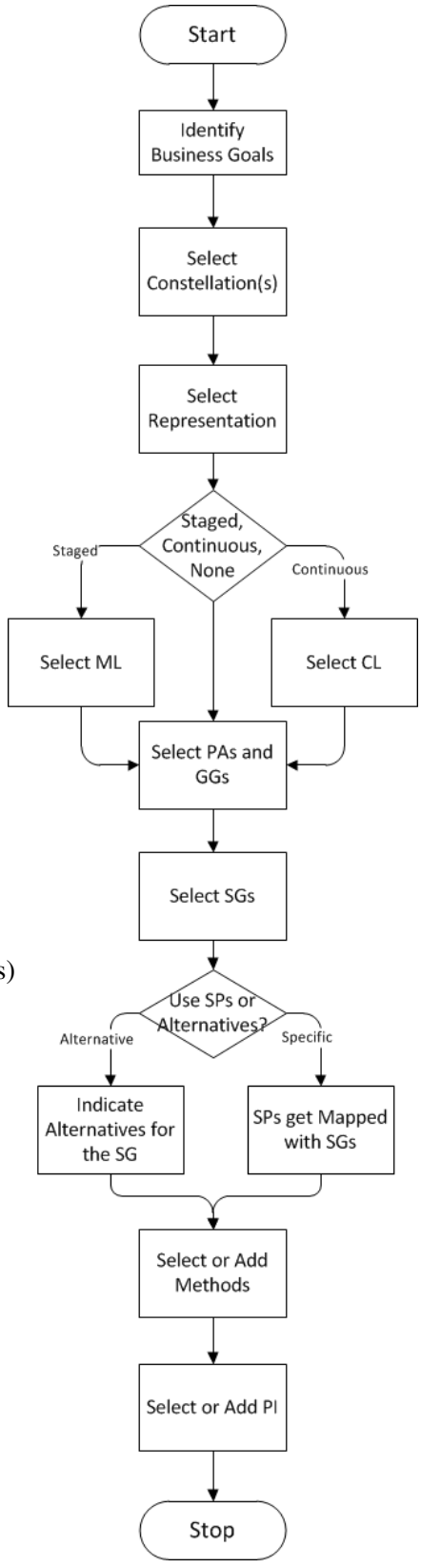


Figure 4.16: Flowchart of the setup of the EQualPI framework.

An example of evaluation is depicted in figure 4.17. The analyst or stakeholder would select a PI to analyse in a period of time. In the example given the PI is *Schedule Estimation Error*. When analysing a project the person verifies in which interval of values the data point stands (in case there are upper and lower thresholds) and analyse the colour of the evaluation accordingly. The evaluation of a department would be given by aggregating the results of the department's projects. In the example, department 1 (D1) has three projects (P1, P3 and P4) and one is red, so it is evaluated as red; D2 has a single project that is yellow, so the result for that department is yellow; and D3 has one project P5 that is green, so the department evaluation is green. When evaluating the organisation, D1 was evaluated as red so the organisation is red.

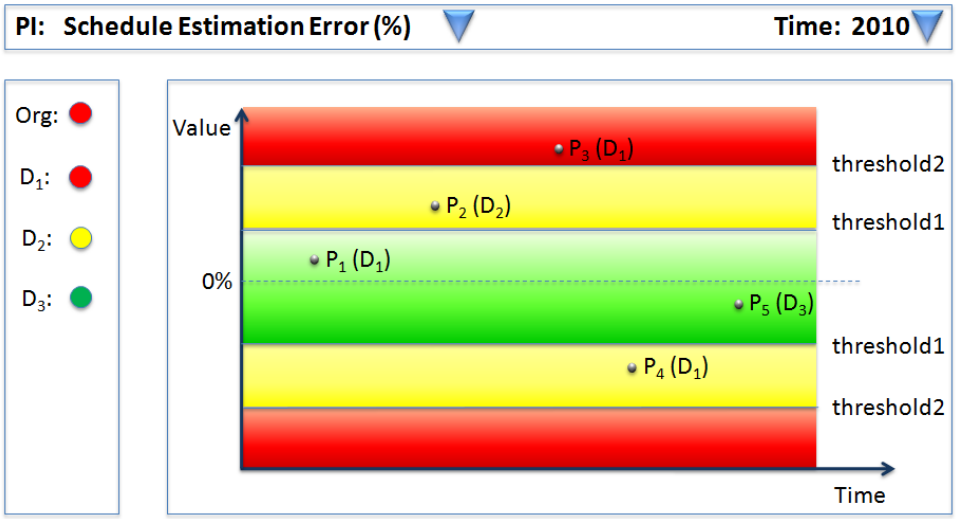


Figure 4.17: Evaluation of the Schedule Estimation Error. Legend: PI - Performance Indicator, Org - organisation, D1 - department 1, P1 - project 1.

The example represented in 4.18 demonstrates the aggregation of results from a PI to a method and to a goal or practice. The organisation has several methods that are part of its processes. Some methods are mandatory, others are optional and in some cases there is a pool of alternative methods and the team can choose one of them. The methods to use can also be imposed by the lifecycle in case of a development process or the methodology used for project management (for example use Scrum or TSP). When a project begins it is necessary to choose the methods that are going to be used. In the figure methods 1 and 2, M1 and M2, are alternative methods to do the same activity, M3 is an optional method and M4 is mandatory.

Taking the example of project P1, they chose the estimation method M2 and have to monitor PI1 and PI5. At the moment that the project is analysed both indicators are green. The team opted to use an optional method M3, monitored with PI3 and had to use M4, because it is a mandatory method, and consequently monitored PI4. Since PI3 is red the project is red and so is M3. For that project M2 is green because both PI1 and PI5 are green and M4 is yellow because PI4 is also yellow.

When analysing CMMI specific practices we can see that SP1 uses M4, SP2 is mapped with M1 or M2, SP3 with M1 or M2 and M3 and SP4 with M1 and M4. For project P1 SP1 is yellow, because M4 is yellow, SP2 is green because M2 is also green and SP4 is red, because the team decided to use the optional M3 and it is red. The project is red in terms of PI, methods and SP.

If a department has several projects it may evaluate its state by aggregating the results of its projects. In the case of department D1, P1 is red which implies that the department is red too. The department can also analyse its results in each PI by aggregating the PI results in each one of its projects. In the case of D2, PI3 is yellow because in one of the projects the indicator was yellow and the other did not have a red. A similar analysis can be done to evaluate how the department is performing each method and each SP.

To evaluate the organisation the analyst aggregates the results of its departments, in this example D1 is red so the result of the organisation is red. The organisation can also evaluate each PI, Method and CMMI goals or practices by aggregating the results of each department. For example, SP1 is yellow because D1 is yellow and D2 is green.

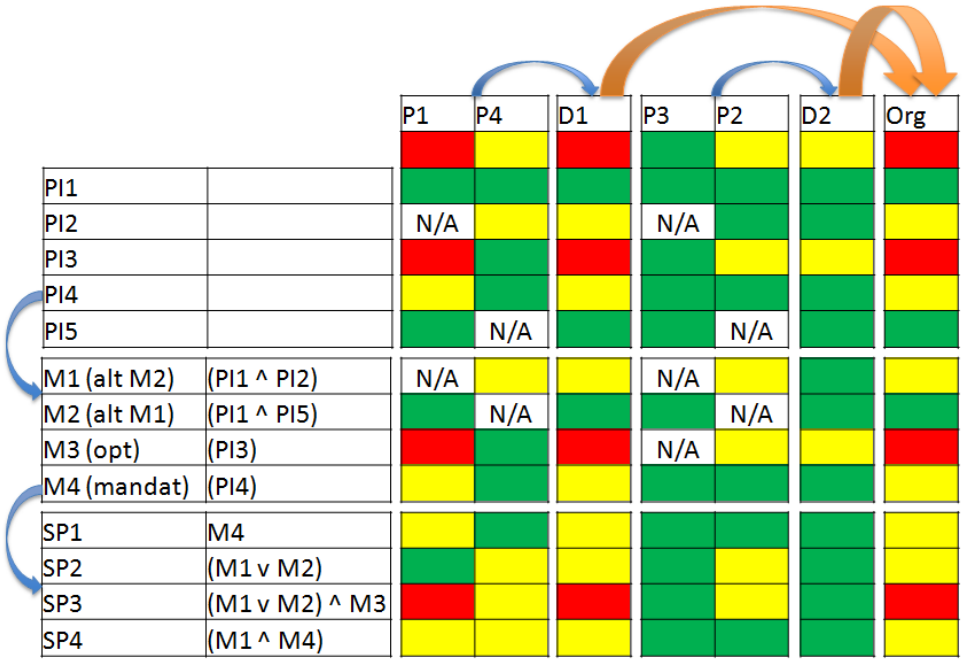


Figure 4.18: Aggregation of evaluation in the source perspective and target perspective. Legend: PI - Performance Indicator, Org - organisation, D1 - department 1, P1 - project 1, alt - alternative, opt - optional, mandat - mandatory, ^ AND, v - OR.

4.5.2 CMMI Implementation

The module **CMMI Implementation** provides a checklist of problems that may occur when implementing CMMI and a set of recommendations (R) on how to implement the model based on

the problems (P), which we numbered. The implementation checklist is based on the literature review that we conducted (discussed in 3.1.3 Problems in Process Improvements, Metrics Programs and CMMI), the analysis that we did of the data of a survey performed by the SEI and three case studies that we conducted in three high maturity organisations (research detailed on 5.2 CMMI HML Implementation). We focused on how organisations prepare for SCAMPI when implementing CMMI to achieve a maturity level.

The literature review showed us that many of the problems found in the high maturity levels of CMMI are actually based in ML2. Next we detail the problems and challenges organisations face when implementing CMMI:

P1. Underestimate time to implement HML

Processes improvements require time, to define, implement, involve people, train them and refine practices before they are stable. CMMI just provides guidelines, so it is still necessary to plan the time required to define the processes themselves. Moreover, when considering high maturity there must be cycles to complete data collection and analysis to refine the models, particularly if they are being implemented for the first time. It is important to plan all this work.

P2. Introduction of HML forgetting ML 2 and 3

CMMI builds maturity/capability that is base from one level to the next. The practices, metrics and standard processes from ML 2 and 3 must be well defined as requirements for HML practices.

P3. Understand the statistical nature of level 4

Understanding the statistical needs to achieve ML 4 is not reduced to have knowledge of statistics and modelling but knowing how to use them in the business without falling in the traps of blind interpretations or over relying in models and baselines, without critically analysing them in the context of the process execution, e.g. context events occurring when a project was using the process.

P4. Copied processes

As stated before, CMMI provides guidelines, not processes. Using the model as is and considering it the process, leaves behind many details of how to do the work, hardly reflects the organisation reality and culture and may not add value.

P5. Multicultural environment and **P6. Impose processes**

The processes must reflect the organisation culture that is also why people must be involved in process definition. An additional benefit of this involvement is that it helps people understand, relate with and embrace the changes.

P7. Dissemination problems

People must be aware of the changes taking place and have all information necessary to do their work once they start using the new processes; hence the importance of coaching, training and providing tool tips and help with the software to use.

P8. Lack of institutionalisation

Besides ensuring right level of dissemination and people involvement, to have institutionalisation also implies that different contexts are understood and contemplated in the program.

P9. Meaningless uncorrelated metrics

The metrics used shall be useful, that is to say, provide information that is needed to achieve a goal. Furthermore, one must not fall in the temptation of blindly trusting causality or establish relations between what is not correlated.

P10. Metrics definition (collect and analyse data)

Define metrics unambiguously, ensure that they are measuring exactly what is needed and knowing to interpret them.

P11. First data collected were uncorrelated

The first data collection may not provide the correlations the organisation is looking for, so it has to refine and do new cycles.

P12. Metrics categorisation

In the first cycles of data collection, at times the baselines are not stable enough. Also, unless there is already a considerable amount of historical data, it not possible to distinguish between different categories of data (for different markets, team experience, team sizes and project sizes). Depending of the context some metrics may no be adequate to measure everything, for example, the defined size metrics need to the nature of the work being measured.

P13. Baselines not applicable to all projects

Having unstable PPBs not specific to the different contexts. Time to collect data, insufficient to gather information of different contexts and verify if new metrics were needed, there were differences in performance and in which context, in certain circumstances, the procedure to collect the data should be different.

P14. Abusive elimination of outliers

Outliers eliminated without understanding whether they were special causes of variation or not, for example, discarding relevant information to the execution of the project.

P15. Not all projects are measurable

Different projects cycles require tools adaptation and in some cases specific metrics, to be measurable.

P16. Effort estimates

Not having different support baselines/tools for different effort estimation methods. For example, having expert judgement without the support of previous knowledge of work-product size and task duration, simply because there are only PPB adequate for code development projects.

P.17 People behaviour

Failing to show people the value of practices that should be applied on their projects or work, and consequently having people not using them or considering they are not applicable to their projects. Also resulting in careless data gathering, compromising their accuracy.

P18. Tools setup and **P20. Tools requirements**

Tools require time to be stable, new defects can be found only when they are already in use and changes to the original requirements may be revealed by usage in work context.

P19. Overhead

When data collection is not fully automated, manual collection may introduce overhead, besides increasing error caused by human mistakes.

The demands of levels 2 and 3 should prepare organisations to adequately use measurement at higher levels, by monitoring appropriate metrics. Nonetheless, some of the problems here identified reflect poor implementation of the MA PA, affecting the organisation results. Such problems become evident when implementing ML 4 because the correlation between variables and problems in the collected data, affect PPM and PPB. Besides, SCAMPI cannot appraise the entire organisation and does not analyse performance measures – if it did, it would become even more expensive. Hence, CMMI rating per se is not a guarantee of achieving expected performance results and organisations need to be aware that there are different methods that can be used on its implementation. Nevertheless, if some recommendations such as the ones we propose in this section are followed, CMMI implementation can be easier, and the problems discussed before can be avoided. Most of these recommendations are solutions used by the studied organisations to overcome their problems.

Entry Conditions

When planning a move towards HML respect time to: have mature levels and institutionalised practices; understand and analyse the needs for HML; find correlations between variables; reach stable metrics, processes, tools and work habits; select meaningful performance indicators

and gather enough stable data points to have statistically meaningful historical data. Organisations need to carefully plan business and process improvement objectives, temporal horizon and resources (time, internal and external human resources, tools, training, etc.). **(R1)**

HML only work with a stable base, hence, introducing their practices can only occur after ML 2 and 3 are mature and institutionalised (Leeson 2009) **(R2)**.

HML require understanding statistics and modelling. Guarantee the involvement of an expert in quantitative methods (e.g., statistician), preferably with experience in software and if possible also in CMMI, who can help better understanding processes behaviour and correlations between variables, along with providing adequate statistical tools to different contexts **(R3)**. Introducing a Six Sigma initiative in the organisation eases the introduction of the statistical knowledge necessary to the organisation workers **(R4)**.

There must be a top down and bottom up revision of the organisation's processes, improvements/innovations, goals and quantitative goals **(R5)**.

Process Definition and Implementation

The implementation of the model should reflect the culture of the organisation, and not be a copy imposed on personnel. Processes definition should identify current processes (as is) and improvements (to be) so they reflect organisation's culture and people good practices (Leeson, 2009) **(R6)**. When defining processes it is important to involve the experts, including those who use the process to do their work: project, technical and quality managers; developers; testers, etc. **(R7)**.

In multicultural organisations and when acquiring new companies imposing processes can result in a loss of knowledge and resistance to change. Different business units should share practices used and lessons learnt. Each business unit would then gradually and naturally adopt the other's practices if they better fulfilled needs. This approach allows creating processes without losing good practices, benefiting from cultural differences. **(R8)**

There should be goals specific for different business units, departments and projects, which must be related to the organisation business goals **(R9)**. Such setting allows having goals monitored at all levels, avoiding loss of visibility by middle management in each level **(R10)**.

Commitment from the entire organisation is essential, including top management, middle management and the people who are actually doing the work (Leeson, 2009) **(R11)**. Training needs to be adequate for each role and to include not only the what to do, how to do and hands on components but also the why shall we do it, what will we achieve and how do we see it **(R12)**. Top management needs to set goals, plan gradual institutionalisation, monitor and reward **(R14)**. However, when organisations are large they should consider even more gradual dissemination, spreading practices in a small group of projects and gradually involving new ones, which can be done also profiting from team members' mobility. For that it is essential that they understand the processes. To have people commitment it is crucial that they understand the new practices, which can be achieved by coaching projects and people (Humphrey, 2006), guiding and accompanying them **(R13)**.

Metrics and processes definitions mature when used in practice because it is when problems arise that it becomes more evident how procedures can actually be done. It is necessary to give some time to let processes and metrics mature before producing their final versions. **(R15)**

Metrics Definition

To establish business objectives and identify the indicators of the processes performance, organisations can use methods such as the goal-driven measurement (Park et al., 1996) **(R16)**.

Understanding metrics is a process that is completed when projects are using them as the final processes define. It is utterly necessary to train the entire organisation without undervaluing the effort in such tasks **(R18)**.

Measures need to be defined with a set of repeatable rules for collecting and unambiguously understanding data and what they represent (Florac et al., 2000), if different people use them differently, then their definition is inadequate. The level of detail of metrics needs to be completely defined, understood **(R17)** and to consider the different types of projects' context, including the technology used **(R19)**. For example, in some technologies there are more KLOC, the time to execute unit tests is negligible, etc. Another example is project type: outsourced, maintenance and development projects, for instance, will have different measurement and control needs. Those factors affect the metrics definition.

Define basic software processes about which data should be collected, then concatenate and decompose data in different ways to provide adequate information at project and organisation levels (Kitchenham et al., 2006). If necessary, data should be normalised to make them visible to top management. **(R20, R24)**. It is preferable to begin with a sub-process executed often and with a small number of variables so results come faster **(R21)**. When the process is stable, then extend to other processes and more complex ones (Florac et al., 2000).

Metrics databases take time to become stable and allow the construction of relevant PPM and PPB. The data need to be categorised (R23). (Florac et al., 2000) refer to this process as "separating or stratifying data that belong to the same cause system". Nonetheless, to have adequate categorisation it is necessary that the different projects fully cycle to completion. Either the organisation has a significant number of concurrent projects with small lifecycles or it begins to work with first limited baselines that evolve with time. **(R22)**

Pilot projects are useful for stabilising processes, procedures and tools. The way people use tools may change the way metrics should be collected. Only after those projects are over and the practices are clearly defined, will the organisation be ready for training, the processes/procedures and tools be fully and correctly documented and people be able to learn and apply the practices. Changes may then be deployed so that processes become institutionalised.

Metrics Usage

Certain outliers can be removed from databases but it is necessary to pay attention to those not immediately understood. They can indicate that a process is having a new behaviour (better or worse performance), be a common situation or indicate the existence of a different process, with

a different behaviour and therefore originate new sub-processes (Florac et al., 2000). One way of avoiding the error of abusively eliminating such outliers is to monitor the process without the outlier in parallel to the process with the outlier, then decide the most adequate action:

- perform CAR;
- eliminate the outlier;
- establish a new baseline because process performance improved;
- create new sub-processes, in case of having sub-processes.

Florac et al. (2000) give an example of how to do it. **(R25, R26)**.

Regarding effort estimation, expert judgment is more adequate in certain circumstances, in particular when there is absolutely no previous knowledge of the project (Grimstad and Jorgensen, 2006) **(R28)**. Effort estimation does not necessarily need to be based on KLOC to be based on historical data; it can be based on other size metrics, phase duration or the time spent on task **(R27, R29)**. When no data are available at all do iterative planning, so that when data from a previous cycle are available they can be used to plan the following **(R30)**. To have useful, reliable data, the personal data shall not be used for evaluation purposes **(R31)**.

Tools Setup

Manual data collection is time-consuming and error prone (Hamon and Pinette, 2010), so it should be automated. To avoid overhead in the collection process, the information system needs to have limited human intervention, e.g., reporting effort and measuring code. Effort spent on different software applications for doing the tasks may be measured and part of the effort automatically labelled; the person only verifies and corrects eventual errors by the end of a block of tasks. This avoids forgetting to report effort or constantly interrupting tasks to manually report. The information system should be composed by automatic storage tools connected to the development environment (Johnson et al., 2005). **(R35)** It is imperative that data collection is precise, if it was not so previously, people need to change their mentality and display discipline **(R36)**.

It is important to understand that tools need time to be set up, especially when evolving existent ones **(R32)**. The data collected when correcting those tools defects, which have impact on the definition of the metrics and of the process should not be used to build PPB, because the process is not stable. For the same reason, PPM may also need to be recalibrated, for example. **(R33, R34)**

We compile all problems and recommendations in a checklist (see table in figure 4.19) to be used by organisations when implementing CMMI. The checklist provides guidance in the sequence of what shall be done to implement CMMI, gives organisations focus in the model as a whole, and not only a single target level to be achieved, and includes the problems that organisations should be aware of in order to avoid them.

Category	Problem	Recommendations	Refs	PA/GG
Entry Conditions	*P1. Underestimate time to implement HML* , it takes long to accumulate meaningful historical data.	R1: Plan considering activities such as maturing levels, analysing and understanding HML, maturing PPB and PPM, collecting data repeatedly until meaningful performance indicators can be systematically obtained.	CI,CIII, DS	
	P2. Introduce HML forgetting ML 2 and 3	R2: Before moving to HML guarantee that ML 2 and 3 are mature and institutionalised.	CIII,L	
	P3. Understand the statistical/quantitative nature of level 4: Underestimation of time to change mentality from ML 3 to quantitative thinking, and time to implement ML 5. Insufficient involvement of management/stakeholders in models decisions. Ineffective models.	R3: Involve a statistician with experience in software and preferably on CMMI. R4: Introduce Six Sigma initiative. R5: Review goals and quantitative goals top down and bottom up when implementing CMMI.	CI,CIII, DS,HS, TBT L	MA OPP QPM
Process Definition and Implementation	*P4. Copied processes:* from CMMI.	R6: Processes shall reflect the culture of the organisation, not be a copy of the model imposed to the personnel. R7: Involve experts and process users in the definition of processes.	CII L	
	P5. Multicultural environments: people dealing differently with change.	R8: Interaction between business units to share processes and lessons learnt to design processes together.	CII	
	P6. Impose processes on acquired organisations with the loss of good practices.	R8, R9: Have goals specific to different business units, departments and projects, related to the organisation business goals. R10: Have indicators to monitor them at different report levels.	CII,16	
	P7. Dissemination problems: Difficulties in applying new practices, in particular in understanding how to collect, analyse and interpret metrics. Managers using PPM/PPB do not understand results and PPM modellers lacking mentoring/coaching and access to a statistician.	R11: Have commitment from the entire organisation: involve top management, middle management and the people who are actually doing the work. Have a sponsor. R12: Training shall include what to do, how to do, hands on, benefits and how can benefits be seen. Have different levels of training. Specialised training for sponsors and top management, process group and all roles that are affected by changes. Train top management on: sponsorship; goal setting; monitoring and rewarding (at different goals levels); on the process (understand it). R13: Coaching of projects and people (guiding and accompanying) and monitoring (from top management).	CI,DS L,H	GP2.5 GP2.6
	P8. Lack of institutionalisation: Not all projects used the new practices. Lack of adherence to processes or unused processes.	R14: Top management: set goals (when, who, what); include goals for gradual institutionalisation, monitor and reward. R13, R15: Metrics and processes definitions mature when used in practice, need time to define final versions.	CI,CII, R,S,CD M, HP	GG 2 GP2.5
Metrics Definition	*P9. Meaningless Uncorrelated Metrics:* Misinterpretation of metrics due to lack of context information; useless indicators and at times not aligned with business and technological goals.	R16: Use goal driven measurement, or equivalent, to establish quantitative goals. R17: Measures defined with a set of repeatable rules for collecting and	CI,DS,K PGF, FCB	MA SP1.4 MA SP2.2
	P10. Metrics definition (collect and analyse data): Not adequate to all contexts, vague, allowing errors in collected data due to different interpretations, inconsistent measures for aggregation across the organisation.	unambiguously understand the data and what it represents. R18: Use different size measures according with the work product. R19: Identify different context that need to be associated with the metrics in order to adequately interpret them. R20: Do variables normalisation to ensure that metrics are usable in the entire organisation.	CI,CII, DS,G,B HP, K, FCB	MA SP1.3 MA SP1.4 MA SP2.1 MA SP2.2
	P11. First data collected were uncorrelated and too many indicators at the beginning.	R21: Conduct all necessary data collection cycles to find correlated metrics.	CIII, HP	OPP
	P12. Metrics Categorisation: Not all contexts data available. Unstable baselines without different categories. Not enough contextual information for data aggregation.	R22: Give time for the metrics databases to become stable and allow the construction of relevant PPM and PPB. Different projects full cycles completed.	CI,CII, DS	OPP
	P13. Baselines not applicable to all projects	R23: Categorise data. R24: Aggregate normalised data only for global view.	CI,CII	OPP SP1.3 OPP SP1.4 QPM SP2.2
Metrics Usage	*P14. Abusive elimination of outliers:* exceptional causes of variation occurring once per project or new baseline being established.	R25: Quarantine outliers which cause is not immediately identified. R26: Recognise data points that are not outliers but are unique and recurrent.	CI FCB	MA SP1.4 MA SP2.2
	P15. Not all projects are measurable: Not collecting data from projects with a data structure different from the standard. Not using all derived metrics because of lack of definition of base measures adequate to context.	R27: Base measures should be defined for different work and then normalised to allow calculating derived measures. R14	CI,CII, DS	MA SP1.3 MA SP2.3
	P16. Effort estimates: without using historical data of effort or size.	R28: Expert judgment is more adequate in certain circumstances. R29: Use any related historical data: size, phase duration, time spent on task. R30: Do iterative planning and use real time sampling of processes when there is no previous data available.	CII GJ, DS	PP SP1.2 PP SP1.4
	P17. People behaviour: inaccurate personal data reports. Resistance to collect new/additional data after ML3.	R12, R13, R31: Never use personal data to evaluate people.	CI,CII, DS	
Tools Setup	*P18. Tools setup:* Problems in tools after deployment. Using the tools in practice and in different projects contexts allowed to identify undetected problems and necessary improvements.	R32: Tools are improved when used in practice, save time for their setup. R33: When correcting tools defects that have impact in the metrics definition and the process, do not use the collected data to build PPB.	CI	OPM SP2.2 OPM SP2.3
	P19. Overhead: in tools usage (data collection not completely automatic). *P20. Tools requirements:* New needs still being identified, new tools still being developed.	R34: Once PPM and PPB are stable only collect data that is needed. R35: Use automatic and unperceived data collection systems, with limited human intervention (start/stop and confirm). R36: Guarantee that data collection is precise (discipline people and change their mentality).	CI HP, J	

Figure 4.19: CMMI Implementation Checklist: list of activities to follow in order to avoid common problems.

When using EQualPI to support the implementation of CMMI, the organisation can load their processes and methods following the setup steps(4.4 Manage Configurations). After setting up the framework, the organisation will have the methods aligned with practices and performance indicators. The next step is to load the data. The processes may be unstable but the results shown by the framework will reflect the processes and data in place in the organisation at a given time. Therefore, as processes are changed so is the framework. The organisation will be able to see the evolution of their CMMI implementation throughout time.

4.5.3 MA Recommendations for High Maturity

We did an analysis of the reports on two SEI surveys (discussed in 3.2 Survey on MA Performance in HML Organisations) and their data (detailed in 5.2.1 Further analysis of the HML Survey Data) to find recommendations for process performance that were related with organisations achieving high maturity.

MA plays an important role on HML, including in the definition and use of PPM and PPB. We complemented the statistical analysis done by Goldenson et al. (2008); McCurley and Goldenson (2010), who analysed the relations between factors that contribute to see value in the PPM implemented, and focused on the factors that related with organisations achieving the desired CMMI goal. The recommendations are compiled in table 4.1.

Table 4.1: MA recommendations for HML (based on (Goldenson et al., 2008; McCurley and Goldenson, 2010; Lopes Margarido et al., 2013))

Purpose	**Recommendation**
Building Valuable Models	Put emphasis on "healthy ingredients"
	With purposes consistent with "healthy ingredients"
	Diversity of models to predict product quality
	Diversity of models to predict process performance
	Use statistical methods: regression analysis for prediction, analaysis of variance, SPC control charts, designs of experiments
	Data quality and integrity checks
	Use simulation or optimization methods: Monte Carlo simulation, discrete event simulation, Markov or Petri-Net models, probabilistic models, neural networks, optimization
Factors related with HML CMMI goal achievement	Have good documentation relative to process performance and quality measurement results
	Use simulation/optimisation techniques
	Have diverse methods of simulation/optimisation
	Have models with emphasis on "healthy ingredients"
	Have models for purposes consistent with "healthy ingredients"

Continued on next page

Table 4.1 – *Continued from previous page*

Purpose	Recommendation
	Substantially use statistical techniques
	Regularly use PPM in status and milestones reviews
	Managers must understand well PPM and PPB, which is related with the following two
	PPM and PPB creators must understand the PPM and PPB definition given by CMMI
	PPM and PPB creators must understand when PPM and PPB are useful
	Have experts available to work in PPM
	Distinguish missing data from zeros
	Check data precision and accuracy

4.5.4 Process Improvements

Doing a process improvement involves steps that are common regardless what the organisation intends to change but also have particularities. There are several methodologies that can be used (e.g. DMAIC, PDCA) and it is not our intention to provide a new one. As well, explaining how process improvements shall be carried is out of the scope of this thesis. Therefore, we stick to steps we followed in a process improvement experiment we did[1], and highlight important elements to consider while doing the process improvement. Our improvement was piloted with undergraduates and graduated students, and was later implemented in an HML organisation that is currently using it.

Step 1 - Identify a need and characterise the current process

The first step of process improvements is the identification of a need, for example: an organisation may want to achieve a business goal that cannot be attained in the current organisation setting, or the organisation may be facing a problem that needs to be solved as it is affecting business. - In our experiment the **need** was to reduce the number of defects from the requirements phase, only detected in posterior phases of the software development cycle.

When the need exists it is necessary to find why the need cannot be fulfilled in the current setting, the root cause of the problem or which processes can be changed to achieve the goal. - Our target process area was Requirements Management more specifically the process **Review Requirements**. In the case our process improvement, to detect defects more effectively in requirement reviews, we gathered a defect classification taxonomy, specific for requirements.

It is important to be able to measure improvements otherwise it is not possible to determine if they were beneficial or nefarious. It is necessary to determine the performance indicators that

[1] The experiment is detailed in 5.3 Requirements Process Improvement.

need to be monitored and create a baseline of those indicators. When EQualPI is used to monitor process improvements that task is eased because it is even possible to detect the effect of the improvement in other processes by monitoring other indicators. The baseline provides the state of the current process, *as is*, including the indicators values in that state. At this stage the improvement is characterised in terms of what is the need and the goal to achieve.

Step 2 - Identify and define improvement

It is necessary to determine the changes to implement, what methods will be used and what metrics can be used to monitor the changes. In any process improvement it is necessary to have selection criteria to determine the methods to use. In case of having a problem to solve whose origin is unknown one must select an adequate set of metrics to monitor and analyse them to determine the root cause of the problem and help define a solution for it. - In the case of piloting our improvement in an organisation, it would be necessary to understand the number of defects that were detected in posterior development phases that were due to requirements defects. Therefore, the metrics to monitor would be the defects per phase originated in the requirements phase.

Step 3 - Determine selection criteria and select improvement methods accordingly

This step is specific of what we did to design our improvement, and may only be applicable on similar ones.

In our case we had to assemble a classifiers list. We reviewed the literature to find what defects classifications were used and which ones where specific/more adequate to classify requirements defects. Another aspect we had to find were what recommendations there were on how to correctly define a classification list.

Freimut et al. (2005) indicated the quality properties of a good classification scheme, that we used as a reference while assembling the classification of requirements defects.

Step 4 - Set improvement goals and how to validate them

Organisations have to ensure that improvements are not only effective but also efficient so they must guarantee a return of investment (ROI). If our improvement was to be done in an organisation it would be necessary to analyse the costs of requirements reviews and of fixing defects in development phase. The current costs and goals to achieve would have to be documented. A simulator could be used to predict the ROI of the improvement, where the costs of training people and updating tools would also be considered.

In the organisation setting, to test our process improvement, the final number of defects per phase originated in the requirements phase would have to reduce to consider the improvement to be successful. In our setting, we had the goal of ensuring that the classification list was useful to classify requirements defects and classification by different people would be uniform. We designed our experiment to ensure people understood the list, the classifiers and defects were not confounded so different people would classify them the same way. The validation was done measuring the level of concordance of individuals when classifying the same defect. Formalising the hypothesis H, when reviewing requirements specifications:

H_0 - all subjects use the same value to classify the type of a defect.

H_1 - not all subjects use the same value to classify the type of a defect.

We did a Cochran test that is binomial, so we considered that when the subjects chose the most used classifier they answered as the **majority (1)** and when they used any other classifier, they chose **other (0)**. We used the Fleiss' kappa to measure the level of agreement between subjects to classify the same defect.

In any improvement that involves classification a similar experiment design can be used. In some cases, one way of evaluating people's adherence to process is using a survey.

Step 5 - Pilot the process improvement

To pilot improvements it is necessary to select projects for doing the improvement, include a control group that follows the current practices and projects using the new practices. The criteria to select pilots can vary, but here are some examples:

- Projects with effort and cost margin, to ensure the team has time to follow new practices without compromising the project's successful completion;
- Projects of short duration, to get results faster, in case the improvement does not require a given project size.

The teams need to receive training in the methods that are going to be applied. In the case of our experiment we gave the subjects instructions on how to participate in the experiment and included a table with the definition of each defect type and an example of how to use it.

Step 6 - Analyse pilot results

Analyse the results of the pilot to verify if the improvement actually occur, the impact on the indicators monitored and weather the improvement needs refinement and a new pilot or not. Repeat steps 2, 3, 4 and 5 as needed.

Step 7 - Prepare final version

When applicable

In our case would be update existing tools, include tool tips to help people remember definitions, include definitions and examples on help. Test tools in pilot projects, to test them and use in practice in order to refine as needed, train all teams.

Step 8 - Progressively deploy and control

Do the progressive deployment as indicated in 4.4 Manage Configurations, while controlling the indicators.

Chapter 5

EQualPI Validation

In this chapter we present the validation to the EQualPI and its components. We defined and validated the framework based on the literature reviews, using metamodelling and data modelling, conducting case studies, conducting data analysis on surveys and organisations' projects, building regression models using those data, and conducting quasi-experiements. In all our statistical analyses we considered a confidence interval of 95%, α or significance level of 0.05.

The model provided with EQualPI was generated through the analysis of SEI TSP data. The recommendations gathered to implement CMMI resulted of three case studies, conducted on organisations rated at CMMI level 5, a literature review and further analysis of data of a survey conducted by the SEI and involving organisations that were appraised at HML. The recommendations of process improvements are based on the researchers experience in the software engineering industry, experiments conducted with students and adoption of the improvement we designed and validated in an organisation.

5.1 Evaluation of the Estimation Process

Considering the Effort Estimation Evaluation Model (4.3.2 Effort Estimation Evaluation Model) included on the EQualPI module **Performance Indicators Models** the model is not an effort estimator. There are many models and simulators of effort or cost estimation in the literature (3.4 Related Research on Effort Estimation), and it is not our intention to develop a new one, but to develop a framework to evaluate the quality of the CMMI practices, demonstrated on PP SP1.4. To determine the quality of implementation of the practice "Estimate Effort" we used Effort Estimation Accuracy, defined by controllable and non-controllable factors. Similarly to the effort/cost estimation models or simulators, the data of these factors is used to determine the effort estimation accuracy. We summarise the reseach work done to build the Effort Estimation Evaluation Model in table 5.1.

In our research we reviewed TSP to identify performance indicators that could be applied in the characterisation of Effort Estimation Accuracy. Tamura (2009), provides the arguments to follow this approach and gives three examples of PPMs built with TSP data. TSP teams collect all

Table 5.1: Summary of information of the validation of the package Performance Indicators Models.

Label	Description
Motivation	Demonstrate the feasibility of the framework by implementing an indicator model to measure the quality of implementation of "Estimate Effort" defining the percentage of controllable and non-controllable factors. Define the framework data dictionary, including specific information for the model being implemented.
Method	Do a literature review on research that analyse effort estimation models to validate hypothesis, define the performance indicator to use, classify the factors used on effort estimation into controllable and non-controllable and design the experiment. Apply data analysis methods to build the regression model.
Results	Defined the data dictionary and respective data model. Identified additional recommendations on building tools to collect and store process execution data to avoid the difficulties we found. Produced a model to evaluate the quality of implementation of "Effort Estimation", and TSP models of "Actual Hours".

base measures necessary to control performance and understand the status of a project, and predict its course to completion. With other sources of information they can be used to design process performance models. Such measures, systematically collected by the team, give fine grained detail of size, defects, effort and schedule. The quality and reliability of the data is fundamental to build the PPMs needed for CMMI HML.

Jørgensen (2007), showed similar concerns to the one that leads us to evaluate the quality of the estimation process rather than developing another effort estimation model, questioning the "effect of issues of system dynamics on the meaningfulness of accuracy measurement, and problems related to the outcome-focus of the measurement, i.e., the fact that we are only evaluating the outcome of an estimation process and not the estimation process itself." He also points out problems related with the definition and interpretation of the term *effort estimates*. The people giving estimates may not know how they will be interpreted and estimates from analogy-based models may not be comparable with the ones from regression-based models. Since optimization functions can differ some models can systematically provide higher estimates than others.

5.1.1 Data Dictionary

The definition of the EQualPi's data dictionary was based not only in the effort estimation literature review but also in the analysis of the TSP framework, as it includes several metrics that allow to plan, manage and control the entire software development process.

In TSP, software development teams collect data for several metrics useful to fully characterise the software development process in terms of how the product was built, how effective it is and how efficient the development process was. Furthermore, by iterative planning development

and applying the PROBE method the information used in TSP to plan the activities necessary to develop the product highly contributes to more accurately define which functionalities the product will have, the activities necessary to build and verify it and the effort necessary to conduct all activities.

In our research we analysed the several sources of TSP data in order to define the variables that are considered and determine their source. We built a data dictionary defining all the variables and from it the TSP Database was built, which is a repository of a considerable large amount of data that was validated by the SEI.

The data dictionary, as it is based in TSP metrics, is useful for researchers that intend to analyse TSP data, due to the fact that it fully covers the software development lifecycle, and to TSP teams so they become aware of where the data are when they need to analyse them and fully benefit from them.

We developed a data dictionary to describe all TSP variables that are recorded by TSP teams. The data dictionary allowed the creation of the TSP Database where all TSP projects data that were provided by TSP practitioners to the SEI are stored. The use case of the database is presented in figure 5.1.

TSP data can be stored in the *TSP workbook* and *PSP workbook*, which are *Excel* documents that can be obtained here (http://www.sei.cmu.edu/tsp/tools/tspi-form.cfm), or using the *Dashboard*, which is a tool that can be obtained here (http://www.processdash.com/). Organisations can also have their own tools to store the TSP data so for those organisations the data dictionary columns **Primary Source** and **Secondary Source** work only as a reference to remind them of what variables we are referring to. The data can also be stored in the *TSP Launch*, *Team Survey*, *TSP Postmortem*, *Quality Plan*, *Training Records* or in *Meeting 1: Management Presentation*, for example. The **TSP User** collects, compiles and verifies the integrity of the TSP/PSP data. The data is analysed during the project by the team, on the progress and post-mortem meetings, and each team member analyses its own data. The data is reused to plan subsequent project cycles and projects.

The **TSP Database** includes data of projects from different organisations using TSP, who are the **Source** of data. **TSP Organisation**s can use the **TSP Database** data to compare their results and/or use it as historical data for estimation, in projects where they do not have their own data. Researchers in general can use the data not only to conduct their research but also to compare their results.

5.1.2 Data Extraction and Characterization

We received a list of projects, already with aggregation of plan and actual variables and validation of the individual effort records (through the Benford statistic). We also had access to the TSP database of workbooks, from where we extracted the data required by the data dictionary, including the one needed to analyse the estimation process. The data table that included the information we could "reverse analyse" to understand the estimation process used was related to the tasks. We extracted tasks with actual hours higher than 0, which had a size estimate. Even though, this

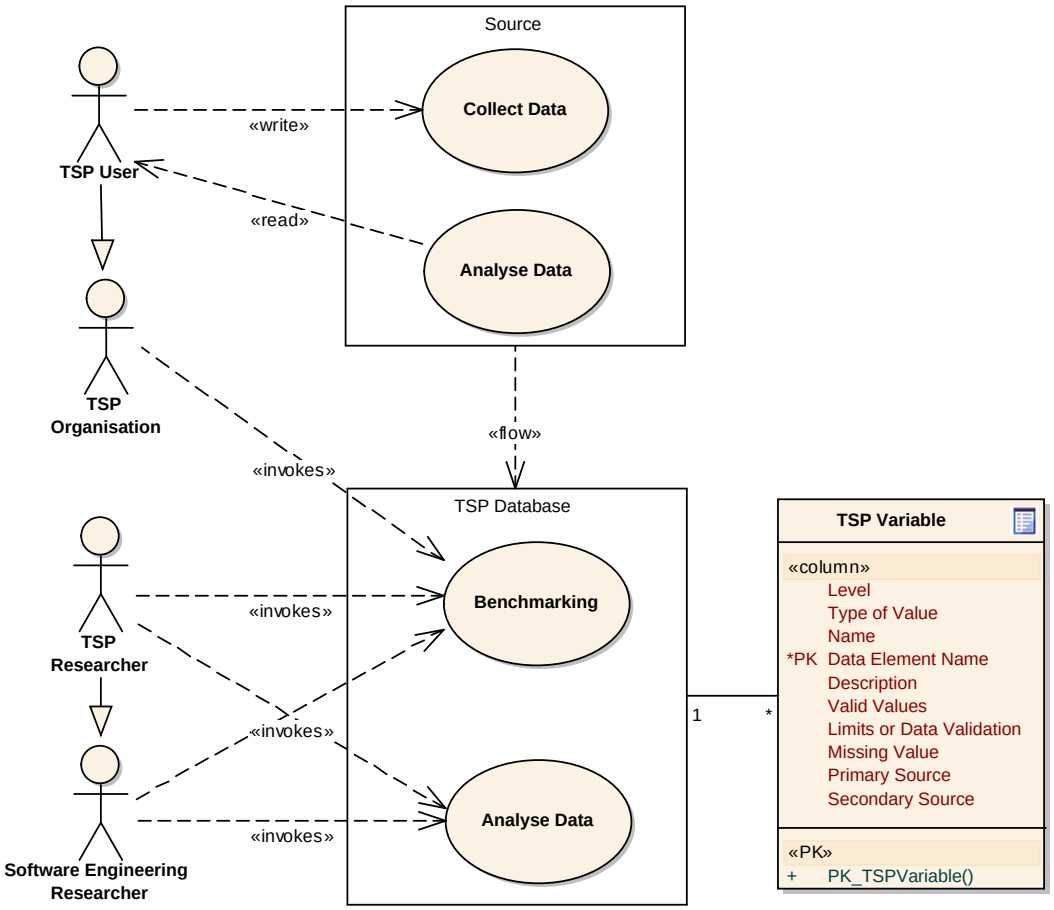


Figure 5.1: Use case of the TSP Database.

procedure introduces the researcher bias, we cannot assure that the reason for the time to be zero is overestimation or the workbook is incomplete. Thus, we only gathered data of completed tasks.

In our data analysis we wanted to use variables based on estimated, planned and actual values of effort, time in task measured in hours, size and defects in the dimensions number of detected defects and time spent fixing them. We observed that only two workbooks had planned injected defects [1], none of them planned injected defects in documentation (in TSP the considered documents are requirements, design, detailed design) and no workbook had estimated defects removed. Consequently, we did not consider defects estimation and respective effort in the model.

We tend to think that there are phases were defects are injected and phases where they are removed, when the filters are used. However, the database shows injected defects in phases that are of defects removal. This behaviour could be expected in testing phases, as it is common that solving a problem may inject a defect when developers do not consider all dependencies, but, interestingly enough, we also found defects injected in code inspections and reviews, where

[1] After cleaning duplicates there was just one project that estimated defects injected

normally we would not expect the injection of defects. We want to alert organisations to these facts, so they consider them when planning their projects. We noticed that teams had planned the defects injected per phase but did not plan the defects removed per phase.

5.1.3 Data Munging

The TSP Database had information of 257 workbooks, of which 234 had size estimates, while the consolidated projects list only had 114, four of which were not present in the database version we analysed. When checking for the data to guarantee uniqueness of projects there is no field that allows us to group all workbooks of the same project together and the ones in the consolidated list were classified with the ID of the workbook. We found several duplicates that we removed, finishing with a sample of 88 projects to analyse. Of those, six do not have any part measured in LOC, reducing the sample to 82 projects.

We removed duplicated workbooks just keeping the most recent version with more complete information, using the following check criteria to find similar data:

- Same number of defects or consistently increasing number of defects on the same phases;
- Defects with the same descriptions;
- Same increasing schedule, i.e. sharing start week and having increasing number of weeks, increasing actual/plan in last weeks hours, same actual/plan hours in the same completed weeks and same Plan Schedule;
- Same team members on tasks with the same descriptions.

We also noted 352 tasks with Plan Hours equal to 0. Of those, 38 were estimated, as they had estimated hours but the number of engineers was 0, hence, when multiplied the plan hours resulted in 0. A total of 348 tasks with 0 planned hours were did not use the estimate, which could mean they were not considered when the plan was consolidated. The other tasks without estimated hours were not estimated at all. They could be considered as estimated but not executed but being a small number and not having information to understand the reason for that to happen we removed those data points (0,5% of the sample).

After cleaning the data we ensured that was correctly classified, in the case of nominal variables, for that we needed clear factors. That was the case of the variables **Size Measure** and **Phase Name**, on both we found several values to describe the same factor, incorrectly increasing the number of factors. The Size Measure factors went from 75 to 28 levels, while the 30 levels of Phase factors were corrected and reduced to 23. In the case of the Size Measure we found several **Components** (or parts) with same ID and Size, only having size measure in some of the development phases, so we did the correction, when possible, analysing them case by case. Similarly, we corrected when possible missing values of other Phases that corresponded to parts of the same size.

5.1.4 Process Variables Definition and Data Aggregation

We defined the **Process Variables** based on the TSP planning and quality plan guidelines (Humphrey, 2006). Table 5.2 indicates the recommended values of the TSP Guidelines that we considered to define the Process Variables that are indicated on table 5.3 which also includes a brief description.

Table 5.2: TSP Planning and Quality Plan Guidelines (Humphrey, 2006) that we considered in our model.

Variable	**Guideline Value**
RequirementsInspection/Requirements	> 0.25
HighLevelDesignInspection/HighLevelDesign	> 0.5
DetailedDesignReview/DetailedDesign	> 0.5
DetailedDesign/Coding	> 1
CodeReview/Coding	> 0.5
Detailed Design	22.1%
Detailed Design Review	11.1%
Detailed Design Inspection	8.8%
Coding	20.0%
Code Review	10.0%
Compiling	3.4%
Code Inspection	8.8%
Unit Test	15.8%
Rate per Hour for New or Large modifications	10 LOC/hour
Small Changes to Large Systems	5 LOC per hour
Code Reviewed per Hour	< 200 LOC/hour

We defined the variables at task level, component level and project level. Not all variables are interpretable at task and component level, for example at task level that is the case of ratio variables, based on two development phases, because each task has just one phase. Hence, we built the model at the project level. The aggregation at component level was not possible by simply stating that a component or part has a unique ID, as we did to define the project ID, that after removing duplicates could be unequivocally identified. Therefore, we defined the component as the parts with same **ID** implemented in the same project, which have same **Size**, use the same **Size Measure** and are estimated at the same **Rate per Hour**. These conditions are necessary to determine the value of certain Process Variables that depend on the component total estimated, planned and actual times. Additionally, we need to consider that at task level many of the guidelines are defined as followed, 0, or not followed, 1, but when aggregated at component or project level they are determined as a percentage of tasks or components where the guideline was followed.

In the case of times, ratios and rate guidelines, if a project did not follow a guideline to estimate that may mean that they used their TSP historical data or such data did not exist for the particular project and the guidelines were not suitable in that context. Consequently, the estimates may have been based on an industry benchmark or expert judgement, as found to be more adequate for the particular project/team.

Table 5.3: Process Variables used to verify process compliance or determine the value of the planned metrics that define the process variable.

Variable	Description
Size Used	Percentage of tasks that used the size estimate to plan the hours spent in a task by multiplying the size by the estimated hours and number of engineers.
<Phase> Percentage	Considering all phases of the project, percentage of time spent on a given Phase (e.g. Requirements).
<Implementation Phase> Percentage	Considering only implementation phases, percentage of time planned to be spent in that particular phase (see implementation phases in 2.5 Effort Estimation in CMMI and TSP).
<Implementation Phase> Allocation Percentage	Considering only implementation phases, percentage of the allocation guideline recommended to be planned to spend in that particular phase.
<Defect Insertion Phase> / <Defect Removal Phase> Value	Ratio between the time planned for a reviewing or inspection phase and the corresponding defect insertion phase, e.g. between Code Inspections and Coding.
<Defect Insertion Phase> / <Defect Removal Phase> Followed	Percentage of tasks or components where the respective estimated defect insertion and removal time were planned following the guideline
Implementation Rate	Percentage of tasks or components that used the implementation rate guidelines
Review Rate	Percentage of tasks or components, with code review phases (Code Inspection or Code Review) that followed the recommended review rates.

5.1.5 TSP Estimation Model

To better understand the the effects of the variables resulting of the estimation process we built a model, that we called TSP Estimation Model, using them as independent variables. The first model was done to confirm that the variable Planned Hours is more correlated with Actual Hours than the variable Estimated Hours, because the plan already considers number of engineers, for example. The variable plan hours has a very high correlation with actual hours, in fact the model has an Adjusted R Square of 92,8%.

$$Actual = f(Plan): \quad ActHrs = \beta_0 + \beta_1 PlanHours \tag{5.1}$$

$$ActualHours: \quad ActHrs = 24,692 + 1,164 PlanHours \tag{5.2}$$

We built the regression model of actual hours, **Actual 1**, as a function of the data gathered to build the plan, showing the variables that for these data explain actual hours, i.e. variables of time, estimated using TSP. The model has an Adjusted R Square of 88,8%, and its equation is expressed

in 5.3.

$$ActualHours_1 = f(est.param.): \quad ActHrs = -222,122 + 3,317CodeInspTime + 1,048CodeTime + \\ + 1,465DLDTime + 24,496TeamSize + 9,248CompileTime \tag{5.3}$$

The estimates could not be done without considering the estimated size and implementation and review rates, or considering the time to execute similar tasks. Therefore, we include the parameters of estimated size and rate per hour used considering the TSP guidelines or using historical data, which still results on a significant regression model (significance of 0.000), but two of the added parameters are not significant (>= 0,05). Regardless, the Adjusted R Square of the model in equation 5.4 is slightly higher than the one that only considers significant variables, and still explains 89,1% of the variability with a 0,000 significance level.

$$ActualHours_2 = f(ad.est.param.): \quad ActHrs = 13,075 + 0,01PlanSize + 3,724CodeInspTime + \\ + 0,96CodeTime + 1,333DLDTime + 20,565TeamSize + 7,835CompileTime + \\ + 73,384ImplementationRate - 258,555ReviewRate \tag{5.4}$$

We present all the coefficients and respective significance of the Actual Hours models on 5.2. This indicates which variables of the plan actually explain 89% of the time of the actual hours, how accurate the plan was or if it was followed or not.

Model	Actual 1			Actual 2		
Variables	**Beta**	**Std. Error**	**Sig.**	**Beta**	**Std. Error**	**Sig.**
Intercept	**-221,122**	76,422	0,005	13,075	130,935	0,921
PlanSize				0,001	0,003	0,658
CodeInspTime	**3,317**	0,475	0,000	**3,724**	0,512	0,000
CodeTime	**1,048**	0,229	0,000	**0,960**	0,263	0,000
DLDTime	**1,465**	0,299	0,000	**1,333**	0,310	0,000
TeamSize	**24,496**	8,492	0,005	**20,565**	8,547	0,019
CompileTime	**9,248**	2,476	0,000	**7,835**	2,528	0,003
ImplementationRate				73,384	114,421	0,523
ReviewRateFollowed				**-258,555**	117,212	0,031

Figure 5.2: Actual models coefficients. The significant ones are signalled in bold.

The TSP model is already accurate as can be seen from the adjusted R Squares of equations 5.2, 5.3 and 5.4. However, there is still a deviation from the actual results, 11% that is not explained. The EEA model was developed help explain the deviation and know which variables should be considered to improve it. It can be used with two purposes:

- Evaluate the quality of estimation when planning a project by predicting the deviation from the actual that will be expected;

- Support the decision makers whether to act on the expected deviation or not, by acting over the model variables to reduce the EEA.

5.1.6 Effort Estimation Accuracy Model

As we were doing multivariate analysis we ensured that our continuous variables were standardised. For that reason we used percentages of time in phase (in the overall project and regarding the development phases), and the ratios between phases duration, similarly to the definitions of the TSP guidelines themselves. We studied the effects of the independent variables individually on the dependent variable EEA (4.3) and MER. With this we could anticipate which predictors would be part of the model. Nonetheless, we also took into consideration that, when analysed together, the effects of the predictors are different.

Figure 5.3 presents the histogram and the descriptive statistics of each dependent variable. None of them follows the normal distribution. The EEA is slightly positively skewed, while MER is much more positively skewed.

Our requirements for the model, being aware that the adjusted R Square could be low, were:

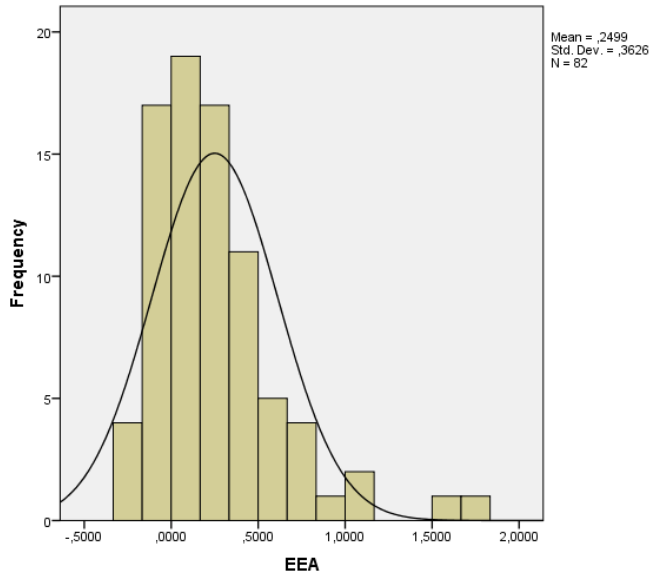
- Be a significant model (level of significance < 0.05);
- Only Have significant coefficients;
- Prevent model over-fitting by having a reasonable number of k coefficients in the model when compared to the number n of subjects.

Regarding the last rule k should follow the equation 5.5:

$$NumberofCoefficients: \quad k < \frac{n}{3} \;\; or \;\; k < \frac{n}{10} \tag{5.5}$$

The number of observations in our case was 82 projects, so keeping the number of regressors lower or equal to 8 allows us to respect the rule to prevent over-fitting .

We decided not to do any transformations to the variables to improve precision because the goal of the research was to provide additional information that can be interpreted and explained and ready to use by practitioners. Firstly we tested a model considering all the guidelines variables using SPSS Automatic Linear Modelling using the *Forward Stepwise* method, using the *Information Criterion* for entry/removal, to select the variables that are significant to the model and have an effect in the dependent variable. In the automatic model generation SPSS trims outliers, replaces missing values and transforms the variables, and therefore we just used it to guide us in the right variables selection. Next we did a linear regression, considering only the variables considered on the last step of the model, using the *Enter* regression method and *Pairwise* cases to handle missing values. We opted to use Enter because we wanted to check the individual contribution of the variables that, together, improve the prediction of the dependent variable and if any of them were not significant to the model we would removed a posterior execution. We selected the *Pairwise* method instead of the *Listwise* since only one project had data for all the variables. Moreover,

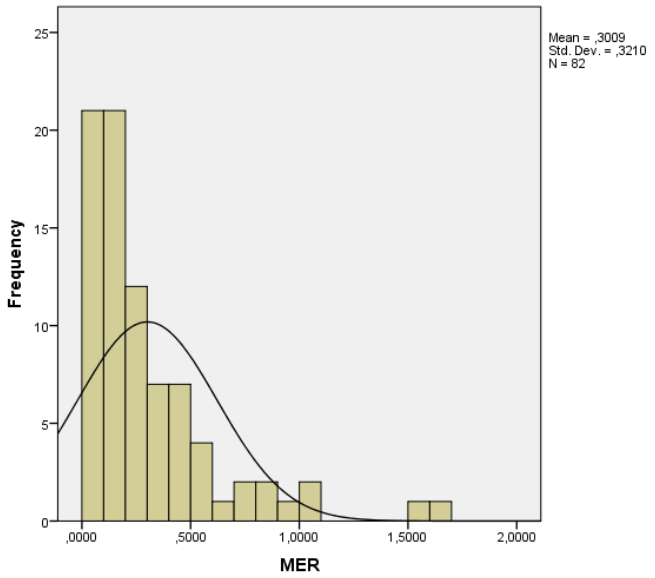


Statistics

EEA

N	Valid	82
	Missing	0
Mean		,249894
Median		,177700
Mode		-,2363[a]
Std. Deviation		,3626194
Skewness		1,622
Std. Error of Skewness		,266
Kurtosis		3,614
Std. Error of Kurtosis		,526
Minimum		-,2363
Maximum		1,6943
Percentiles	25	-,010050
	50	,177700
	75	,414675

a. Multiple modes exist. The smallest value is shown



Statistics

MER

N	Valid	82
	Missing	0
Mean		,300921
Median		,191400
Mode		,0067[a]
Std. Deviation		,3209977
Skewness		2,183
Std. Error of Skewness		,266
Kurtosis		5,768
Std. Error of Kurtosis		,526
Minimum		,0067
Maximum		1,6943
Percentiles	25	,091600
	50	,191400
	75	,414675

a. Multiple modes exist. The smallest value is shown

Figure 5.3: EEA and MER histograms and statistics. The upper graph and table refers to EEA and the lower to MER.

we considered inadequate to replace missing values by means or medians as the missing cases could either mean that the projects did not use those phases, or the workbook in the TSP database was incomplete, e.g. corresponding to an intermediate cycle of the project. The procedure lead to a statistically significant model with adjusted R Square of 0,295 and 4 coefficients, all of them significant as well. This means that the model explains 29,5% of the variability of the data not by chance.

Even though the automatic regression model identified outliers (see figure 5.4), we decided not to remove them since we did not have context to explain those cases. If we removed them the

Adjusted R Square of the model would increase but that could also result in model overfit, given the reduced number of projects and their diversity.

Record ID	MRE	Cook's Distance
80	1,6943	0,464
35	0,4788	0,213
8	1,5174	0,167
73	0,1289	0,098
68	0,1190	0,073
72	0,4251	0,055

Records with large Cook's distance values are highly influential in the model computations. Such records may distort the model accuracy.

Figure 5.4: EEA and MER models outliers.

The significance level and coefficients of the variables used on each model are presented on the table in figure 5.5. The ones marked in bold are significant.

The model for the Effort Estimation Accuracy, measured with EEA is the one on equation 5.6.

$$EffortEstimationAccuracy: \quad EEA = \beta_0 + \beta_1 DLDRPerc + \beta_2 CODEINSPPercDev + \\ + \beta_3 CRCODEVal + \beta_4 CMMI + \varepsilon \tag{5.6}$$

β_0 is the intercept, CMMI could also be considered a factor model that varies from 1 to 5, even though 4 is not represented in the sample data that we analysed. A change of 1 unit on each of the coefficients individually represents a change of β_n units in the Effort Estimation Accuracy. "The error term ε represents all sources of unmeasured and unmodelled random variation (Leek2013)" in the Effort Estimation Accuracy. With the data we used, β_n assume the values indicated on the regression model in equation 5.7.

$$EEA = 0,251 - 0,032 \times DLDRPerc - 0,019 \times CODEINSPPercDev + \\ + 0,683 \times CRCODEVal + 0,142 \times CMMI + \varepsilon \tag{5.7}$$

The model can be used by practitioners to evaluate the expected accuracy of their effort estimation process, and improve it by varying the coefficients. Knowing that a variation of 1 unit in the percentage of total project time spent reviewing the Detailed Design will cause a decrease in the EEA of around 0,251, if the variation is on the percentage of time spent on the development phases to do Code Inspections will cause a decrease of aorund -0,019 on EEA. The effect of varying one unit in the ratio between time spent on code reviews and time spent coding will increase

the EEA in 0,683 and the level of CMMI increases the EEA in 0,142.

We did a regression model for MER as well, because from the indicators commonly used in Software Engineering when comparing estimation models(see 3.4 Related Research on Effort Estimation), MER measures the inaccuracy relative to the estimate (Foss et al., 2003). Both dependent variables are modelled by the same coefficients but with different magnitude. The Adjusted R Square of MER is slightly higher, 31,8%. The MER Model is presented on equation 5.8.

$$MagnitudeofErrorRelative: \quad MER = \beta_0 + \beta_1 DLDRPerc + \beta_2 CODEINSPPercDev + \\ + \beta_3 CRCODEVal + \beta_4 CMMI + \varepsilon \tag{5.8}$$

$$MER = 0,386 - 0,031 \times DLDRPerc - 0,020 \times CODEINSPPercDev + \\ + 0,560 \times CRCODEVal + 0,118 \times CMMI + \varepsilon \tag{5.9}$$

A variation of 1 unit in the percentage of total project time spent reviewing the Detailed Design will cause a decrease in the MER of around 0,031, if the variation is on the percentage of time spent on the development phases to do Code Inspections will cause a decrease of around -0,02 on MER. The effect of varying one unit in the ratio between time spent on code reviews and time spent coding will increase the MER in 0,560 and the level of CMMI increases the MER in 0,118.

On figure 5.5 we present the coefficients data of the EEA and MER models. The standard error of the coefficients is lower on MER. All our models, have less than 8 variables, respecting the rule to avoid overfit.

Model	EEA			MER		
Variables	**Beta**	**Std. Error**	**Sig.**	**Beta**	**Std. Error**	**Sig.**
Intercept	0,251	0,181	0,171	0,386	0,158	0,018
CMMI	0,142	0,046	0,003	0,118	0,040	0,005
DLDRPerc	-0,032	0,012	0,010	-0,031	0,011	0,005
CodeInspPercDev	-0,019	0,009	0,034	-0,020	0,007	0,011
CRCodeVal	0,683	0,302	0,028	0,560	0,263	0,038

Figure 5.5: EEA and MER coefficients: Beta, Standard Error and Significance

Both models are summarised on table 5.4. The respective ANOVA can be found on table 5.5. The MER regression model explains more variability than the EEA. We used the Durbin-Watson to test the null hypothesis, that there is no autocorrelation between the residuals. Both models present a value higher than the upper Durbin-Watson Statistic for models with $K = 4$ regressors and $n = 82$ subjects (> 1,743) and the test statistic is close but still lower than 2, so the residuals are not auto-correlated.

We had very few projects of organisations appraised at CMMI, none of them at level 4. We noticed that the organisations with CMMI level 5 had worse EEA and MER. That fact cannot

be explained with the data we had but perhaps those organisations were just starting to use TSP, having no TSP historical data yet.

Table 5.4: EEA and MER Models summaries.

Parameter	**EEA**	**MER**
R (Person's coef.)	0,586	0,604
R Square	0,343	0,364
Adjusted R Sq.	0,295	0,318
Est. Std. Error	0,304	0,265
Durbin Watson	1,880	1,970

Table 5.5: EEA and MER ANOVA

	Parameter	**EEA**	**MER**
Regession	Sum of Squares	2,663	2,216
	Degrees of Freedom	4	4
	Mean Square	0,666	0,554
	F Statistic	7,185	7,885
	Significance	0,000	0,000
Residual	Sum of Squares	5,095	3,864
	Degrees of Freedom	55	55
	Mean Square	0,093	0,070
Total	Sum of Squares	7,758	6,079
	Degrees of Freedom	59	59

5.1.7 Cross Validation of the Standard Error

Considering the already small number of projects and the fact that not all of them have values for all variables we decided to use the entire data set to train the model and estimate its error using cross validation of the models instead of using a training and test set, i.e. using part of the sample to generate the regression model and the test set to verify its prediction error.

We created k-fold where k was 4, resulting in 4 samples, one of them with 22 projects and the remainder 3 with 20. Then, we used the Standard Deviation (3.8) of the residual error to determine the mean error of the 4 folds. The results are summarised in table.

Even though the MER model is more accurate, having lower error and explaining a higher percentage of the data variability, we opted to display both of them because if the practitioner wants to distinguish over and underestimation the information is available on the EEA model.

Table 5.6: 4-fold cross validation of the standard error of the estimates of the models EEA and MER.

Test Set	EEA	MER	Nr. Projs.
Sample 1	0,2903	0,2460	22
Sample 2	0,4336	0,3965	20
Sample 3	0,3162	0,2703	20
Sample 4	0,3144	0,2535	20
Mean SD	0,3386	0,2916	***Total*:** 82

5.1.8 Limits to Generalisation and Dataset Improvements

The models we built would be more complete if we had a sample with more projects where the different TSP planning procedures had been followed and all phases were included. When checking the variables correlations we noticed that Requirements, Requirements Inspections, High Level Design and High Level Inspections and the corresponding Process Variables would be relevant to this research. Furthermore, even though we are aware that it is harder for people to accurately estimate injected and removed defects (especially injected) the model could have considered such estimates, if we had the data. However, just having a single data point does not allow us to reach any conclusions. These variables should be considered because when executing the project the actual defects have influence on the quality of the product and require time to be fixed, affecting the actual hours. If the data is available, new Process Variables should be designed based on phase yields, percentage defect free, defect density, defect injection and removal rates, as defined by Humphrey (2006).

We introduced researcher bias when extracted only projects with plan and actual zero, however some tasks may have not been planned and still have been added to the records and executed and some tasks may have been planned but may have not been completed. This decision was consciously taken as the status of the workbooks may not have been the final (end of project), therefore we could not be certain of the reasons for them to be 0 and whether they should be kept or not.

We found several missing values and different case/naming for the same values that could have been avoided by providing dropdown lists, auto-complete and some automation. For project execution purposes it may not have been a problem as engineers already knew the meaning of the information, but the analysis of the workbooks for Quality reviews and now for research purposes could benefit from ensuring mandatory information and auto-complete whenever possible. It is important to use uniform and phase names and size measures to ensure they are not named similarly, affecting posterior data analysis. It is also important to ensure people understand the definition of the phase name and size measure value to avoid the creation of new ones due to lack of understanding.

We also reviewed the workbooks to identify those that were a different version of the same project. Currently there is no workbook versioning in the database, each version gets a new work-

book ID. It is important to ensure the database has a versioning mechanism of workbooks, so when an existent one is updated, instead of receiving a new ID it gets a new version, time stamped, that allows sequencing sequencing. So researchers can choose to analyse the same project over time or only analyse the latest version.

5.2 CMMI HML Implementation

This section refers to the validation of the CMMI implementation procedures presented on 4.5.2 - CMMI Implementation. The purpose of this research was to identify problems and challenges that the organisations face when implementing CMMI, in particular ML 4 and 5, find recommendations to help implementing CMMI to avoid those problems and identify factors that are relevant to achieve HML, in the particular helpful to define PPM and PPB. We summarize this research work in table 5.7.

Table 5.7: Summary of information of the validation of the package CMMI Implementaion.

Label	Description
Motivation	Not all organisations using CMMI achieve the same gains in performance. Some results demonstrate that organisations with other process improvement initiatives can perform better in terms of quality. SCAMPI already missed problems of lack of understanding of the statistical nature of ML 4 and lack of institutionalisation of practices.
Method	Do a literature review to find problems in the implementation of CMMI, metrics programs and process improvements. Conduct a cases studies in organisations appraised at CMMI level 5. Conduct an extended SCAMPI with further documents/tools analysis and interviews. Analyse the data of SEI surveys conducted on organisations appraised at HML.
Results	Identification of problems and difficulties in implementing CMMI. Identification of weaknesses of SCAMPI that contribute to the failure in the detection of problematic implementations. Built recommendations that guide organisations in the implementation of CMMI in order to avoid HML implementation problems.

We conducted three case studies in multinational organisations that develop software (CI, CII and CIII) assessed at CMMI for Development ML 5, staged representation. Case studies on CI and CIII were conducted immediately after the appraisal. Our purpose was to identify real problems and difficulties in the implementation of CMMI and find recommendations to avoid them. We mainly focused on MA and HML, but also analysed other CMMI PAs. The research questions we intended to answer, which we considered when designing the case studies and analysing all data, were the following:

- What was the strategy to evolve to the new ML?
- What difficulties and problems occurred in the implementation of the new practices?

- What is the process definition?
- How was the process defined?
- How are people using the process?
- How are people collecting, analysing and interpreting process data?
- What is the impact of the new process on people's work?

The results of these case studies, the literature review that we did to find the problems in CMMI (see 3.1.3Problems in Process Improvements, Metrics Programs and CMMI), metrics programs and process improvements and the analysis of two surveys that the SEI conducted in HML organisations allowed us to build and validate the CMMI implementation checklist (see 4.5.2 CMMI Implementation). In this section we signal the problems found in each organisation with CI, CII, CIIG or CIII, respectively and the ones found in the analysis of the survey data with SDA. The problems (P) and recommendations (R) are numbered.

A key problem is that measuring organisations performance is outside SCAMPI scope. The assessment verifies if the techniques applied allow achieving CMMI goals (Masters et al., 2007). There are only two PAs where performance improvements are explicitly analysed: CAR, where the effect of implemented actions on process performance should be evaluated; and OPM, where the selection and deployment of incremental and innovative improvements should be analysed (CMU/SEI, 2008).

5.2.1 Further analysis of the HML Survey Data

We analysed the data of the 2009 survey to sustain problems and recommendations with empirical evidence. Regarding the recognition of the efforts of people involved in MA initiatives, even if the difference is not very significant, the graph (Figure 5.6) shows that a larger number of organisations that achieved HML did give promotions or monetary incentives. The difference seems to be more relevant when they were given to Project Engineers, Technical Staff and Project Managers, i.e. people working closer to the projects.

We used the statistics described on the table in figure 5.7, to understand the relation between achieving HML (V1) and doing a given practice. We tested the dependency of several variables and compared the groups of organisations that achieved HML and the ones that did not. On the table in figure 5.8 we report the tests results of dependent variables (p-value < 0.05 in the Chi-square test) with different central tendency between groups (p-value < 0.05 in the Mann-Whitney test).

We verified that there is a relation between the understanding that the creators of PPM have of the CMMI intent (V4 to V7) and achieving HML. Figure 5.9 shows that a bigger percentage of organisations that achieved HML understood very well or extremely well the intent of CMMI. In both surveys the relation between understanding results of PPM and PPB, by Managers who use

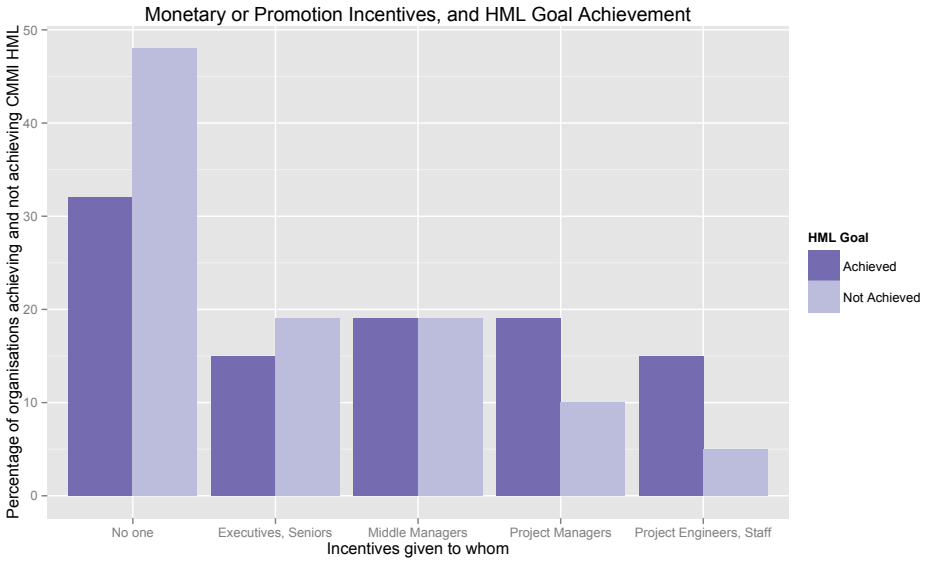


Figure 5.6: Relation between giving incentives to people who use use and improve MA, and the achievement of the HML goal (Lopes Margarido et al., 2013).

Test	Description	Hypotheses
Levene	**Purpose:** Verify conditions to use the Mann-Whitney U test. The samples must have the same variance.	H_0 *(Null Hypothesis) – the two samples (Achieved and Not Achieved) have the same variance*
	If p-value < 0.05 we can reject the null hypothesis: the variance of the two samples are different. If p-value >= 0.05 keep the null hypothesis: the variances of the two samples are the same.	H_1 *(Alternative Hypothesis) – the two samples have different variance*
Mann-Whitney	**Purpose:** Verify if the two samples have similar median.	H_0 *– the two samples have similar median*
	If p-value < 0.05 we can reject the null hypothesis: the median of the two groups is different. If p-value >= 0.05 there is no difference between the groups.	H_1 *– median of Achieved > median of Not Achieved*
Chi-square	**Purpose:** Verify dependency between two variables.	H_0 *– the variables are independent*
	If p-value < 0.05 we can reject the null hypothesis: the variables are dependent. If p-value >= 0.05 the variables are independent.	H_1 *– the variables are dependent*

Figure 5.7: Statistics and hypotheses that were tested.

them (V2), and the achievement of HML was quite strong (Figure 5.10). We observed a relation between managers that understand the results better (V2) and creators that understand better the CMMI intent (V4 to V7) (Figure 5.11). A similar behaviour was verified with the availability of PPM experts to work (V8) (Figure 5.12).

The graphs in Figure 5.13 indicate that integrity data checks also seem to be related with the achievement of HML. The statistical tests support that distinguishing missing data from zeros and checking data precision and accuracy are related with achieving HML.

With the analysis that we did we found that the following variables are related with the achievement of the desired HML:

- How well managers understand PPM and PPB;
- How well PPM and PPB creators understand the CMMI intent;
- Distinguish missing data from zeros;

Variables	Levene	Mann-Whitney U	Chi-Square
V1: CMMI HML goal Achievement V2: How well managers understand PPM and PPB results	F = 0.0399 p-value=0.8423	W = 200.5 p-value = 1.44e-05	X-square = 20.64731 p-value = 0.0003719468
V1 V3: PPM and PPB creators understand the definition of PPM given by CMMI	F = 2.1284 p-value = 0.1489	W = 875 p-value = 7.75e-05	X-square = 3.271229 p-value = 0.0003911981
V1 V4: PPM and PPB creators understanding of the definition of PPM given by CMMI	F = 1.4462 p-value = 0.2333	W = 875 p-value = 7.75e-05	X-square = 20.53658 p-value = 0.0003911981
V1 V5: PPM and PPB creators understand the definition of PPB given by CMMI	F = 0.0484 p-value = 0.8264	W = 902.5 p-value = 1.711e-05	X-square = 19.64407 p-value = 0.0005870212
V1 V6: PPM and PPB creators understand when PPM are useful	F = 0.0082 p-value = 0.9281	W = 920 p-value = 7.735e-06	X-Square = 23.23521 p-value = 0.000113636
V1 V7: PPM and PPB creators understand when PPB are useful	F = 0.1445 p-value = 0.7051	W = 931.5 p-value = 4.198e-06	X-square = 24.84557 p-value = 5.403748e-05
V2 V4,5,6,7	F = 3.1665 p-value = 0.07963	W = 576.5 p-value = 0.02413	X-Square = 9.809072 p = 0.0202608
V2 V8: Availability of experts to work in PPM	F = 0.0095 p-value = 0.9227	W = 163 p-value = 2.094e-05	X- Square = 23.21129 p-value = 0.000114894
V1 V9: Distinguishing missing data from zeros	F = 0.8992 p-value = 0.3463	W = 855.5 p-value = 2.249e-05	X-square = 16.1262 p-value = 5.926e-05
V1 V10: Checking data precision and accuracy	F = 2.5641 p-value = 0.1139	W = 761 p-value = 0.00389	X-square = 7.0344 p-value = 0.007996

Figure 5.8: HML 2009 survey – further data analysis. Results of the tests done with the groups of organisations that achieved and did not achieve HML and that were shown to have the same variance through the Levene test.

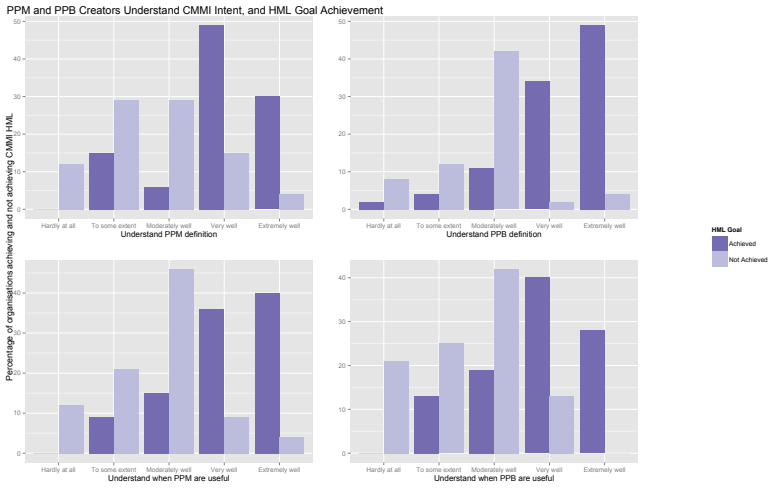


Figure 5.9: Relation between understanding the CMMI intent with PPM and PPB by their creators, and the achievement of the HML goal.

- Check data precision and accuracy.

These relations reinforce the importance of doing proper integrity data checks to have meaningful and reliable PPM and PPB supporting high maturity PAs. Furthermore, to ensure that managers understand PPM and PPB results correctly, and consequently take appropriate actions, PPM and PPB creators must understand their CMMI meaning and usefulness, and experts must be available.

5.2.2 Case Studies

In CI we interviewed the CMMI programme sponsor, posing direct questions and an open-end question to which the interviewee answered by narrating the story of the program. We had a

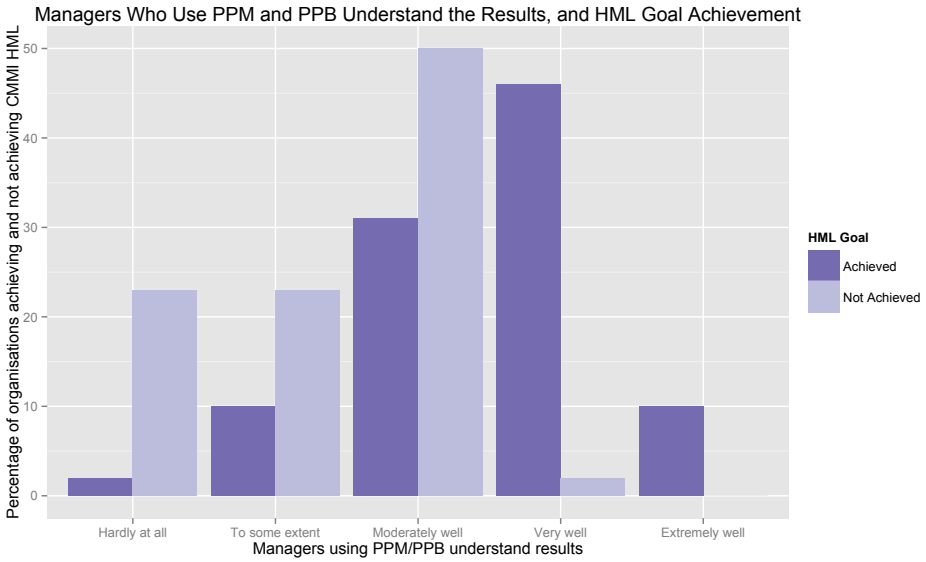


Figure 5.10: Relation between managers who use PPM and PPB understanding the obtained results, and the achievement of the HML goal.

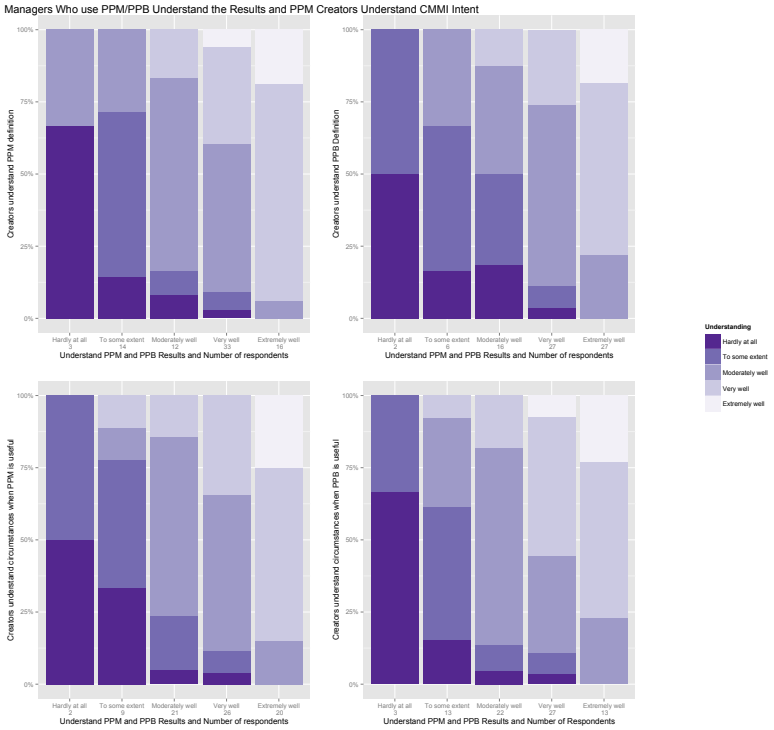


Figure 5.11: Relation between PPM and PPB creators understanding the CMMI intent and managers who use them understanding their results.

similar interview with the program responsible. In both interviews we identified interviewees, projects and other documentation to analyse. We analysed the company Quality Management

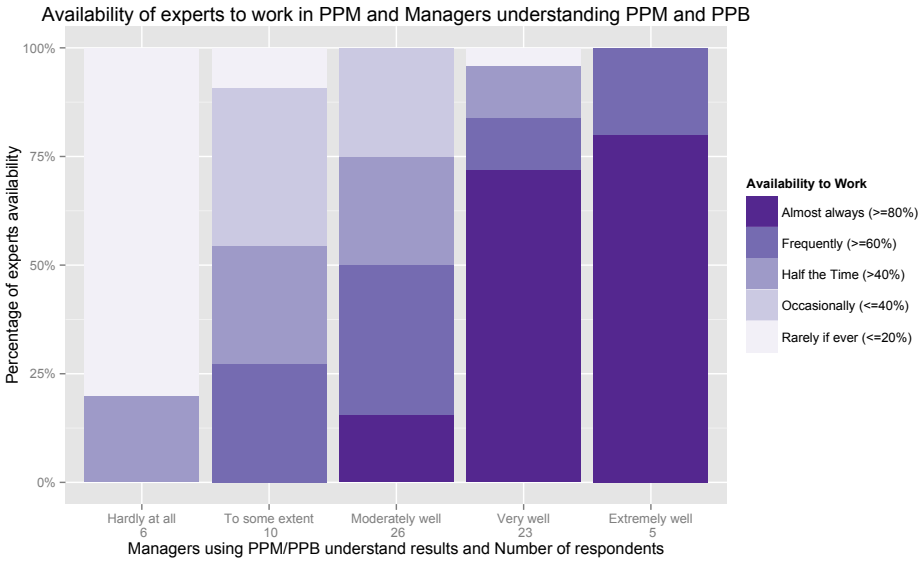


Figure 5.12: Relation between the availability of experts to work in PPM and managers who use them understanding their results.

System (QMS), Information System and SCAMPI A repository. We also interviewed practices and tools implementers, and project teams, including the appraised ones (whose documentation we analysed). Analysing CI data we found that the main problem stemmed from rapidly evolving to ML5 without giving enough time to have stable tools, processes, PPB and people behaviour.

CII is a business unit, located in several countries, that is part of CIIG, a CMMI level 5 organisation. In CII we interviewed the responsible for the CMMI programme, beginning with directed questions and finishing with descriptive questions regarding the story of the program. Afterwards we analysed CIIG QMS and Information System. Finally, we had a meeting with the CMMI program responsible and discussed our results and conclusions. In CII we found several problems related to metrics; most limitations came from the fact that size was not being measured, and time spent on tasks stopped being accurately collected. Many of the identified problems were originated by the resistance to change and difficulty in presenting, to CIIG, metrics adequate for CII.

In CIII we interviewed a consultant involved in the appraisal of the organisation, i.e. a person who performed an actual observation on the case. The main difficulty faced by CIII was to move to statistical thinking. The problems found on the organisations are indicated on the next paragraphs.

P1. Underestimate time to implement HML

We found this problem in CI, CIII and in the SDA. CI re-planned the CMMI implementation programme several times until Six Sigma was introduced, allowing them to better understand HML demands.

In CIII implementation turned out to be more complex than anticipated and the programme took longer than planned.

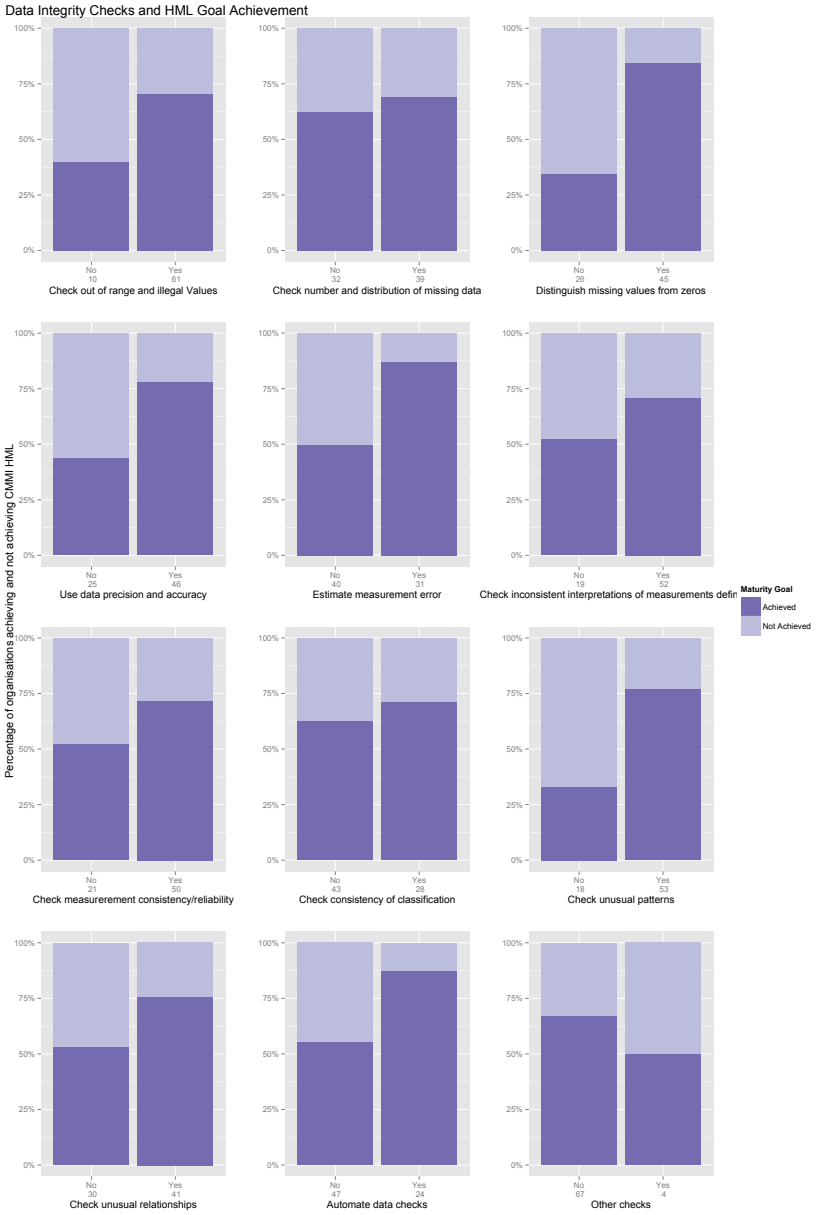


Figure 5.13: Relation between performing data integrity checks and the achievement of the CMMI HML goal.

P2. Introduction of HML forgetting ML 2 and 3

This problem was also identified by Leeson (2009). It occurs often and CIII was no exception.

P3. Understand the statistical nature of level 4

This problem was reported in the literature by Hollenbach and Smith (2002) and Takara et al. (2007) and found in CI, CIII and SDA.

Move to statistical thinking and quantitative management was the main challenge CI challenge faced by CI.

Changing mentality to HML was a significant shift for CIII, because preparing and using the quantitative component takes time to mature. This problem may have been one of the causes of P1.

P4. Copied processes

We found some processes in the CIIG QMS that were copies of CMMI, not reflecting the organisation's culture. That may have been one of the reasons for the problem P5. Multicultural environment.

P5. Multicultural environment and **P6. Impose processes**

CIIG acquired other companies and imposed its processes on them; consequently, their good practices, certain metrics and good visibility of processes were often lost. Another problem was that people from different cultures have different ways of working: in certain cultures orders are taken without question, while in others people need to understand the benefits of working in a certain way, otherwise they will resist change.

P7. Dissemination problems

This problem happened in CI and SDA.

In CI people noticed communication improvements. Nonetheless, the dissemination of information regarding processes and tools usage was not totally effective: some people still had difficulties to apply the new practices.

P8. Lack of institutionalisation

This problem was reported in the literature by several authors: Radice (2000); Schaeffer (2004); Charette et al. (2004) and Hamon and Pinette (2010). The problem occurred in both CI and CII.

In CI not all project teams were applying the new practices. This problem was also related to people behaviour and resistance to change.

In CIIG not all projects and business units performed at the same maturity level.

P9. Meaningless uncorrelated metrics

This problem was analysed by Kitchenham et al. (2006) and occurred in CI and SDA.

In CI we found a case of metrics being misinterpreted due to lack of understanding of the context of one business area.

P10. Metrics definition (collect and analyse data)

This problem was reported by several authors: Goulão (2008); Hamon and Pinette (2010) and Barcellos (2009). It occurred in SDA, CI and CII.

In CI people still had difficulties in collecting data in certain contexts and in their interpretation.

CIIG imposed KLOC (thousand lines of code) as the applicable size metric, which CII did not consider adequate to its types of projects.

P11. First data collected were uncorrelated

This problem was also identified by Hamon and Pinette (2010) (HP)

It occurred in CIII at start, which implied conducting new data collection cycles and new searches for correlations. This may have been one of the causes of P1.

P12. Metrics categorisation

This problem was found in SDA, CI and CII.

In CI data for high maturity had been collected for a short period of time, so the baselines were not stable enough. It was not possible to distinguish between different categories of data (for different markets, team experience, team sizes and project sizes), therefore the data were compiled in PPB categorised by technology only.

As CII could not use KLOC to measure size they could not use many derived measures.

P13. Baselines not applicable to all projects

This was a problem common to CI and CII.

In CI PPBs were still unstable, or inadequate for all types of projects. Time to collect data was insufficient to gather information of different contexts and verify: 1) if new metrics were needed; 2) if there were differences in performance and in which contexts; 3) if in certain circumstances the procedure to collect the data should be different.

CIIG had centralised PPBs not applicable to all business units' realities and projects, e.g., the development lifecycle phases had different durations depending on the business. Another problem was that productivity was measured considering a business day as unit of time but in some locations it had different duration in number of hours.

P14. Abusive elimination of outliers

In CI we found one situation when it was not perceived that the outlier occurred at least once in some projects. Some outliers are just indicators that the process improved its performance (Spirula member, 2010 personal communication).

P15. Not all projects are measurable

This problem occured in SDA, CI and CII.

In CI tools were not yet prepared to collect data in certain projects (maintenance, with several phases or outsourced), because their data structure was different from the standard projects (development). Besides, measurements specific to maintenance and outsource projects were not defined.

P16. Effort estimates

While in CIIG effort estimation was based on their historical data of effort and size, CII estimates were based on expert judgment without using any tools or models.

P.17 People behaviour

This problem happened in CI, CII and SDA.

In CI changing mentality was a challenge; some people did not see value in new practices or stated that they were not applicable to their projects. It is difficult to convince people to report effort accurately: they normally report contract hours, not real effort, or do not report effort as they are finishing tasks, leaving the reporting until later.

CII workers stopped reporting effort accurately, only reporting contractual hours of work.

P18. Tools setup and **P20. Tools requirements**

In CI the existing Information System evolved to support the new practices, but people were still detecting problems and requesting improvements, in tools and processes, resultant of using the tools in practice and in different projects contexts.

P19. Overhead

This problem happened in CI. To record time spent on tasks people manually filled a form per task and the data collection of the new metrics was only partially automated. This was also due to the insufficient Tools setup time, to understand all needs the usage of the tools.

The demands of levels 2 and 3 should prepare organisations to adequately use measurement at higher levels, by monitoring appropriate metrics. Nonetheless, some of the problems here identified reflect a poor implementation of the MA PA, affecting the organisations results. Such problems become evident when implementing ML 4 because the correlation between variables and problems in the collected data, affect PPM and PPB. Besides, SCAMPI cannot appraise the entire organisation and does not analyse performance measures – if it did, it would become even more expensive. Hence, CMMI rating per se is not a guarantee of achieving expected performance results and organisations need to be aware that there are different methods that can be used on its implementation. Nevertheless, if some recommendations such as the ones we proposed in 4.5.2CMMI Implementation are followed, CMMI implementation can be easier, and the problems discussed before can be avoided. Most of these recommendations are solutions used by the studied organisations to overcome their problems.

Entry Conditions

R1: We used SDA to show that having PPM experts available to work is related with understanding PPM and PPB results by managers using them. In CIII the implementation plan was long and all activities needed to be executed on the estimated time. At first, they considered that if an

activity had overrun the schedule, time could be recovered by shortening others. The thought was abandoned once they realised that time could not be shortened in any task.

R2: CI analysed gaps to address problems in lower maturity levels; however, those processes were affected by changes to implement HML and there should have been a new cycle for them to mature. In CIII the move from ML3 to ML5 was uninterrupted, so ML3 matured and did not erode in the meantime.

R3: SDA showed relations between PPM and PPB creators understanding CMMI and organisations achieving HML.

R4: In CI, Six Sigma helped to gain insight of information needs to achieve quantitative goals, solve problems and design PPB and PPM.

R3 - R5: were also followed by CI and CIII, as they were part of the CMMI implementation process.

Process Definition and Implementation

R6 and R7: were followed by CI and CIII by first understanding the existent process, identifying gaps and involving internal experts and users in the definition of improvements and new processes.

R8: CII applied R8 in a Direction with specific needs, i.e., analysed other business units metrics in order to adopt the ones that could be applicable to their projects lifecycle.

R9 and R10: helped CI maintain the visibility of processes and projects at different organisation levels and were part of the CMMI implementation process in CIII.

R11 - R13: were used by both CI and CIII. Regarding the recommendation of *training on benefits and how can they be seen (R12)* we cannot be sure it was effective on any of them. Additionally, in CI the coach corrected people's mistakes instead of guiding them.

R14: was used by CI and CIII, the dissemination of processes was gradual, as they were ready to be deployed directly from pilot projects to the entire organisation. However, when organisations are large they should consider even more gradual dissemination, spreading practices in a small group of projects and gradually involving new ones, which can be done also profiting from team members mobility. SDA showed that the incentives to people improving and working in MA were more frequent in organisations that achieved HML.

R15: was used by CIII.

Metrics Definition

R16: was used by both CI and CIII so there was a clear view of which metrics were used to monitor different levels of goals and what was their definition.

R17: was also followed by CI but definitions needed to mature to ensure unambiguous collection and interpretation. In CIII it was necessary to define new metrics for ML5 to have the desired confidence, because the integrity of existent data from ML3 could not be assured. With time the definition of metrics was improved to tune the process models.

R21: was part of the CMMI implementation process of CIII.

Metrics Usage

R25 and R26: SDA showed a relation between achieving HML and checking data precision and accuracy.

R29: was followed by CI.

R30: CI followed R30 in Team Software Process^{SM} pilot projects.

R31: CI did not use personal data for evaluation purposes. Even following R31, CI had difficulties to convince people to accurately report effort – that is why we suspect that showing the benefits on training may not have been effective.

Tools Setup

R32: on both CI and CIII the tools initially used were more rudimentary. As processes, metrics and performance models and baselines were defined more complex tools were adopted or implemented.

R35: followed on both CI and CIII. However, it is always difficulty to totally eliminate human intervention to report effort, especially when people have other tasks than just developing code, for example.

R34: the experience in CIII was that initially it was necessary to collect data of all variables they felt could be important to create models and establish baselines. In time the non-used metrics were abandoned, leaving only the necessary ones.

In a follow up of CII we found that R18 and R27 were used when they defined their specific metrics and implemented the estimation tool.

We compile all problems and recommendations in the checlist on table 4.19 that should be used by organisations implementing CMMI. The checklist guides them in the sequence of what shall be done to implement CMMI, gives them focus on the model as a whole, and not only a single target level to be achieved, and includes the problems that organisations should be aware of in order to avoid them.

5.2.3 Problems Analysis and Limits to Generalisation

In Table 5.14 we signal where the problems we found occurred with "Yes". 59% of the problems occurred either in other organisations or were mentioned in the literature (LR) or SEI Data Survey (DS). The number of problems we detected in each organisation increased with the depth and insight provided by a more complete design of the case study. Nevertheless, we found two groups of problems common to two different groups of two organisations.

Several problems found in CI were also detected in CII, four of them are related with metrics definition and usage and the other two are related with institutionalisation and people behaviour, respectively. Another two problems found in CI also occurred in CIII, both of them related with

	Found In				
Problem	***CI***	***CII***	***CIII***	***LR***	***DS***
P1. Underestimate time to implement HML	Yes	Yes	Yes		Yes
P2. Introduce HML forgetting ML 2 and 3			Yes	Yes	
P3. Understand the statistical/quantitative nature of level 4	Yes	Yes	Yes	Yes	Yes
P4. Copied processes		Yes			
P5. Multicultural environments		Yes			
P6. Imposed processes		Yes			
P7. Dissemination Problem	Yes				Yes
P8. Lack of Institutionalisation	Yes	Yes		Yes	
P9. Meaningless Uncorrelated Metrics	Yes			Yes	Yes
P10. Metrics definition (collect and analyse data)	Yes	Yes		Yes	Yes
P11. First data collected were uncorrelated			Yes	Yes	
P12. Metrics Categorisation	Yes	Yes			Yes
P13. Baselines not applicable to all projects	Yes	Yes			
P14. Abusive elimination of outliers	Yes				
P15. Not all projects are measurable	Yes	Yes			Yes
P16. Effort Estimates		Yes			
P17. People Behaviour	Yes	Yes			Yes
P18. Tools Setup	Yes				
P19. Overhead	Yes				
P20. Tools Requirements	Yes				
P21.Complicated Indicators without triggers for actions				Yes	
P22. Inexperienced Implementers				Yes	
P23. Complex solutions hard to maintain				Yes	
P24. Out of date measurement plans				Yes	
P25. Return of investment of metrics ignored				Yes	
P26. Senior management not involved in establishing objectives, policies and the need for processes				Yes	Yes
P27. Sponsor not playing its role and delegating authority				Yes	
P28. Software Engineering Performance Group not managed				Yes	
P29. Organisations focused on achieving ML more than improving the quality of their products or services				Yes	
P30. CMMI not understood				Yes	Yes
P31. PPM focused on final rather interin outcomes					Yes
P32. PPM considered expensive and expendable by management					Yes
P33. Too much time reporting instead of analysing					Yes
P34. Frequency of data collection insufficient for mid-course					Yes
P35.Trouble convincing management about value					Yes
Total:	14	12	4	16	15
Exclusive:	4	4	0	7	10
Shared:	10	8	4	9	5

Figure 5.14: Problems found in the case study organisations, the organisations surveyed by the SEI (DS) and the literature review (LR).

assuring entry conditions. CII was just a business unit of CIIG, who was rated ML5 for a long time, so we cannot verify if they faced similar entry conditions problems. However, we realised that the metrics problems found could be due to CII lack of understanding of the requirements for HML and statistical nature of ML4. We cannot even conclude that CIII did not face the metrics problems because we did not analyse their PPM, PPB, metrics definitions and usage in person.

37.5% of problems that we found in the literature were also detected in CI, CI and CIII, CI and CII, and CIII, respectively. 53.3% of problems found in the DS organisations were common to the ones found in our case study organisations, 16.7% of which were also found in the literature review.

Even if there are limits to the generalisation of our results the percentage of problems shared in more than one organisation/source indicates that they can occur when implementing HML, so organisations should be aware of them.

Due to access limitations, the three case studies had a different design so they cannot be considered multiple-case studies (Yin, 2009). Only part of the design of CI was repeated on CII, and in CIII we only interviewed a consultant involved in the appraisal. We can classify it has a semi-multiple case study. In CI and CII we used multiple sources of evidence, assuring construct validity. However in CIII we could not assure it. In all cases we had our results reviewed by key informants. To ensure internal validity we did pattern matching by classifying information and aggregating it under each category; built explanations and addressed rival explanations. External validity was partially tested by replicating part of the design used in CI in CII, and analysing DS. Nonetheless, for each case study we used theory.

We detected part of the problems found in the literature review (3.1.3 Problems in Process Improvements, Metrics Programs and CMMI) and additional ones:

- Processes copied from the model (P4);
- Ignored multicultural environment (P5);
- Imposed processes (P6);
- Abusive elimination of outliers (P14);
- Incomplete base for effort estimate methods in use (P16);
- Difficulties in giving the tools time to become stable (tools setup) (P18);
- Overhead introduced by the data collection (P19);
- No baseline reset or model recalibration after tools' requirements changes (P20).

Regarding limits to generalisation, we only analysed three cases but some of the problems that we identified were also found in the literature and DS. Subsequently, we consider that these problems can be common to other organisations implementing CMMI, measurement programs or doing software process improvements.

5.3 Requirements Process Improvement

This section refers to the validation of part of the steps defined in 4.5.4 Process Improvements. The purpose this research is to compile a defect classification and define the steps to consider when doing process improvements, we summarise this research work in table 5.8.

Table 5.8: Summary of information of the validation of the package Process Improvements.

Label	Description
Motivation	Requirements defects are on the tops causes of software failures and factors impacting software maintainability. In 2011 there was no defect classification specific of software defects. To ensure the quality of the defect classification we did an experiment.
Method	Do a literature review to define a classification of defects adequate to use in requirements reviews. Conduct experiments with undergraduate and graduate students. Analyse the data of the first experiment to refine the classification of defect types.
Results	Gathered a classification for type of defect, specific for requirements, that is now in used in at least an organisation. Defined the steps to consider when doing process improvements.

Chen and Huang (2009) analysed the impact of software development defects on software maintainability, and concluded that several documentation and requirements problems are amongst the top 10 higher-severity problems (see 3.4). The authors demonstrated the impact of the software requirements defects in the maintenance phase of a software project, when the defects affect the client, in case of failure. In the same year, Hamill and Katerina showed that requirements defects are among the most common types of defects in software development and that the major sources of failures are defects in requirements (32.65%) and code (32.58%). Therefore it is crucial to prevent the propagation of requirements defects to posterior phases.

Card (1998) stated that "Classifying or grouping problems helps to identify clusters in which systematic errors are likely to be found." Hence, it is important to have an adequate taxonomy to classify requirements defects, that support the following goals:

1. Identify types of defects that are more frequent or have a higher cost impact;
2. Analyse the root cause of requirements defects;
3. Prepare requirements reviews checklists;
4. Reduce risks associated with common problems in the requirements management process, such as bad communication, incomplete requirements, and final acceptance difficulties.

The Orthogonal Defect Classification (ODC) is frequently used by practitioners, but it is more adequate to classify code defects than defects in the requirements specifications (Henningsson and

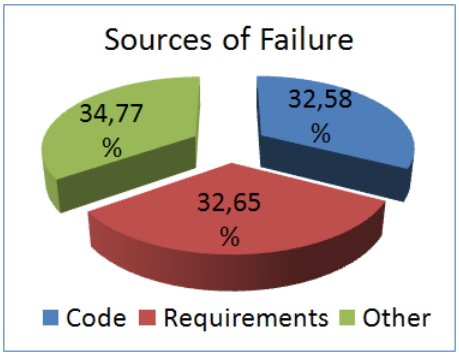


Figure 5.15: Major software sources of software failures.

Wohlin, 2004; Freimut et al., 2005). There are several classifications identified in the literature, but none of them is indicated as being the most adequate for the classification of requirements defects, and, to the best of our knowledge, their quality properties were not validated. In our research we did a literature review and proposed values for the attribute type of defect in the case of requirements using the recommendations of Freimut et al. (2005). We conducted an experiment to validate the quality properties of the proposal and test the following hypotheses, indicated in 4.5.4 Process Improvements, when reviewing requirements specifications.

In the context of this process improvement a **defect** is a fault, as defined by IEEE (1990), extended to include all the software development artefacts (code, documentation, requirements, etc.). A defect is a problem that occurs in an artefact and may lead to a **failure**. We consider the requirements review as an inspection method.

Freimut et al. (2005), indicate the quality properties of a good classification scheme:

1. Clearly and meaningfully define the attributes of the classification scheme;
2. Clearly define the values of the attributes;
3. Ensure it is complete (every defect is classifiable by using the scheme);
4. Guarantee that it contains a small number of attribute values - the authors recommend 5 to 9 attributes, since this is the number of items that human short-memory can retain (Chillarege et al., 1992);
5. Aggregate attribute values, to reduce ambiguity (Bell and Thayer, 1976), whenever they are less significant, that is, when they rarely occur, and detailed categories may be aggregated into a single one. For the attribute "type of defect" we consider that it is important that the values are unambiguous, that is, only one value is applicable to one defect.

We did the literature review (3.3 Defect Classification Taxonomies) and assembled a list of classifiers that we wanted to reduce to a useful one. We removed classifiers following the criteria:

- Not applicable to requirements phase;

- Inadequate to review a document;
- Vague an generic;
- Over-detailed;
- Duplicated, classifiers with the same meaning.

The following classifiers were excluded for the indicated reasons: Considered important only for change management: Not in current baseline, New and Changed Requirement and Not Traceable; Too vague (given the intention of having a complete and clearly defined list of values): General, Other and Inadequate; Subsumed by another (Inconsistent): Incompatible; Too generic (given the existence of separate, more specific, classifiers): Incorrect or Extra Functionality; Over-detailed (given the existence of the more generic classifiers Missing/Omission, Incorrect and Inconsistent, and the intention of keeping a small number of attribute values): classifiers 19 to 33 and 35 in Table III detailing what is missing, incorrect or inconsistent (the details can be given in the defect description).

The following classifiers with overlapping meanings (and small frequencies in some cases) were aggregated into a single one, to avoid ambiguity: Missing/Omission and Incomplete Missing or Incomplete; Over-specification, Out of scope, Intentional Deviation and Extraneous Information Not Relevant or Extraneous; Unclear and Ambiguity Ambiguous or Unclear; Infeasible, Unachievable, Non Verifiable and Unstestable/Non Verifiable Infeasible or Non-verifiable. Finally, some classifiers were slightly renamed.

The resulting 9 values for the type of defect attribute, with definitions and examples, are listed in Table II. We tried to give a clear and meaningful definition for each value.

5.3.1 Experiments with Students

We conducted two experiments with different groups of people and similar classifiers. The final list (table 5.9) used in the 2nd group had more detail in the values, definitions and examples. The 1st group was composed of master graduate students that had learnt how to develop a SRS document, and were familiar with inspections and defect classifications. The 2nd group was composed of third year undergraduate students that were familiar with SRS documents, inspections and defect classifications. We provided to each group the same SRS and the list of its defects. The subjects should register the type of defect in a form that included: the defects to classify, and distinct fields for the classifier, doubts between classifiers or to a new classifier and corresponding definition. The classification of the defects would indicate if the classifiers were ambiguous (one defect with different classifiers),meaningless (incorrectly classified) or incomplete (new classifier suggested).

Table 5.9: Classification of type of defect for requirements (final version) (Lopes Margarido et al., 2011a).

Classifier	Definition	Example
Missing or Incomplete	The requirement is not present in the requirements document. Information relevant to the requirement is missing, so the requirement is incomplete. If a word is missing without affecting the meaning of the requirement the defect shall be classified as a typo.	"The system will allow authentication of authorised users." The way to access the system is not detailed. Is it by using a login and corresponding password? Using a card? And what happens when a non-authorised user tries to access it? If the requirement includes the expression *To be Defined* (TBD) is incomplete.
Incorrect Information	The information contained in the requirement is incorrect or false, excluding typographical/grammatical errors or missing words. The requirement is in conflict with preceding documents.	Stating that "The Value Added Tax is 23%" when the correct value is 12%.
Inconsistent	The requirement or the information contained in the requirement is inconsistent with the overall document or in conflict with another requirement that is correctly specified.	One requirement states that "all lights shall be green" while another states "all lights shall be blue" (IEEE, 1998); one of the requirements is inconsistent with the other.
Ambiguous or Unclear	The requirement contains information or vocabulary that can have more than one interpretation. The information in the requirement is subjective. The requirement specification is difficult to read and understand. The meaning of a statement is not clear.	The requirement "An operator shall not have to wait for the transaction to complete." is ambiguous, depends on each person's interpretation. To be correctly specified it should be, e.g., "95% of the transactions shall be processed in less than 1 s." (IEEE, 1998).
Misplaced	The requirement is misplaced either in the section of the requirements specification document or in the functionalities, packages or system it is referring to.	Include a requirement about the server application in the section that refers to the web-client application.
Infeasible or Non-verifiable	The requirement is not implementable, e.g., due to technology limitations. The requirement implementation can not be verified in a code inspection, testing or using other verification method. If the requirement is non-verifiable due to ambiguity, incorrectness or missing information, use the corresponding classifier instead.	"The service users will be admitted in the room by a teleportation system." The teleportation technology has not sufficiently evolved to allow the implementation of such requirement. "The message sent to the space for potential extraterrestrial beings should be readable for at least 1000 years."
Redundant or Duplicate	The requirement is a duplicate of another or part of the information it contains is already present in the document, becoming redundant.	The same requirement appears more than once in the requirements specification document, or the same information is repeated.
Typo or Formatting	Orthographic, semantic, grammatical error or missing word. Misspelled words due to hurry. Formatting problems can be classified in this category.	"The system reacts to the user sensibility, i.e. if the user is screaming the system stops." The word sensibility is different from sensitivity. When a picture is out of the print area.
Not relevant or Extraneous	The requirement or part of its specification is out of the scope of the project, does not concern the project or refers to information of the detailed design. The requirement has unnecessary information.	If the customer is expecting a truck then the requirement stating "The vehicle is cabriolet." is out of the scope of the project. A requirement that should have been removed is still in the document.

The results of the experiments are summarised in the pictogram on figure 5.16, that show the proximity of the subjects' answers (dark/red circles) to the classifier that we expected them to use (bright/yellow circles) in each defect. The size of the circle gives the number of the students that used a certain classifier. There were 29 defects to classify (x axis). The classifiers, doubt between classifiers or new classifier are represented in the y axis. We noticed that in the 1st experiment no defect was unanimously classified and in the 2nd several ones were. In both experiments certain defects were classified differently but with similar percentages. These observations induce us to conclude that certain defects will be differently classified, for their own characteristics. The full report of our work includes all experiments' results and analysis (Lopes Margarido, 2010).

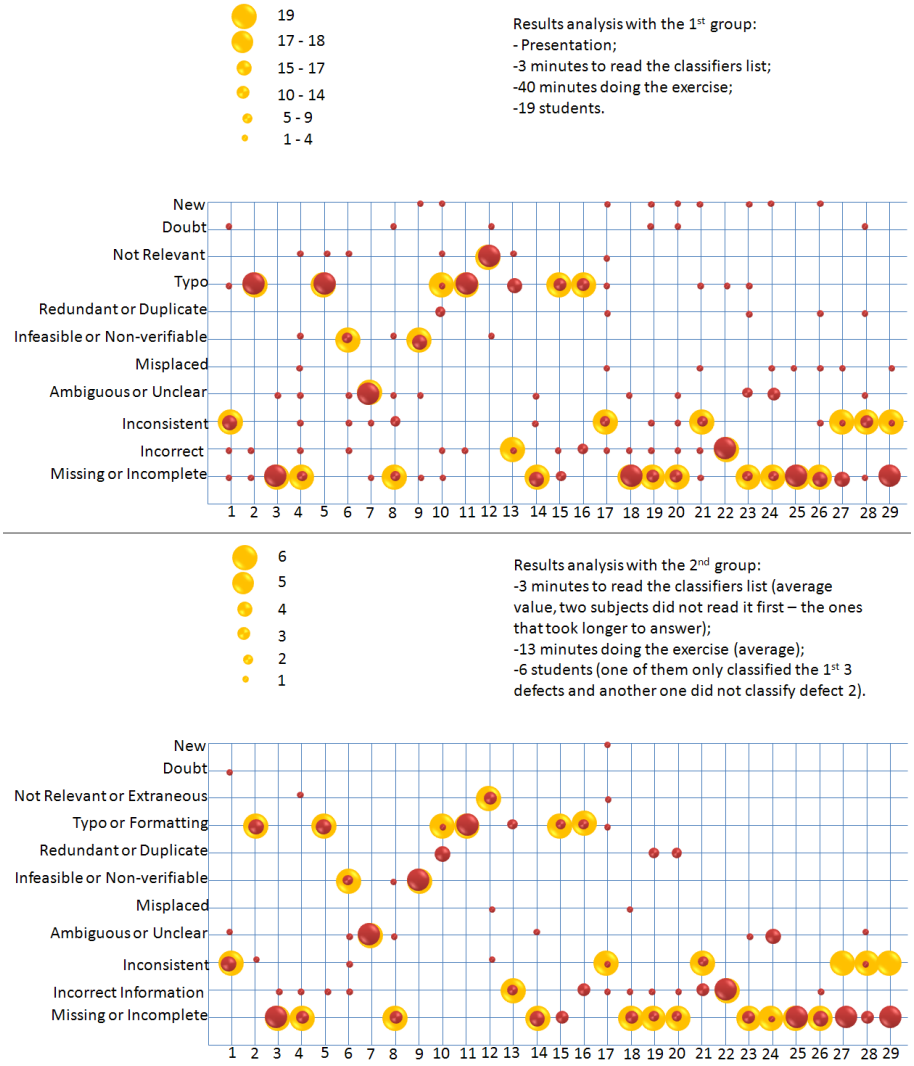


Figure 5.16: Results of the 1st experiment are represented the upper pictogram and of the 2nd are in the bottom.

The two experiments we did are not totally comparable: the experience of the individuals on defects classification and the size of the group and the treatment (values of the type of defect

attribute) were different. Despite that, the degree of agreement of the subjects, given by the Fleiss' kappa measure, was moderate in both experiments (0.46 in the 1st experiment and 0.44 in the 2nd) (Landis and Koch, 1977). We also did a Cochran test to verify our hypotheses. Since the test is binomial, we considered that when the subjects chose the most used classifier they answered as the majority (1) and when they used any other classifier, they chose other (0). The significance value indicates that the subjects answered the same way (0.60 in the 1st group and 0.63 in the 2nd, i.e. p-value > 0.05 which indicates that we cannot reject *H0*). Using the same transformation of data we did the McNemar test to verify if the results of the experiments were similar. The percentages of subjects classifying as the majority or using other classifier were similar on both experiments (see figure 5.17).

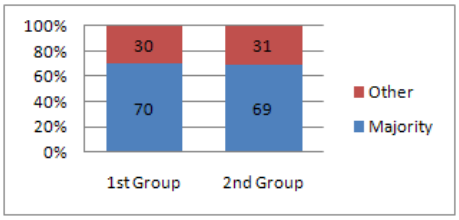


Figure 5.17: Results of the McNemar test. The experiments have approximate results.

In our opinion, the following facts may have contributed to the subjects moderate degree of agreement:

- The subjects were not the ones identifying the defects which may increase the error of misinterpretation (and consequent misclassification) of the defects;
- The subjects were not involved in the development and did not have access to the developers of the SRS document. This is similar to the problem reported in an experiment of Walia and Carver (2007);
- Certain words in the description of defects induced the selection of the classifier named with a similar word;
- The defects are expressed in natural language, which introduces ambiguity in the classification process.

Based on the experiments conducted, we suggest some recommendations for organizations that want to use requirements defects' classifications in an effective and consensual way:

- People should be trained in the usage of the defects classification focusing in the distinction of the classifiers, the clarification of their definitions, practical examples and exercises;
- To avoid that people apply a classifier based on its name only (often insufficient), without considering its definition, have the definition easily available, e.g., as a tool tip in a tool.

5.3.2 Adoption by an Organisation

The requirements defect classification we created was introduced in an organisation in 2013 part of an improvement initiative. The classification results were used to do a Pareto chart to evaluate which requirements defect types contributed to 70% of the requirements defects: Ambiguous or Unclear (25%), Typo or Formatting (17%), Incorrect Information (16%) and Missing or Incomplete(12%) (Figure 5.18). However, the classification was extended to have a *Not Applicable* and two types related with the document itself, respectively **Not Requirements - Content** and **Not Requirements - Typos or Formatting**.

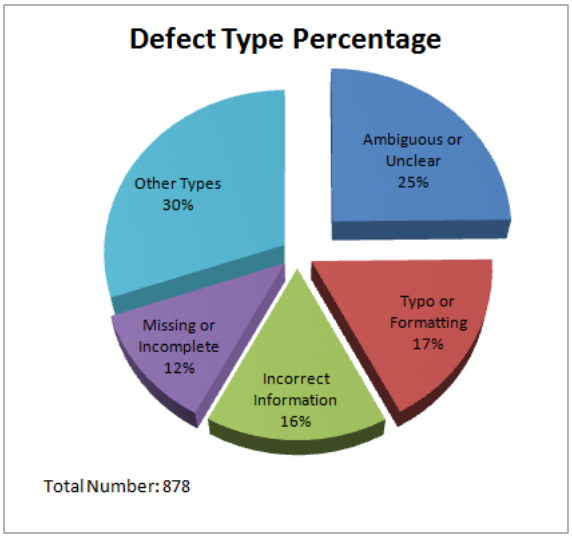


Figure 5.18: Percentage of defects found in requirements reviews by type.

The organisation considered that the use of the new requirements taxonomy successfully characterised defect types, so adopted it. The analysis results were inputs to an CAR project and used to define solutions to address the problems in the origin of these types of defects and prevent them in future reviews. The metric we recommended that should be monitored to improve the requirements review process was the number of requirements defects only found on posterior phases of the development cycle, although the organisation was interested in measuring other aspects of requirements reviews and did several other improvements that cannot isolate their effect from the one of using the taxonomy. Nonetheless, the number of defects found in requirements was considerably higher than before introducing the taxonomy and related improvements.

We agree with Card when he states that a defect taxonomy should be created in such a way that it supports the specific analysis interests of the organisation that is going to use it, namely in the implementation of defect causal analysis (Kalinowski et al., 2008). In our work based on a literature review we assembled a classification for defect types in requirements specifications, following the recommendations in (Freimut et al., 2005). Such classification is important to support the analysis of root causes of defects and their resolution, to create checklists that improve requirements reviews and to prevent risks resulting from requirements defects. When choosing a

classification for requirements' defects, organisations need to be aware of the problems of using them. People may interpret the classifiers differently and doing retrospective analysis of defects simply based on the type of defects might be misleading. Experiments similar to the one here presented may be conducted to determine the degree of consensus among their personnel.

In the case of our experiment we realised that in order to prevent the propagation of requirements defects to other software development phases it was important to improve requirements reviews. Besides using reviews checklists the use of an adequate classification to characterise defects found in requirements would:

- Make reviewers aware of the reason why the defect happened;
- Make requirements analysts more concious of what mistakes to avoid when eliciting requirements;
- Support the requirements analyst in the correction of the detected defects, through the additional understanding provided by the specific classification for requirements defect types;
- Support defects analysis;
- Allow to build a requirements checklist based on those defects.

Chapter 6

Conclusions

Concerning CMMI problems, the DoD expressed the necessity to "Develop meaningful measures of process capability based not on a maturity level, e.g. Level 3, but on process performance" (Schaeffer, 2004). CMMI V1.3 is more focused on the performance of organisations but SCAMPI is becoming more efficient (Phillips, 2010), as it reduced the number of necessary evidence – eventually increasing the probability of leaving problems undetected. In this research we develop a framework to evaluate the quality of implementation of the CMMI practices, EQualPI. The evaluation is based on the quality of outcomes, and to demonstrate the framework we build a performance indicator model to evaluate CMMI PP SP1.4 "Estimate Effort and Cost".

6.1 Research Achievements

The difficulties in implementing CMMI, in particular HML, are common to the problems found on metrics programmes and software process improvements in general. In particular, Software Engineering metrics can be ambiguous (Goulão, 2008; Breuker et al., 2009), preventing an implementation common to all organisations. With the objective of understanding CMMI problems better, we conducted a literature review, three case studies and further analysed the data of a survey conducted by the SEI. We compiled a list of problems and recommendations to help organisations implementing CMMI, and additional recommendations to support them on the choices to make regarding the implementation of MA when aiming for HML. Interestingly enough, part of the identified problems is rooted in the CMMI lower maturity levels (2 and 3) as they must be stable before moving to high maturity. The evidence we analysed show that the problems were common to the different sources of information. In the case of the ones we found in the case studies, we verified that they also occurred in other organisations. Furthermore, there are also several problems in the Measurement and Analysis process area that become more evident when implementing CMMI maturity level 4, as they affect process performance models and baselines.

With the conducted case studies we added problems to the ones in 3.1.3 Problems in Process Improvements, Metrics Programs and CMMI and 3.2 Survey on MA Performance in HML Organisations: copied processes, multicultural environments, imposed processes, baselines not

applicable to all projects, abusive elimination of outliers, effort estimates, tools setup, overhead and tools requirements. We consider that with a more complete analysis of CII and CIII we could have found more problems.

There is a wide variety of methods to implement CMMI practices. As the model is just a guide telling what to do, but not how to do it, room is left for various implementations that may not always lead to the desired performance results. Moreover, SCAMPI's objectives do not include appraising performance. Consequently, problems and difficulties can occur when implementing CMMI, some of which can persist after appraisal. EQualPI is a framework for self-assessing the quality of implementation of CMMI practices and effects of improvements, based on compliance, efficiency and effectiveness (Lopes Margarido, 2012). The framework helps preventing implementation problems and allows better control of organisations performance. We defined EQualPI's architecture architecture up to level 2, defining all its layers, their included packages and corresponding modules. The framework includes the metamodel to shape all its layers. The metamodel establishes the alignment between the EQualPI repository and the CMMI Goal or Practice. While the methods in the repository support the implementation of the Goal/Practice, the Performance Indicators quantitatively evaluate its achievement. The evaluation is done by aggregation depending on the source (project, department or organisation) or target (PI, Method or Goal/Practice).

The repository includes the data dictionary, domain model and a performance indicator model to evaluate the quality of the practice "Estimate Effort". The data model is already complete enough to include the effort estimation and the development processes. The EEA model allows organisations to anticipate the accuracy of the effort estimates achieved when estimating their projects and to act on certain factors to improve that accuracy. The model explains approximately 30% of the variability of EEA meaning that the remainder is related to non-controllable factors, related with project execution. We did a regression model of the actual hours estimated using TSP and the accuracy is already high. The 30% of EEA we explain are also used to shed light on the percentage of the actual hours that the used methods did not explain.

EQualPI includes a module to manage configurations already to consider continuous improvement of the organisation processes and allow piloting and progressive deployment of process improvement initiatives, also evaluating their impact. Once the framework is completely populated with the organisations processes, methods and performance indicators and a change is piloted, the effect of that change on other practices can be measured.

The package procedures includes how to setup the framework, select the methods and perform an evaluation of practices. A checklist of recommendations is included to avoid problems in the implementation of th CMMI practices. Those resulted of the semi-multiple case study we did on CMMI HML organisations, and analysis of a survey to organisations implementing MA aiming at achieving CMMI HML. Based on the survey technical reports and applying statistical methods to that survey data, we gathered a set of recommendations to have high maturity measurement and analysis, which can be used on any measurement program. We also provide steps to consider when doing process improvements. Part of those steps were needed to do a process improvement in the requirements review process. From that improvement we assembled a defect type classification

list specific for requirement defects, that is now used by an HML organisation. Practitioners can use the model to improve the Effort Estimation Model by acting on the variables of the model, that is reviewing the planned values estimated, before moving to the development phase of the project.

We defined a regression model useful to analyse the quality of the effort estimation process. It was firstly thought for the CMMI PP SP 1.4, based on data collected from TSP projects but ultimately it can be applied to any effort estimation process, being just a matter of changing the rules used to estimate and the recommended values used for estimation.

The use of TSP data relies on the use of size which is fit to the guidelines of PP SP1.4. In particular, several TSP rules are designed for LOC based projects, namely rates per hour and review rates per hour. This shall not be a reason for organisations that do not use such size measure to consider this research work not applicable to them; on the contrary, there are several adaptations that can be done and also several recommendations that are still applicable:

- Size can measured in any unit that fulfil their needs, as long as it is consistently measured in all projects. Since the goal of any process is to use the organisation's historical data, after a cycle of development there will be a dataset to begin understanding what are their own development rates and, with time, recommended values;
- Percentage of time spent per phase is independent of the size measure, any means used to determine the effort needed for coding will allow determine the time per phase;
- Review rates can also be adjusted in time by defining what optimal time should be spent reviewing a given work item to gather a relevant number of defects (which can give confidence that yield will be reduced).

Another reason for choosing TSP data is the fact that it is consistently collected by different organisations. Many TSP metrics are common to other methods and are defined to plan and follow the development of software. The amount of factors that we can monitor with TSP metrics makes us believe that we can build a more complete model.

6.2 Answering Research Questions

In the following paragraphs we revisit the research questions addressed in this PhD work:

- Why some organisations do not achieve the expected benefits when implementing CMMI? Depending on the methods that the organisation selected to implement the practices, their results may not be the expected. The problems found on organisations implementing CMMI show that if they are not detected and overcome the implementation can be flawed and they will not perform at the level they aimed for. Part of the problems found are rooted on a poor implementation of MA, a level 2 practice. We discussed those problems on 3.1.3 Problems in Process Improvements, Metrics Programs and CMMI, 3.2 Survey on MA Performance in HML Organisations and 5.2.3 Problems Analysis and Limits to Generalisation.

- Why SCAMPI does not detect implementation problems, or does not address performance evaluation in all maturity levels?
  SCAMPI has still limitations on its sampling and coverage rules and its purpose is not to evaluate performance, as discussed in 3.1.4 SCAMPI Limitations.

- What additional recommendations can we provide to organisations to help them avoid problems when implementing CMMI?
  We present recommendations (4.5.2 CMMI Implementation) to help organisations implement CMMI, avoiding the problems found when implementing it. They are compiled on a checklist (table in figure 4.19). As part of the implementation problems is rooted on MA and the most challenging levels to implement are the high maturity ones, we include 4.5.3 MA Recommendations for High Maturity.

- How can we evaluate the quality of implementation of the CMMI practices ensuring that organisations fully get their benefits and perform as expected?
  We propose that the quality of implementation is measured using performance indicators that measure the results of the practice and its efficiency, effectiveness, and compliance. The EQualPI evaluation procedures exemplify how it is done (4.5.1 EQualPI Setup, Tailoring and Evaluation).

- Is it possible to define metrics to evaluate the quality of implementation of CMMI practices focused on their effectiveness?
  To demonstrate that it is possible to evaluate the quality of implementation of a practice focusing on its effectiveness we built the Effort Estimation Accuracy model to evaluate PP SP1.4. The objective of the practice is to provide estimates to plan project execution. We evaluated the effectiveness of the practice through the deviation of the actual effort relative to the estimate, showing how reliable the estimate was. The execution of the project also influences this result, but knowing the percentage it represents on the indicator, as part of the non-controllable factors, we determined how effective the estimate was. The answer to this question is yes, and we built the model to evaluate PP SP1.4 on 5.1 Evaluation of the Estimation Process.

- Can we determine the effects, expressed in a percentage, of non-controllable factors in an evaluation metric?
  In the EEA model, useful to anticipate the quality of the estimates and act on the model coefficients to improve the estimates, the percentage of controllable factors is given by the model Adjusted R Square, as it represents the percentage of the dependent variable that is explained. For the used data the controllable factors represent 30% of the variation. The non-controllable factors are represented on the remainder 70%.

We showed the relationship between the quality of implementation of a CMMI practice and the quality of the outcome of the application of that practice, exemplified on PP SP1.4. Therefore, "it is possible to objectively measure the quality of implementation of the CMMI practices by

applying statistical methods, in the analysis of organisations' data, in order to evaluate process improvement initiatives and predict their impact on organisational performance".

6.3 Challenges and Limits to Generalisation

EQualPI was designed aligned with CMMI, includes a regression model for one of its practices and its procedures are targeting CMMI. Considering that its based on the problems of metrics programs and other process improvements, it can be used to evaluate other practices of other standards and models. Any set of methods can be mapped with the processes defined to comply with a different model and the measures are defined based on the goal to achieve, rather than the ones in CMMI.

The data model is applicable to iterative and non-iterative development cycles, considering that when there are no cycles the project only executes one cycle. Even though, the design of the model was based on the TSP data, the variables are common in software development. The adaptation to projects following Scrum would be more challenging even though some in some cases would just require relabelling the variable.

EQualPI is defined and validated but it still needs to be extended to other practices and be fully populated with the corresponding methods, performance indicators and goals. Additionally, it was not instantiated in an organisation, which may be challenging for its complexity. Besides, to provide the data is still necessary that the organisation collects it.

We identified several factors that impact the generalisation of the research results, such awareness allowed us to keep them controlled to isolate their interference in the results the best we can. We kept a record of context and those factors that are not object of our research, to allow repeatability.

Human Factors

The experience of a project team influences the results of projects. Also the quality of individuals' work can positively or negatively influence the project's results. The quality of the data used to build the models depends on the people's rigour recording it. The Benford statistic helped us ensure that the effort data in the TSP repository was reliable.

Aggregation Factors

When aggregating results of individuals and different teams, inaccurate data points' noise can be cancelled by other. In other situations, the outputs of the team influences the following phase. Such noise can change the aggregated information. Once we detected inaccurate data, e.g. the several projects with same defects descriptions we isolated them, not considering them in the model.

Complexity Factors

The complexity of projects depends on different factors such as size, duration, complexity of the product, newness of the technology, etc. We also recorded other context factors such as team

size and distribution. Those factors were documented but not included in the model (are part of the error).

Biasing Factors

Other factors that limit the generalisation of the research results are the bias of the researcher and the analysed organisations. In general, to avoid researcher's bias the work was submitted to other researchers' reviews, conferences and journals. Organisations themselves can bias the results by keeping certain information or altering the data shared. For this reason the data analysis when possible we analysed data of several organisations. We also did sanity checks on the received organisations' data.

Measurement Selection Bias

Grimstad and Jorgensen (2006) highlight three measurement selection factors that are particularly important:

1. Exclusion of cancelled projects, leading to a too positive view of estimation;
2. Exclusion of estimated projects that never started, leading to a negative view of estimation (according to the authors optimistic plans would be more likely to win bidding, for example);
3. Inclusion of projects to "confirm the desired output of the analysis" leaving others, named "confirmation" bias.

We did not consider cancelled projects before start, as they would not be present in the database; we also did not find cases of projects that were estimated but did not have any records of data resultant from their execution. These two biasing factors were out of the scope of the research. Even though we did not find cases of cancellation before even starting we only considered completed parts in the design of the model, ignoring the parts that were planned but not executed. The decision was based on the fact that we had no absolute confirmation of whether the workbook was of a completed project or referring to an intermediate development cycle.

Regarding the third factor we did not select projects by convenience but we also did not get the degree of variability we desired as if we were able to get the same data from other repositories, as we just evaluated the effects of using methods that are common to TSP.

6.4 Future Research Work

As future research work, it is necessary to extend the framework to other practices and methods, also including more performance models, and to instantiate it in an organisation. With data of different maturity levels, it will be possible to evolve the map of maturity profiles to other methods and performance indicators. In the future, EQualPI can be used for benchmarking, supporting multiple clients that eventually will be able to compare their performance with the anonymous data

of others that allow selected data to be used by others. The data dictionary is already prepared to support multiple organisations, but the architecture of the framework needs an additional package in the business layer to perform data consistency checks.

With a more complete dataset other Process Variables can be included in the EEA model. Also, having accurate data of different cycles can help determine the influences of partial estimates or effects of re-estimation. Having other sources of data, a similar approach to the one we followed when building the EEA model, can be followed to define EEA models for other estimation methods such as the one used in scrum, for example.

References

Fernando Manuel Pereira da Costa Brito e Abreu. *Engenharia de software orientado a objectos : uma aproximação quantitativa*. Doctoral thesis, 2001.

A. Frank Ackerman, Lynne S. Buchwald, and Frank H. Lewski. Software inspections: An effective verification process. *IEEE Software*, 6(3):31–36, 1989.

Larry Apfelbaum and John Doyle. Model based testing, 1997.

James Armstrong, Richard Barbour, Richard Hefner, and David H. Kitson. Standard cmmism appraisal method for process improvement (scampism): Improvements and integration. *Systems Engineering*, 5(1):19–26, 2002.

Monalessa Perini Barcellos. *Uma Estratégia para Medição de Software e Avaliação de Bases de Medidas para Controlo Estatístico de Processos de Software em Organizações de Alta Maturidade*. Doctoral, 2009.

Victor R. Basili and David M. Weiss. Evaluation of a software requirements document by analysis of change data, 1981.

T. E. Bell and T. A. Thayer. Software requirements: Are they really a problem?, 1976.

Terry Bollinger and Clement McGowan. A critical look at software capability evaluations: An update. *IEEE Software*, 26(5):80–83, 2009.

Petrônio L. Braga, Adriano L. I. Oliveira, and Silvio R. L. Meira. A ga-based feature selection and parameters optimization for support vector regression applied to software effort estimation, 2008.

Dennis Breuker, Jacob Brunekreef, Jan Derriks, and Ahmed Nait Aicha. Reliability of software metrics tools, 2009.

Paul D. Byrnes. What's all the fuss... what's really different about scampi v1.3, June 2011.

Rachel Callison and Marlene MacDonald. A bibliography of the personal software process (psp) and the team software process (tsp). Technical Report CMU/SEI-2009-SR-025, CMU/SEI, October 2009 2009.

Michael Campo. Why cmmi maturity level 5? *CROSSTALK The Journal of Defense Software Engineering*, (January/February):15–18, 2012.

G. Canfora, Félix García, M. Piattini, F. Ruiz, and C. Visaggio. Applying a framework for the improvement of software process maturity. *Software Practice and Experience*, 36(3):283–304, 2006.

David N. Card. Learning from our mistakes with defect causal analysis. *IEEE Softw.*, 15(1):56–63, 1998.

Robert Charette, Laura M. Dwinnell, and John McGarry. Understanding the roots of process performance failure. *CROSSTALK The Journal of Defense Software Engineering*, 17(8):18–22, 2004.

Jie-Cherng Chen and Sun-Jen Huang. An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software*, 82(6): 981–992, 2009.

Ram Chillarege, Inderpal S. Bhandari, Jarir K. Chaar, Michael J. Halliday, Diane S. Moebus, Bonnie K. Ray, and Man-Yuen Wong. Orthogonal defect classification - a concept for in-process measurements. *IEEE Transactions on Software Engineering*, 18(11):943–956, 1992.

Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. *CMMI for Development®: Guidelines for Process Integration and Product Improvement*. SEI Series in Software Engineering. Addison-Wesley, Massachusetts, 3 edition, 2011.

CISQ. Automated function points (afp), 2014.

CISQ. Cisq code quality standards, 2016.

CMU/SEI. Criteria for audits of high maturity appraisals, 2008.

CMU/SEI. Standard cmmi® appraisal method for process improvement (scampism) a, version 1.3: Method definition document. Technical Report CMU/SEI-2011-HB-001, CMU/SEI, 2011. SCAMPI Upgrade Team.

A. Colombo, E. Damiani, F. Frati, S. Oltolina, K. Reed, and G. Ruffatti. The use of a meta-model to support multi-project process measurement. In *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*, pages 503–510, 2008.

Bill Curtis. Software quality measurement. In *Software Assurance Forum*. CMU/SEI, 2010.

Noopur Davis and Jim McHale. Relating the team software processsm (tspsm) to the capability maturity model® for software (sw-cmm®). Technical Report CMU/SEI-2002-TR-008, ESC-TR-2002-008, CMU/SEI, March 2003.

Noopur Davis and Julia Mullaney. The team software processsm (tspsm) in practice: A summary of recent results. Technical Report CMU/SEI-2003-TR-014, ESC-TR-2003-014, CMU/SEI, 2003.

Tore Dybå. *Experiences in Process Modelling and Enactment: an Investigation of the Importance of Organisational Issues*. Doctoral dissertation, 2001.

Pascoal Faria. A path for performance improvement: the personal software process (psp) and the team software process (tsp), 6-11-2009 2009.

Bob Ferguson. Leading indicators of program management, 2010.

Robert Ferguson, Dennis Goldenson, James McCurley, Robert Stoddard, David Zubrow, and Debra Anderson. Quantifying uncertainty in early lifecycle cost estimation (quelce). Technical Report CMU/SEI-2011-TR-02, 2011.

William A. Florac, Anita D. Carleton, and Julie R. Barnard. Statistical process control: Analyzing a space shuttle onboard software process. *IEEE Softw.*, 17(4):97–106, 2000.

T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *Software Engineering, IEEE Transactions on*, 29(11):985–995, 2003.

Bernd Freimut, Christian Denger, and Markus Ketterer. An industrial case study of implementing and validating defect classification for process improvement and quality management, 2005.

Alfonso Fuggetta. Software process: a roadmap, 2000.

F. García, M. Bertoa, C. Calero, A. Vallecillo, and F. Ruiz. Towards a consistent terminology for software measurement. *Information and Software Technology*, 48(8):631–644, 2006.

F. García, M. Serrano, J. Cruz-Lemus, F. Ruiz, and M. Piattini. Managing software process measurement: A metamodel-based approach. *Information Sciences*, 177(12):2570–2586, 2007.

Félix García, Francisco Ruiz, José Cruz, and Mario Piattini. Integrated measurement for the evaluation and improvement of software processes. volume 2786 of *Lecture Notes in Computer Science*, pages 94–111. Springer Berlin / Heidelberg, 2003.

Dennis R. Goldenson, Diane L. Gibson, and Robert W. Ferguson. Why make the switch? evidence about the benefits of cmmi, 2004.

Dennis R. Goldenson, James McCurley, and Robert W. Stoddard II. Use and organizational effects of measurement and analysis in high maturity organizations: Results from the 2008 sei state of measurement and analysis practice surveys. Technical report, CMU/SEI, 2008.

Miguel Carlos Pacheco Afonso Goulão. *Component-Based Software Engineering: a Quantitative Approach*. Doctoral, 2008.

Robert B. Grady. *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc., 1992.

Stein Grimstad and Magne Jorgensen. A framework for the analysis of software cost estimation accuracy, 2006.

Anonimous Research Group. Search nach measures in cmmi v1.2, April 2007.

Maggie Hamill and Goseva-Popstojanova Katerina. Common trends in software fault and failure data. *IEEE Trans. Softw. Eng.*, 35(4):484–496, 2009.

Patrick Hamon and Olivier Pinette. Les indicateurs mesure & analyse. Technical report, Spirula, 22-06-2010 2010. Mauvaises pratiques.

CH. V. M. K. Hari and P. V. G. D. Prasad Reddy. A fine parameter tuning for cocomo 81 software effort estimation using particle swarm optimization. *Journal of Software Engineering*, 5(1): 38–48, 2011.

J. Huffman Hayes. Building a requirement fault taxonomy: Experiences from a nasa verification and validation research project, November 2003 2003.

Jane Huffman Hayes, Inies Raphael, David M. Pruett, and Elizabeth Ashlee Holbrook. Case history of international space station requirement faults, 2006.

Kennet Henningsson and Claes Wohlin. Assuring fault classification agreement - an empirical evaluation, 2004.

Craig Hollenbach and Doug Smith. A portrait of a cmmism level 4 effort. *Systems Engineering*, 5 (1):52–61, 2002.

Nien-Lin Hsueh, Wen-Hsiang Shen, Zhi-Wei Yang, and Don-Lin Yang. Applying uml and software simulation for process definition, verification, and validation. *Inf. Softw. Technol.*, 50 (9-10):897–911, 2008.

Watts S. Humphrey. The software engineering process: Definition and scope, 1988.

Watts S. Humphrey. Introduction to software process improvement. Technical Report CMU/SEI-92-TR-7, CMU/SEI, June 1992 (Revised June 1993) 1992.

Watts S. Humphrey. *PSPSM A Self-Improvement Process for Software Engineers*. The SEI Series in Software Engineering. Addison-Wesley, 2005.

Watts S. Humphrey. *TSPSM: Coaching Development Teams*. The SEI Series in Software Engineering. Addison-Wesley, 2006.

IEEE. Ieee standard glossary of software engineering terminology, 1990.

IEEE. Ieee recommended practice for software requirements specifications, 1998.

Investopedia, 2007.

ISO. Fdis 9126-1 software engineering - product quality - part 1: Quality model., 2001.

R. Jeffery and M. Berry. A framework for evaluation and prediction of metrics program success. In *Software Metrics Symposium, 1993. Proceedings., First International*, pages 28–39, 1993.

Philip M. Johnson, Hongbing Kou, Michael Paulding, Qin Zhang, Aaron Kagawa, and Takuya Yamashita. Improving software development management through software project telemetry. *IEEE Softw.*, 22(4):76–85, 2005.

Capers Jones. *Applied Software Management: Assuring Productivity and Quality*. Software Engineering Series. McGraw-Hill, Inc., New York, 1991.

Capers Jones. *Software Engineering Best Practices - Lessons from Successful Projects in the Top Companies*. McGraw Hill, 2010.

M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *Software Engineering, IEEE Transactions on*, 33(1):33–53, 2007.

Magne Jørgensen. Regression models of software development effort estimation accuracy and bias. *Empirical Software Engineering*, 9(4):297–314, 2004.

Magne Jørgensen. A critique of how we measure and interpret the accuracy of software development effort estimation, 2007.

Eung Sup Jun and Jae Kyu Lee. Quasi-optimal case-selective neural network model for software effort estimation. *Expert Systems with Applications*, 21(1):1–14, 2001.

Marcos Kalinowski, Guilherme H. Travassos, and David N. Card. Guidance for efficiently implementing defect causal analysis, June 2008.

Marcos Kalinowski, Emilia Mendes, David Card, and Guilherme Travassos. Applying dppi: A defect causal analysis approach using bayesian networks. In M. Ali Babar, Matias Vierimaa, and Markku Oivo, editors, *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*, pages 92–106. Springer Berlin / Heidelberg, 2010.

Mark Kasunic. Performance benchmarking consortium, 11-17 November 2006.

Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. Towards a framework for software measurement validation. *IEEE Trans. Softw. Eng.*, 21(12):929–944, 1995.

Barbara Kitchenham, Cat Kutay, Ross Jeffrey, and Colin Connaughton. Lessons learnt from the analysis of large-scale corporate databases, 2006.

Peter Kueng. Process performance measurement system: a tool to support process-based organizations. *Total Quality Management*, 11(1):67 – 85, 2000.

J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

Peter Leeson. Why the cmmi® does not work, 9-12 June 2009.

Isabel Lopes Margarido. Requirements defects classification list. Technical Report PRODEI-0903-TR-001, Faculty of Engineering, University of Porto, 2010-08-06 2010.

Isabel Lopes Margarido. Summary of literature review on effort estimation. Technical Report PRODEI-0903-TR-003, Faculty of Engineering, University of Porto, 2012.

Isabel Lopes Margarido, João Pascoal Faria, Marco Vieira, and Raul Moreira Vidal. Classification of defect types in requirements specifications: Literature review, proposal and assessment, 15-18 June 2011 2011a.

Isabel Lopes Margarido, João Pascoal Faria, Marco Vieira, and Raul Moreira Vidal. Cmmi practices: Evaluating the quality of the implementation, 2011b.

Isabel Lopes Margarido, João Pascoal Faria, Raul Moreira Vidal, and Marco Vieira. Challenges in implementing cmmi® high maturity: Lessons learned and recommendations. *Software Quality Professional*, 16(1), 2013.

Cuauhtemoc Lopez-Martin. A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables. *Appl. Soft Comput.*, 11(1):724–732, 2011.

Robyn R. Lutz and Carmen Mikulski. Requirements discovery during the testing of safety-critical software, 2003.

JoséM Álvarez, Andy Evans, and Paul Sammut. Mapping between levels in the metamodel architecture. In Martin Gogolla and Cris Kobryn, editors, *UML 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, volume 2185 of *Lecture Notes in Computer Science*, pages 34–46. Springer Berlin Heidelberg, 2001.

Steve Masters, PhD Sandi Behrens, Judah Mogilensky, and Charlie Ryan. Scampi lead appraisersm body of knowledge (sla bok). Technical Report CMU/SEI-2007-TR-019, ESC-TR-2007-019, CMU/SEI, 2007.

Lawrence McCarthy. Piloting results-based appraisals, 2009.

James McCurley and Dennis R. Goldenson. Performance effects of measurement and analysis: Perspectives from cmmi high maturity organizations and appraisers. Technical report, CMU/SEI, 2010.

John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley, 2002.

James McHale, Timothy A. Chick, and Eugene Miluk. Implementation guidance for the accelerated improvement method (aim). Technical Report CMU/SEI-2010-SR-032, CMU/SEI, 2010.

S. Mishra and B. H. Schlingloff. Compliance of cmmi process area with specification based development. In *Software Engineering Research, Management and Applications, 2008. SERA '08. Sixth International Conference on*, pages 77–84, 2008.

Gary Mogyorodi. Requirements-based testing: an overview, 2001.

Kjetil Moløkken and Magne Jørgensen. A review of surveys on software effort estimation. *Journal of Systems and Software*, 70(1-2):37–60, 2004.

O. Monkevich. Sdl-based specification and testing strategy for communication network protocols, 1999.

Paula Monteiro, Ricardo Machado, Rick Kazman, and Cristina Henriques. Dependency analysis between cmmi process areas product-focused software process improvement. volume 6156 of *Lecture Notes in Computer Science*, pages 263–275. Springer Berlin / Heidelberg, 2010.

Bob Moore and Will Hayes. Building a credible scampi appraisal representative sample, 2005.

Bob Moore and Will Hayes. Practical advice on picking the right projects for an appraisal, 11-17 November 2006.

Ofer Morgenshtern, Tzvi Raz, and Dov Dvir. Factors affecting duration and effort estimation errors in software development projects. *Inf. Softw. Technol.*, 49(8):827–837, 2007.

Mahmood Niazi, David Wilson, and Didar Zowghi. A maturity model for the implementation of software process improvement: an empirical study. *J. Syst. Softw.*, 74(2):155–172, 2005.

William R. Nichols, Mark Kasunic, and Timothy A. Chick. Tsp performance and capability evaluation (pace): Customer guide. Technical report, 2013.

OMG. Software & systems process engineering meta-model specification, 2008.

E. Palza, C. Fuhrman, and A. Abran. Establishing a generic and multidimensional measurement repository in cmmi context. In *Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard*, pages 12–20, 2003.

Robert E. Park, Wolfhart B. Goethert, and William A. Florac. Goal-driven software measurement - a guidebook. Technical report, CMU/SEI, August 1996 1996.

Shari Lawrence Pfleeger, Ross Jeffery, Bill Curtis, and Barbara Kitchenham. Status report on software measurement. *IEEE Softw.*, 14(2):33–43, 1997.

Mike Phillips. Cmmi v1.3 planned improvements, 28 June - 1 July 2010.

Mike Phillips and Sandy Shrum. Process improvement for all: What to expect from cmmi version 1.3. *CROSSTALK The Journal of Defense Software Engineering*, 23(1):10–14, 2010.

Adam A. Porter, Jr. Votta, Lawrence G., and Victor R. Basili. Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, 21(6):563–575, 1995.

Simona Pricope and Lichter Horst. Towards a systematic metric based approach to evaluate scampi appraisals, June 15-17 2009.

Arthur Pyster. What beyond cmmi is needed to help assure program and project success? unifying the software process spectrum. volume 3840 of *Lecture Notes in Computer Science*, pages 75–82. Springer Berlin / Heidelberg, 2006.

Ron Radice. Statistical process control in level 4 and level 5 software organizations worldwide, May 4 2000.

Ron Radice. Scampism with sw-cmm®, 2003.

Bryce Ragland. Measure, metric, or indicator: What's the difference? *CROSSTALK The Journal of Defense Software Engineering*, 1995.

Hans Sassenburg. Standard investigation method for benchmarking it organisations (simbio), 2009.

Hans Sassenburg and L. Voinea. Does process improvement really pay off?, 28 June - 1 July 2010.

Mark Schaeffer. Dod systems engineering and cmmi, November 17, 2004 2004.

G. Michael Schneider, Johnny Martin, and W. T. Tsai. An experimental study of fault detection in user requirements documents. *ACM Trans. Softw. Eng. Methodol.*, 1(2):188–204, 1992.

David Schreb. Accelerated improvement method (aim). Technical report, CMU/SEI, May 2010 2010.

Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. Scrum.org, 2013.

SEI. Sema, 2016.

CMU SEI. Accelerated improvement method (aim). Technical report, May 2010 2010.

R. K. Smith, J. E. Hale, and A. S. Parrish. An empirical study using task assignment patterns to improve the accuracy of software effort estimation. *Software Engineering, IEEE Transactions on*, 27(3):264–271, 2001.

Thanwadee Sunetnanta, Ni-On Nobprapai, and Olly Gotel. Quantitative cmmi assessment for offshoring through the analysis of project management repositories, July 2-3 2009.

Adriano Takara, Aletéia Xavier Bettin, and Carlos Miguel Tobar Toledo. Problems and pitfalls in a cmmi level 3 to level 4 migration process, 12-14th of September 2007 2007.

Shurei Tamura. Integrating cmmi and tsp/psp: Using tsp data to create process performance models. Technical Report CMU/SEI-2009-TN-033, CMU/SEI, November 2009 2009.

Hans van Vliet. *Software Engineering: Principles and Practice*. Willey, New York, 2007.

Gursimran S. Walia and Jeffrey C. Carver. Development of requirement error taxonomy as a quality improvement approach: A systematic literature review. Technical Report MSU-070404, Department of Computer Science and Engineering, 2007.

David R. Webb, Dr. Gene Miluk, and Jim Van Buren. Cmmi level 5 and the team software process. *CROSSTALK The Journal of Defense Software Engineering*, pages 16–21, 2007.

David N. Wilson, Tracy Hall, and Nathan Baddoo. A framework for evaluation and prediction of software process improvement success. *J. Syst. Softw.*, 59(2):135–142, 2001.

Robert K. Yin. *Case Study Research Design and Methods*. Applied Social Research Methods Series. SAGE, fourth edition, 2009.

Horst Zuse. *A Framework of Software Measurement*. Walter de Gruyter & Co., 1997.

Appendix A

Effort Estimation Methods

A.1 Effort Estimation Methods

Effort estimation methods classification from our literature review on effort estimation (3.4):

Table A.1: Effort Estimation methods.

Name	Classification	Ref.	Comments
COCOMO I and II	model based [4] statistical model [7]	[2, 4, 7, 8] [9-11]	needs recalibration [2] [12], validity of some factors in our days is questionable [12] Boehm, 1984, 1988 [7]
FPA Metrics	model based [4]	[4, 10, 13]	
FPA		[8, 14]	"based on metric using user specifications, such as number of inputs, master files, number of logical files, number of interfaces and number of outputs to estimate software size.[14]"
MK II FPA		[8]	
Capacity Related and Price-to-Win	not “pure”	[4]	
Delphi		[10]	
Price-S		[15]	
SEER-SEM		[15]	
Putnam’s SLIM model	statistical model	[7, 10, 15]	Boehm, 1984; Putnam, 1978
Putnum		[13]	

Continued on next page

Table A.1 – *Continued from previous page*

Name	Classification	Ref.	Comments
Doty model	statistical model	[7, 9-11]	Boehm, 1984; Herd, 1977
Bailey + Basili Meta model	statistical model	[7, 9, 10]	Bailey&Basili, 1981; Boehm, 1984
TRW model	statistical model	[7]	Boehm, 1984; Wolverton, 1974 [7]
Halsted Equation		[9-11]	
Walston-Felix		[9][11]	
Anish Mittal		[10]	
Swarup		[10]	
Least Squares Regression (LSR)		[2, 16, 17]	"generates regression model based on statistic minimizes the sum of squared errors to determine the best estimates for coefficients[9].[16]"
Expert Judgement		[4, 5]	"there is no evidence that formal estimation models are more accurate [4]"
Top-down and Bottom-up		[4]	"can be used in combination with other methods [4]"
Use Case Based	model based	[4]	
Use Case Points (UCP)		[8]	"The Use Case Points (UCP) estimation method introduced in 1993 by Karner estimates effort in person-hours based on use cases that mainly specify functional requirements of a system [11] [12]. Use cases are assumed to be developed from scratch, be sufficiently detailed and typically have less than 10-12 transactions." The method "is an extension of the Function Points Analysis and MK II Function Points Analysis [21]. [8]"
Artificial neural networks	machine learning	[2]	
Fuzzy Logic (FL)	machine learning	[5, 9]	GP, COCOMO and PSO have almost similar properties. FL has the lowest MMRE.

Continued on next page

Table A.1 – *Continued from previous page*

Name	Classification	Ref.	Comments
Neural Networks	machine learning	[5, 16]	"The neural network with hidden layers allows the non-linear mapping function between the causing input factors and output results. (Jun and Lee, 2001)"
Radial Basis Function (RBF) neural networks	machine learning	[18]	
MLP neural networks machine learning		[18]	"Multi-layer perceptron – applied in classification, regression and time series forecasting. [18]"
Wavelet neural networks	machine learning	[18]	
Genetic Programming (GP)	machine learning	[5, 9, 18]	
Genetic Algorithm	machine learning	[18]	
Regression Trees *Multiple additive regression trees*	machine learning machine learning	[5] [18]	"Model Trees – machine learning method for classification and regression. The leaves perform linear regression functions. Produce more understandable results than MLP."
Case-based Reasoning	machine learning	[5]	"attempts to seek a solution of the most similar past case(s), and modifies the solution considering the differences from the new target case (Jun and Lee, 2001)." Analogy with past projects (Morgenshtern et al., 2007)
Bagging predictors	machine learning	[18]	
Support vector regression (SVM)	machine learning	[18]	"based on statistical learning theory. Outperformns radial basis functions neural networks (RBFN) for software effort estimation in NASA projects' data. [18]"
Hierarchical Bayesian inference		[19]	
Bayesian Network		[16]	

Continued on next page

Table A.1 – *Continued from previous page*

Name	Classification	Ref.	Comments
Particle Swarm Optimisation (PSO)		[9, 11]	
	Features Selection		
Feature Subset Selection (FSS)		[15]	feature subset selection (FSS) and extrapolation, the selection and effort estimation is based on software parts
Effort Unit Matrix		[13]	Effort estimation for php forms, databases and documents
GA for feature selection		[18]	Reduces number of input features.
	Analogy-based Tools		
ESTOR		[11]	Size
ANGEL		[11]	Size
ACE		[11]	Size
COCONUT		[11]	Search a and b parameters in COCOMO I

A.2 Factors Related with the Process

The following table presents the factors that can be considered on the effort estimation process.

Table A.2: Factors considered on effort estimation.

Name	Definition	Ref.	Comments
Expert judgment skills	"ability to estimate the development effort of a software project applying judgement-based estimation methods"	[22]	Estimation ability factor

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Flexibility in product and process execution	"If the project has a flexible scope, a simplification of the product can compensate for initially poor estimates and thus reduce estimation complexity and risk."	[22]	Estimation Complexity factor Incurring in the risk of being criticised for our decision, we will consider flexibility in product and process execution as a controllable factor, in particular flexibility in product. This is the context of agile development, products that can change as they evolve. So we consider that this is a controllable factor.
Inconsistent use of terminology	"When there is a lack of clear definitions of terms and there exist differences in interpretations of important estimation terminology, variance in estimation error cannot automatically be attributed to variance in estimation ability or estimation complexity."	[22]	We will try as much as possible to identify if the estimate includes a buffer and quantify it; is a weighted average of optimist, pessimist and most probable or simply represents 'most likely effort'.
Analysts capability: acap(COCOMO(CI, II) Estimator experience from similar projects	Staff skill level [7] Estimator experience [6]	[6, 7, 12, 22] [26-29]	Increase this to decrease effort [12] Lower duration estimation error when considered the experience in the specific application area in number of projects [6]
Programmer capability: pcap (CI,II) Programmer qualifications	Staff skill level [7] Cumulative experience [30]	[7, 12, 26, 27] [28-31]	Increase this to decrease effort [12]

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Application experience: aexp (CI, II) Programmer experience with application Team experience Staff	skill level [7] Cumulative experience [30] Project uncertainty [6]	[6, 7, 12, 26] [27-29] [30-32]	Increase this to decrease effort [12]
Modern programming practices: modp (CI)	Project requirement [7]	[7, 12, 26, 28] [29, 32-34]	Increase this to decrease effort [12] Productivity increases with "with the high use of top-down design, modular design, design reviews, code inspections, and quality-assurance programs [33]" Increases productivity [34]
Use of software tools: tool (CI, II)	Project requirement [7]	[7, 12, 26, 27] [28, 29, 31, 34]	Increase this to decrease effort [12] Increases productivity [34]
Virtual machine experience: vexp (CI)	Staff skill level [7]	[7, 12, 26, 32] [28, 29]	Increase this to decrease effort [12]
Language experience: lexp (CI) Language and tool experience (CII) Programming language experience Programmer experience with language	Staff skill level [7] Cumulative experience [30]	[7, 12, 26, 27] [28-30] [31, 32]	Increase this to decrease effort [12]

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Schedule constraint: sced (CI) Required development schedule (CII)	Timing Project requirement [7] Resource Constraints [33]	[7, 12, 27, 28] [29, 31, 33, 35]	
Main memory constraint: stor (CI, II) Memory utilisation Main storage constraint	Computing platform [7] Resource Constraints [33]	[7, 12, 26, 28] [29, 31, 32] [33, 34]	Decrease this to decrease effort [12]
Database size: data (CI, II) Database complexity	Characteristics of products [7] Cumulative complexity [30]	[12, 26, 27] [28-30] [31, 36]	Decrease this to decrease effort [12]
Time constraint for CPU: time (CI) Execution time constraints (CII)	CPU occupancy Computing platform [7] Resource Constraints [33]	[7, 12, 26, 27] [28, 29, 31] [32, 33]	Decrease this to decrease effort [12]
Turnaround time: turn (CI) Computer turnaround time	Computing platform [7]	[7, 12, 26, 27] [28]	Decrease this to decrease effort [12]

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Machine volatility: virt (CI) Virtual machine volatility	Computing platform [7]	[12, 26, 28, 29]	Decrease this to decrease effort [12]
Process complexity: cplx (CI) Product complexity (CII) Application process complexity Implementation complexity	Characteristics of products [7] Cumulative complexity [30] Program complexity [33] Project uncertainty [6]	[6, 7, 12, 26] [27-30] [31-33, 35]	Decrease this to decrease effort [12] Productivity decreases with higher percentage of complex code. Product related (non-controllable by project management) [33].
Required software reliability: rely (CI, II)	Characteristics of products [7]	[12, 26-28] [29, 31]	Decrease this to decrease effort [12]
Development for reusability: ruse (CII)		[31]	
Platform volatility: pvol (CII)		[31]	
Platform experience: plex (CII)		[31]	
Personnel continuity: pcon (CII)	Staff skill level [7]	[7, 29, 31, 32]	
Multisite development: site (CII)		[31]	
Documentation needs: docu (CII)		[31]	
Reuse	Project requirement [7]	[7, 26, 28, 29] [32]	
Type of project	Project requirement [7]	[7, 28]	We included this factor in the data dictionary

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Programming language used	Project requirement [7]	[7, 27, 28, 35]	We included this factor in the data classification table
Software development mode	Project requirement [7]	[7, 26, 28]	
Number of source codes	Project requirement [7]	[26, 28, 29, 32] [35]	
Project size (functions or modules)	Project requirement [7]	[7, 35, 36]	
Use of chief programmer team	Project requirement [7] Total methodology [33]	[32][33]	
Team size	Project requirement [7] Work assignment factor [24]	[7, 24, 27, 34] [35]	Lowers productivity [34] From previous work it should increase development effort but it did not happen in the analysed data [24].
Design volatility	Characteristics of products [7]	[26, 27, 29]	
Complexity of delivered codes	Characteristics of products [7]	[7, 26, 27, 29] [35]	
Complexity of application processing	Characteristics of products [7]	[7, 27, 29, 36]	
Complexity of program flow	Characteristics of products [7] Cumulative complexity [30]	[7, 27, 29, 30] [36]	
Processing type	Characteristics of products [7]	[7, 27, 29, 36]	

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Used algorithm	Characteristics of products [7]	[7, 27, 29, 36]	
Number of pages of documents Number of displays and queries	Characteristics of products [7] Characteristics of products [7]	[32, 36]	
Number of personnel	Staff skill level [7]	[7, 26]	
Experience in similar project	Staff skill level [7]	[7, 32]	
Train and education staff Formal training	Staff skill level [7] Total methodology [33]	[7, 26][33]	
Type of computer used	Computing platform [7]	[26, 28, 32]	
Network type	Computing platform [7]	[7, 27]	
Requirement volatility Amount of requirements rewritten	User attributes [7] Requirements [33]	[27, 32, 33, 35] [37]	Productivity increases with accurate and stable requirements specification. Project related factors (under project management control) [33]
Interface complexity	User attributes [7]	[27, 32, 36]	
User participation in specification Client vs ITT specification	User attributes [7] Requirements specification [33]	[27, 32, 33]	Productivity increases with accurate and stable requirements. Project related (controllable) [33].
User originated design changes Customer initiated design changes	User attributes [7] Cumulative complexity [30]	[27, 30, 32]	

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
User experience in application Client Experience	User attributes [7] Client Interface [33]	[27, 33]	Productivity increases with experience. Product related (non-controllable) [33].
Management commitment	User attributes [7]	[27]	
Customer interface complexity	Cumulative complexity [30]	[30]	
Internal communication complexity	Cumulative complexity [30]	[30]	
External communication complexity	Cumulative complexity [30]	[30]	
Programmer experience with machine	Cumulative experience [30]	[30]	
Team previously worked together	Cumulative experience [30]	[30]	
Tree charts	Total methodology [30]	[30]	
Top down design hline Design formalism	Total methodology [30] Total methodology [30]	[30] [30]	
Formal documentation	Total methodology [30]	[30]	
Code reading	Total methodology [30]	[30]	
Formal test plans	Total methodology [30]	[30]	
Unit development folders	Total methodology [30]	[30]	
Number of resource constraints	Resource constraints [33]	[33]	Productivity decreases with the presence of two or more resource constraints. Product related (non-controllable) [33]
Size		[33, 34]	Productivity decreases as the number of development statements increases. Product related (non-controllable) [33]

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Client participation	Client Interface [33]	[33]	Productivity increases with participation. Product related (non-controllable) [33]
HW concurrent with SW development		[33]	Productivity decreases with concurrent hardware development. Project related factor (controllable) [33]
Development computer size		[33]	Productivity increases as computer size increases. Project related factor (controllable) [33]
Personnel experience		[33]	Productivity increases with more experienced programming personnel. Project related factor (controllable) [33]
Project duration		[34]	Lowers productivity [34]
Execution time constraints		[34]	Lower time constraints increase productivity [34]
Moving window	Historical data [38]	[38]	Considering the chronology of projects when using historical data [38]
Level of detail		[6]	Planning at a more detailed level (shorter activities, smaller tasks) results in better data for estimation and reduces statistical errors [6]
Defects		[37]	To estimate rework
Rework		[37]	
Unclear project definition	Project uncertainty [6]	[6]	
Low project importance	Project uncertainty [6]	[6]	
Technology uncertainty	Project uncertainty [6]	[6]	
Estimation goals	Estimation development [6]	[6]	
Team focused processes	Estimation development [6]	[6]	
Participation of other groups	Estimation development [6]	[6]	

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Concurrency	Work assignment factor [24]	[24]	"the degree to which those team members work together or separately" "increased concurrency (reflecting a higher degree of team collaboration) resulted in greater development effort". However, working together increases effectiveness. Allowing team members to focus on a smaller number of tasks improves effort. [24]
Intensity	Work assignment factor [24]	[24]	Degree of schedule compression, i.e. "to which a module's development schedule is expedited. A module with a high intensity level was worked on with sharp focus and few or no hiatuses, while a low intensity level would be associated with a module that may have sat untouched for long periods of time." More compression of the development schedule of modules improves effort. "development effort was found to decrease as intensity increased." When intensity is too high then it may have the opposite effect [24]
Fragmentation	Work assignment factor [24]	[24]	"degree to which team members' time is fragmented over multiple modules" "development effort is found to increase with fragmentation." Breaking down work to tasks that can be accomplished individually improves development effort. [24]
Actors classification	UPC estimation factor [8]	[8]	
Use cases classification	UPC estimation factor [8]	[8]	Based on average of transactions[24]
Number of new or modified actors	UPC additional factors [8]	[8]	
Transaction	UPC additional factors [8]	[8]	Each counted as one use case
Alternative flow	UPC additional factors [8]	[8]	Each counted as one use case

Continued on next page

Table A.2 – *Continued from previous page*

Name	Definition	Ref.	Comments
Special rules for exceptional flows, parameters and events	UPC additional factors [8]	[8]	
Number of points for modification use cases	UPC additional factors [8]	[8]	
Scale Factors (five)			
Predecentedness : prec (CII)		[31]	Previous experience of the organisation
Development flexibility: flex (CII)		[31]	Degree of flexibility in the development process
Risk resolution: resl (CII)		[31]	Extent of risk analysis carried out
Team cohesion: team (CII)		[31]	How well they know each other and work together
Process maturity: pmat (CII)		[31]	Process maturity of the organisation

A.3 Factors Related with the Project Execution

The next table presents the factors that can cause effort estimation deviations.

Table A.3: Factors causing effort estimation deviations.

Name	Definition	Ref.	Comments
Accuracy of an estimation model		[22]	Estimation ability factor
Project management (cost control) ability	to manage the project to the budget [22]	[22]	Estimation complexity factor
Project member skill		[22]	Estimation complexity factor

Continued on next page

Table A.3 – *Continued from previous page*

Name	Definition	Ref.	Comments
Completeness and certainty of information	"measurement error of input variable [22]"	[22]	Estimation complexity factor
Inherent project execution complexity	"Innovative projects, e.g., utilizing "leading edge" technology, and projects developing complex functionality, are inherently more difficult to estimate than repeating or simple projects. Another example of inherent project complexity is size (large projects are more difficult to estimate). [22]"	[22]	Estimation complexity factor
Project priorities	"Projects with a strong focus on time-to-market, for example, typically have less accurate estimates than those with a focus on cost control. [22]"	[22]	Estimation complexity factor

Continued on next page

Table A.3 – *Continued from previous page*

Name	Definition	Ref.	Comments
Logging problems	"Lack of proper logging routines for the actual use of effort may result in there being differences in activities included in the measured actual effort, or may affect whether overtime is recorded or not. [22]"	[22]	Measurement process factor We will have to exclude from our analysis projects were those problems occur. This is a threat to the execution of the case study itself, the organisation may not have sufficient projects were time is accurately logged and choosing only the only the ones that do it can bias the study, because those may be the single projects that use certain effort estimation methods that require more discipline.
Difference between planned and actual output/process	"Software projects may experience increases or reductions in functionality. Similarly, the project may not conduct all planned quality assurance activities or deliver the planned quality. Differences in estimation error may be caused by these differences between planned and actual output/process and not, for example, estimation ability. [22]"	[22]	Measurement process factor We will verify differences between planned functionalities and actual functionalities (compare proposal, requirements and delivery, ask confirmation to project members), quality activities (verify verification activities and ask confirmation to testers).
Design tool		[39]	Good design tools increase productivity (generation of code).
New project members in the middle of project		[39]	Generally slows down projects due to the learning curve

Continued on next page

Table A.3 – *Continued from previous page*

Name	**Definition**	**Ref.**	**Comments**
Availability of resources	Project uncertainty [6]	[6]	
Instability Project uncertainty [6]		[6]	
Client preparedness	Project uncertainty [6]	[6]	
Customer control	Estimation management [6]	[6]	
IT unit control	Estimation management [6]	[6]	
Reporting frequency	Estimation management [6]	[6]	
Team performance assessment	Estimation management [6]	[6]	
Risk assessment	Estimation management [6]	[6]	
Functionalities		[22]	Planned and actually delivered
Defects	Source of unplanned work[37]	[37]	
Realistic expectations	Client	[40]	
Frequency of plan update		[40]	
Frequency of progress control		[40]	