An MDA Proposal To Integrate the Measurement Lifecycle Into the Process Lifecycle

Ayman Meidan[1,3,*], J. A. García-García[1], Isabel Ramos Ramos[1], David Lizcano[2] and María José Escalona[1]

[1]*University of Seville, Avenida Reina Mercedes s/n, Seville, 41012, Spain*
[2]*School of Computer Science, Madrid Open University (UDIMA), Collado Villalba, 28400, Madrid, Spain*
[3]*Aragón Institute of Technology, Calle María de Luna 7, 50018 Zaragoza, Spain*
E-mail: Ayman.meidan@gmail.com; julian.garcia@iwt2.org; iramos@us.es; david.lizcano@udima.es; mjescalona@us.es
[*]*Corresponding Author*

Received 12 March 2021; Accepted 22 July 2021;
Publication 22 October 2021

Abstract

Context: Measuring the Software Development Process (SDP) supports organizations in their endeavor to understand, manage, and improve their development processes and projects. In the last decades, the SDP has evolved to meet the market needs and keep abreast of modern technologies and infrastructures. These changes in the development processes have increased the importance of the measurement and caused changes in the measurement process and the used measures. *Objective*: This work aims to develop a solution to support the measurement activities throughout the process lifecycle. *Method*: Study the current state of the art to identify existing gaps. Then, propose a solution to support the process measurement throughout the SDP lifecycle. *Results*: The proposed solution consists of two main components: (i) Measurement lifecycle, which defines the measurement activities throughout

Journal of Web Engineering, Vol. 20_7, 2081–2130.
doi: 10.13052/jwe1540-9589.2074

the SDP lifecycle, (ii) Measurement definition metamodel (MDMM), which supports the measurement lifecycle and its integration into the process lifecycle. *Conclusion*: This proposal allows organizations to define, manage, and improve their processes; the proposed information model supports the unification of the measurement concepts and vocabulary. The defined measurement lifecycle provides a comprehensive guide for the organizations to establish the measurement objectives and carry out the necessary activities to achieve them. The proposed MDMM supports and guides the engineers in the complete and operational definition of the measurement concepts.

Keywords: Software development process, process measurement, process metrics and indicators, measurement lifecycle.

1 Introduction

Defining and improving the development process is one of the most important strategies used by organizations to enhance productivity and improve the quality of the developed software. The development process is the primary guide for the management of the work teams and the production process. It is also used as a basis for project planning and monitoring. Defining, monitoring, and improving the software development process (SDP) aims to produce high-quality software products and more predictive and productive projects.

Software development is considered to be comprised of three essential components: products, processes, and resources [1]. Developing software is a long, costly, and complex process. The outcome of this process is not only the final product but the production of many intermediate and supplementary artifacts during the development endeavor. The quality of this development process significantly impacts the quality of the resulting product [2–4].

Measuring the SDP and its outcomes is the only way to gain knowledge about them. Besides, the obtained measurements could be used in models for prediction purposes. Moreover, software process measurement supports better understanding, evaluation, and control of the development process, project, and the resulting product. Measurement also enables organizations to have insight into their processes, predict, and improve their quality and performance, which gives organizations a better position to make appropriate and informed decisions as early as possible during the development process [5].

In the last decades, SDP has evolved to meet the market needs and to keep abreast of modern technologies and infrastructures that have influenced product development and its use. These changes in the development processes have increased the importance of the measurement [6] and caused changes in the measurement process and the used measures [7].

For instance, cloud computing allowed to merge software development, deployment, and operation in what is known as DevOps. Measurement is one of the four DevOps perspectives (Collaboration culture, automation, measurement, and sharing) [8]. In this context, measurement promotes communication and the common understanding between development and operations. On the other side, today's software is increasingly developed by teams working in different geographic locations, time zones, and cultures. Management of these kinds of projects is more challenging and complicated than traditional on-site development. The measurement is an essential element for the success of these development projects [7].

These evolutions in the development process, technologies, and infrastructures create new challenges and obstacles for the measurement in data collection, storage, analysis, interpretation, and making decisions based on the measurement results. These challenges and difficulties emphasize the importance of the measurement in the context of the SDP.

This paper proposes a comprehensive measurement life cycle that addresses all activities and artifacts related to the measurement of the SDP. Furthermore, this work uses the Model-Driven Engineering (MDE) paradigm [9] to integrate the measurement process into the process lifecycle in a way that merges the measurement concepts and activities with the process lifecycle activities.

Therefore, we propose three metamodels; the Measurement Definition Metamodel that allows the definition and modeling of the measurement concepts through the process modeling phase, the Measurement Execution Metamodel, which supports the measurement issues during the process execution phase. And the monitoring metamodel that supports the monitoring and reporting of the measurement data.

This work also uses the MDE transformation rules to derive the necessary measurement concepts and artifacts to support the measurement process throughout its lifecycle. To this end, we propose a transformation process to derive the measurement execution model, the monitoring model, and the measurement documentation from the measurement definition model. This transformation process supports the automation of the measurement process.

The result of this work is a theoretical solution guided by models to improve and automate the measurement of the software processes and a software tool to support the application of this theoretical solution in practical environments.

The measurement life cycle proposed in this solution defines the measurement phases and activities in a more complete and specific way compared to the previous proposals. Moreover, the measurement information model presented in this solution facilitates and guides process engineers to define the measurement concepts in a practical and operational form. Furthermore, this proposal uses the MDE paradigm to support and automate the integration of measurement concepts, activities, and artifacts into the SDP activities.

The remainder of this paper is organized as follows: The following section discusses the related works. Section three describes the identified gaps and the established objectives. Section four presents the components of the proposed solution. And section five describes the development of the tool, which allows the practical use of the proposed solution. Section six describes the validation project and the results obtained from this experience. And finally, section seven states the conclusions.

2 Related Work

This section is divided into two parts: the first part presents the previous research carried out by the authors to comprehend the current state of the art and to discover the existing gaps in the domain. The second part discusses some of the existing proposals, modeling languages, and tools related to this work. These proposals were identified and studied during the previous research carried out by the authors. Furthermore, this section describes the process lifecycles and measurement lifecycles found in the literature.

2.1 Understand the Current State of the Art

The first study performed by the authors to understand the current state of the domain is a survey on the existing open-source Business Process Management Suites (BPMS) [10]; this study aims to investigate to what extent the existing BPMSs support the process lifecycle. Furthermore, provide a guide for the organizations to plan and compare the existing BPMSs, which allows them to discover which BPMS best meets their process management needs. The findings of this study indicate a lack of the definition and integration of the Process Performance Indicators (PPI) into the process model and linking the PPIs with the service level agreements.

These findings prompted the authors to perform the second study, a Systematic Mapping Study [11] that focuses on the measurement of the SDP and its implementation projects, mainly to give insight on the measurement of the "Project" and "Process" entities. It identifies, reviews, and classifies the main proposals found in the literature.

The authors carried out these studies to identify and understand the existing proposals and tools in the market and the literature. The results of these studies reveal the lack of support for measurement issues in process modeling and management tools.

The survey results demonstrate the weakness in defining and integrating the measurement into the process life cycle; most of the investigated proposals do not support the definition and integration of the process measurement. This integration promotes process monitoring and improvement.

Furthermore, the mapping study demonstrates the scarcity of research on defining the measurement in the form that allows its integration into the process lifecycle. This study also reveals that the definition of measurements in a complete and operational form and considering the measurement issues in all the process stages is essential for strengthening process improvement and project management.

2.2 Related Proposals

This section describes the existing research attempts to define and integrate the measurement into the SDP, also reveals how the main process modeling languages and tools support and integrate the measurement issues. Moreover, this section outlines the main existing process and measurement lifecycles.

2.2.1 Relevant research, modeling languages, and tools

Measurement is essential for the quantitative management and improvement of the SDP, so it has gained significant interest from researchers and practitioners. There are many proposals in the literature related to measurement definition, modeling, and execution. This section focuses on the model-based proposals.

In [12] authors present a metamodel based proposal for software process modeling. This proposal does not define the measurement as a process element, but the authors mention the necessity to measure the different process elements during the process execution for monitoring purposes. They also discuss the need to apply changes to the process elements to support its measurement (e.g., add some attributes to the process elements).

In [13–15], the authors propose a measurement framework based on Model Driven Architecture (MDA) [16] to measure any software entity (e.g., database structure, process model, and requirement document) based on the metamodel that represents them. They also present a graphical notation language that allows the users to define software measurement models based on software measurement ontology. This work focuses mainly on measuring model elements based on their metamodel (e.g., count the number of tables in a relational database scheme). Thus, this proposal does not focus on measuring the process execution perspective, such as the elapsed time to perform an activity.

In [17], the authors present an approach to combine different metamodels (e.g., SMM [18] and SPEM 2.0 [19]) to model the process and the measures to provide control over the execution of processes. This approach allows the definition of measures only for processes and task elements. Still, it does not allow the process modeler to model the measures in an explicit and operational form within the process model.

In [20], the authors present a model-driven approach for defining, executing, and monitoring the SDPs; it supports the automatic collection of quantitative measures during project execution. The authors describe a metamodel to define the measures. This approach does not define the measures explicitly in the process model, does not consider the manual measures, and does not measure the process artifacts. Furthermore, the measure definition does not address how the values of the measures will be analyzed and used.

The authors in [21] provide a metamodel and tool for the definition and the design-time analysis of PPIs independently of the language used to model the process. This proposal does not reflect the relation between the information needs, the indicators, and the data collected to satisfy this information needs. Moreover, this proposal focuses only on the measures that will be collected automatically; does not support the definition of the manual measurements (e.g., specifies the necessary methods and tools to perform the measurement activities). Furthermore, the proposal does not allow the description of context data to be collected with the measurement value.

In the proposal [22, 23], the authors present a metamodel to define the development process. This metamodel defines the measure as a process element; the proposal derives the process execution model from the definition model. Still, the resulting model does not include the measure element defined in the definition model. This proposal could be developed by extending the metamodel to define the measurement concepts (e.g., information needs), and by adding more attributes (e.g., performer role, unit, context,

collection method, etc.) to the measure element, and also by representing the measure element in the process execution model.

The review of these proposals shows the following main lacks: defining the measurement in an explicit and operational form, (ii) modeling the measurement concepts in the process modeling phase, (iii) defining measurement for the process execution perspective, (iv) establish a relation between the measurement objectives, concepts and the collected data to satisfy these objectives.

On the other side, the industrial standard BPMN 2.0 [24] does not define the measures as a process element. The commercial implementations of this standard (e.g., Bonita BPM [25]) allow the modeler to define an attribute to be measured but does not define the measures as an element to allow the modeler to describe it in an explicit and operational way. SPEM 2.0 [19] defines the measure as a process element in a basic and abstract form. Wherefore, the process modeling tools which use SPEM 2.0 metamodel (e.g., EPF [26] and RMC [27]) do not support modeling the measures operationally and explicitly within the process model.

These academic and industrial works fail to integrate the measurement into the process lifecycle in such a form that allows: (i) the process engineer to define and model the measurement concepts (e.g., information needs, performer, procedure, and context) in operational form during the process definition and modeling phase. (ii) using this definition in the process deployment phase to perform the necessary configurations to collect and store the measurement values. (iii) collecting the measures data during the process execution phase according to the measure's definition. (iv) analyzing the measured data during the process monitoring and analysis phase according to the method indicated in the measure's definition. Furthermore, (v) reporting the measures and its analyses to the indicated role to determine the necessary actions to control, optimize, and improve the process.

2.2.2 Process improvement and lifecycles

A lifecycle can be defined as a series of activities grouped in a set of phases – each with a specific focus – performed to achieve a specific and integrated objective. Given the wide range of application areas, different views of the process lifecycle have been proposed over the past decades. The most recent process lifecycles are summarized below.

Authors in [28] propose a global process revision cycle to create value for organizations. To do this, they contemplate modeling processes as the first step to achieve this goal. In this way, before initiating any design or process

review, the organization must decide the scope of its initial activities. The process lifecycle proposed by these authors is based on the following nine phases: discovery, modeling, simulation, deployment, execution, monitoring, analytics, optimization, and refine.

On the other hand, in [29, 30], the authors establish a process lifecycle that is much more compact than that presented in the previous proposal. In this case, the lifecycle is based on four phases: process design, system configuration, process enactment, and diagnosis.

2.2.3 Measurement process lifecycles

The term measurement lifecycle refers to the entire phases of the measurement process (e.g., measurement definition, application, and the exploitation of the measurement result) [31]. This process aims to collect, analyze, and report objective data and information to support effective management and demonstrates the quality of the products, services, and processes. (ISO/IEC/IEEE 12207-2017-International Standard – Systems and software engineering – Software lifecycle processes 2017, ISO/IEC/IEEE 15288-Systems and software engineering System lifecycle processes 2015).

Over the last decades, several authors have identified and described phases of the measurement process. The main and most recent proposals are summarized below.

Jacquet et al. [34] have decomposed the measurement lifecycle into four successive steps: *Design of the measurement method*. This step includes: defining the measurement objectives, define the measurement object, characterize the measurement concept, and defining the assignment rules. *Measurement method application*. In this step, the measurement data are collected, and the measurement methods – defined in the previous step – are performed to produce the measurement results. *Measurement result analysis*. The results obtained in the previous step are documented, evaluated, audited, and analyzed in this step. *The exploitation of the result*. In this step, the measurement results are used in several forms (e.g., characterizing and predicting purposes).

In a similar form, authors in [35] have divided the measurement process into four steps: definition, collection, analysis, and in the last phase, the analysis results are used to control and improve the process.

On the other hand, in [36], the authors propose a measurement lifecycle comprise of four phases: *Definition*. During this phase, the measures are identified, defined, and linked with the process objectives. *Measuring*. Where the data is gathered. *Analysis*. In this phase, the measured values are compared

with the target values, and the causes of any unexpected value are identified and r*eported*. In this phase, the analysis results are summarized and reported to the users.

Furthermore, the recent version of the standard ISO 15939-2017 [37] defines a measurement process of four phases: *Establish and sustain measurement commitment.* In this phase, the measurement requirements and scope are defined, the management committee is established, and resources are assigned for the measurement activities. *Prepare for measurement.* This phase includes several activities: Define the measurement strategy and identify & prioritize the information needs. *Perform measurement:* which includes collecting, storing, and verifying data. *Evaluate measurement:* this phase emphasis the quality of the measurement process and the information needs.

3 Problem Definition, Objectives, and Influences

In the past years, the software engineering community has proposed many methods, standards, and techniques (e.g., GQM, PSM [38], and ISO 15939) to guide the selection and definition of the measurement concepts and to optimize the measurement process. Unfortunately, most of these methods and processes stop at the point of selecting and identifying the measures and the measurement concepts that satisfy different needs (e.g., monitoring, controlling, estimating, and improving). However, they do not focus on defining the measurement concepts (e.g., indicators, measurement method, and context) in the form that support the measurement process throughout its lifecycle [39]. Previous studies conducted by the authors and the relevant proposals discussed in the previous section show that this situation remains to date.

Defining the measurement concepts in an unambiguous and rigorous (operational) form is essential to support collecting, storing, and analyzing the measurement values. Moreover, it promotes the interpretation and reporting of the measurement results in the form that support engineers and managers to adopt quantitative management, make informed decisions, and develop the improvement plan. Furthermore, the operational definition of the measurement concepts motivates and supports integrating the measurement into the software process [40].

After introducing the importance of the measurement process and its impact on the SDP, the following section summarizes the problems addressed in this work:

The first problem is related to the definition of the measurement concepts in the form that support the measurement throughout its lifecycle. Defining the measurement concepts in such form supports the integration of the measurement into the SDP.

This work tries to answer several questions to address this problem, among them: (i) What are the essential measurement concepts? And what are the necessary aspects (e.g., unit, scale type, performer, and context data) to define these concepts operationally? (ii) How to enrich the definition of the measurement concepts in the form that support their integration into the process lifecycle?

The second problem is related to integrating measurement issues (e.g., concepts, artifacts, and activities) into the process lifecycle. To address this problem, this work focuses on (i) Identify the software process and the measurement process lifecycles. (ii) Define the main activities of these lifecycles. (iii) Integrate the measurement lifecycle into the software process lifecycle.

The third problem is related to the necessity to provide a tool to support the management of both lifecycles (i.e., software process and measurement process) [41] in the form that enhances their integration throughout their lifecycles. Resolving this problem requires the following: (i) Study the existing process management tools. (ii) Develop a solution to integrate the management of the measurement process into these tools.

3.1 Main Objectives

After defining the scope of the problem, the main objectives are described below.

The first objective is ***defining*** the main measurement concepts and ***identifying*** the characteristics that should be satisfied to define them operationally (in the form that supports the measurement throughout the measurement lifecycle). **The second** objective is ***defining*** the measurement lifecycle and ***integrating*** it into the process lifecycle. For this purpose, we propose three metamodels. (i) is the ***Measurement Definition Metamodel (MDMM)*** which allows the definition and modeling of the measurement concepts through the process modeling phase in the form that integrates these concepts into the process lifecycle. (ii) the ***Measurement Execution Metamodel,*** which supports the measurement during the process execution phase (e.g., collecting and validating the measures data). (iii) the **monitoring metamodel,** which supports the monitoring and reporting of the measurement data. And finally, ***merging*** the measurement metamodels with the process metamodels. **The**

third objective is ***defining*** the required *transformation rules* to derive the necessary measurement artifacts (e.g., execution and monitoring models and measurement documentation) from the measurement definition model.

The fourth objective is ***developing*** a *tool* to support the practical use of the proposed solution. This tool allows the process engineers to (i) Define and model the process and its related measurement concepts. (ii) Execute the process considering the measurement issues. (ii) Use the measurement data to support process management and improvement. And **the last objective** is ***validating*** the proposal by applying it in a real environment and evaluating the results of this experience.

To increase the paper's readability, we have excluded the measurement execution metamodel, monitoring metamodel, and the MDE transformation rules. These metamodels and transformation rules will be included in future works.

3.2 Influences

The authors' previous works influenced the development of this work, especially the framework: Product Lifecycle Management for Business-Software (PLM_4BS) [42, 43].

PLM_4BS is a model-driven based framework that aims to model and manage software processes. It defines metamodels or domain-specific languages to define and executes processes. Furthermore, it establishes systematic protocols to support the necessary transformations between the process models. This framework was developed to support the evolution of the NDT (Navigational Development Techniques) [44, 45] methodology, which supports the software development lifecycle.

PLM_4BS is based on a continuous improvement lifecycle. This lifecycle comprises four phases: *modeling phase*, *execution and orchestration phase*, *monitoring phase*, and *the continuous improvement phase*. Currently, the framework provides support for the first two phases only (modeling phase and execution & orchestration phase). Therefore, the main goal of this work is to support the investigations that aim to cover the rest of the process lifecycle defined by the PLM_4BS (Monitoring and improvement). Precisely, this work is part of the research that seeks to promote the integration of the measurement issues into the process lifecycle (e.g., design, modeling, execution, and monitoring) to support the process monitoring and improvement.

After presenting the problem addressed in this work, the main objectives, and the key influences which guided and motivated this work, the next section introduce the proposed solution to achieve the established objectives.

4 The Proposed Solution

In previous sections, we have defined the problem addressed by this work and the main objectives established to resolve it. This section presents the proposed solution to support the measurement throughout the process lifecycle.

4.1 Define and Integrate the Lifecycles

As described in previous sections, several proposals have been presented to determine and describe the main stages of the process lifecycle. Each of these proposals focuses on a specific perspective according to its context of use. By studying these proposals, we find that the main activities of the process lifecycle can be categorized into the following stages: Process discovery and Design, Modeling and simulation, Deployment, Execution, Monitoring and analysis, and finally, the process Continuous improvement phase.

4.1.1 Measurement lifecycle

In recent years, several proposals have been presented to define the measurement lifecycle, main proposals are described in previous sections. Below, we propose a more comprehensive measurement lifecycle. As shown in Figure 1, the proposed lifecycle defines all the activities related to the measurement process:

The first phase of the proposed measurement lifecycle is **measurement Selection and Definition**: in this phase, the measurement objectives and concepts are defined. The engineers use the measurement selection and definition methods (e.g., GQM and PSM) to choose the appropriate measurement concepts that satisfy the measurement objectives.

Modeling phase: In this phase, the measurement definition resulted from the previous phase is represented in a formal and operational form and integrated into the process model. The defined measurement concepts and their relationships could be analyzed and optimized (e.g., for consistency, correlation, and causality issues) [21, 46] in this phase.

Configuration phase: In this phase, the measurement definition established in the previous stage is used to perform the necessary configurations to achieve the measurement activities; the process execution environment is prepared to allow the collection, validation, storing, and reporting the measurement data.

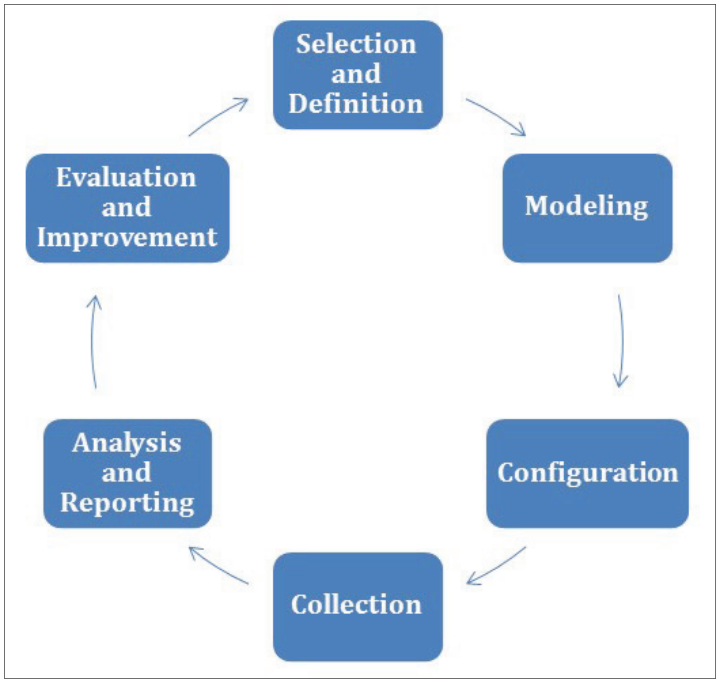

Figure 1 Measurement process lifecycle.

Collection phase: During the process execution, the defined measures are gathered, validated, prepared, calculated, and stored according to the definition established in the first phase.

Analysis and Reporting phase: In this phase, measurement data is analyzed and reported according to the measurement definition; the resulting information is generated, formed, and communicated to the predefined roles indicated in the measurement definition.

Evaluation and improvement phase: In this phase, the measurement process is evaluated, lesson learned, and feedback about the process is gathered and assessed to discover improvement opportunities. There are many validation and evaluation frameworks (e.g., [31, 47, 48] and the industry standard ISO/IEC/IEEE 15288:2015 [49]) that could be used to validate the measurement from two main perspectives: *Relative verification* which evaluates the design objectives of the measurement, the necessary precision, the maturity

of the available knowledge about the attribute, etc. And the *Absolute verification*, this perspective focus on evaluating the measurement principle in itself; to ensure that the process is characterizing what it intended to measure [31].

Below, Table 1 presents a comparison between the previous proposals and the proposal presented in this work. It demonstrates the measurement activities defined in the different measurement lifecycles.

As shown in the previous table, this work identifies and defines the measurement modeling phase. This phase is not described in the earlier proposals. The activities related to this phase support the formal definition of the measurement concept, analyze and optimize these concepts, support the process simulations, and support integrating these concepts into the process model.

Furthermore, this work identifies and defines the relationship between the measurement phases and the process lifecycle phases, allowing the integration of the measurement lifecycle activities and artifacts into the process lifecycle, as shown in the next section.

4.1.2 Integrating the process and measurement lifecycles

The measurement process is closely related to the SDP since the measurement process supports the software process in various phases throughout its lifecycle, such as the design and simulation phase and monitoring and improvement phase [14]. Therefore, the integrated management of both lifecycles is essential to transform the organization toward quantitative management (management by facts). This integration defines the measurement activities which should be carried out at each stage of the development process. This integration also encourages people to adopt the measurements as part of their work [50].

The potential benefits of this integration include: (**i**) The integration of the measurement process into the development process establishes a connection (Figure 2) between the two parts of the development process (that is, the management process and the production process [51]). This connection allows the data flow from the production process to the management process, which is fundamental for management and decision-making. (**ii**) Minimize the redundancy of the measurements and improve their consistency in the organization. (**iii**) Provide a clear and comprehensive measurement plan at the early stage of the development process. This plan identifies and defines the necessary measurement concepts, activities, and artifacts throughout the process lifecycle. (**iv**) Promotes objective communication between the stakeholders by using common concepts and terminology. (**v**) Defining how

Table 1 Comparison between measurement lifecycles

Activity		Jacquet et al.		Y. Zhang et al.		Del-Río et al.		ISO 15939		This proposal
Define measure. Objectives	✓	Design of the measurement method	✓	Definition phase	✓	Definition phase	✓	– Establish measure. – Commitment – Prepare for measurement	✓	Selection and Definition phase
Define measure. Concepts	✓		✓		✓		✓		✓	
Represent measure. concepts in a formal and operational form									✓	Modeling phase
Analyze and optimize measure. concepts									✓	
Integrate measure. concepts into the process model									✓	
Configure the execution environment to support the measurement activities							✓	Perform measurement phase	✓	Config. phase
Gather, calculate and store measurement data	✓	Measurement Method Application	✓	Collection phase	✓	Measuring phase	✓	Perform measure. phase	✓	Collection phase
Validate measurement data	✓	Measurement result analysis					✓	Perform measure. phase	✓	
Analyze measure. data			✓	Metrics analysis phase	✓	Analysis phase	✓	Perform measure. phase	✓	Analysis and Reporting phase
The resulting information is generated and communicated to predefined roles	✓	Exploitation of the result	✓	Exploitation of the result	✓	Report phase	✓	Perform measure. phase	✓	
Evaluate Meas. process							✓	Evaluate measurement phase	✓	Evaluation and improve. phase
Discover improvement opportunities							✓		✓	

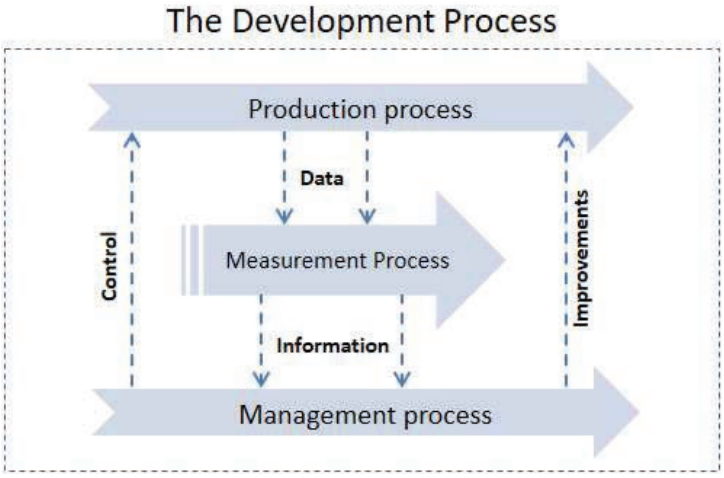

Figure 2 The role of the measurement process in the SDP.

the development process (for example, activities, stakeholders, and results) will be measured at an early stage of the development process promotes the achievement of the process objectives in terms of performance, productivity, and quality, i.e., tell me how you will evaluate me to tell you how I will behave.

This integration could be done by introducing the measurement activities into their corresponding phases in the software process lifecycle and allowing the transition of the artifacts between the two lifecycles. Figure 3 demonstrates the relationship between the software process lifecycle and the proposed measurement process lifecycle. The integration details are described below:

Process *discovery and design*: Throughout this process phase, process engineers define the process's main objectives, activities, roles, and outputs. In collaboration with the process engineers, the measurement team can use these artifacts to (**i**) Define the main measurement goals and concepts. (**ii**) Derive the indicators and measures from these goals using measurement selection methods (e.g., GQM, GQIM, and PSM).

Taking the measurements into consideration at this stage have several benefits: (**i**) allow the management team to communicate their information needs, prioritize their objectives, design the format of the reports, defines the expected values and analysis models, etc. (**ii**) Support the measurement team to a better understanding of the measurement requirements and

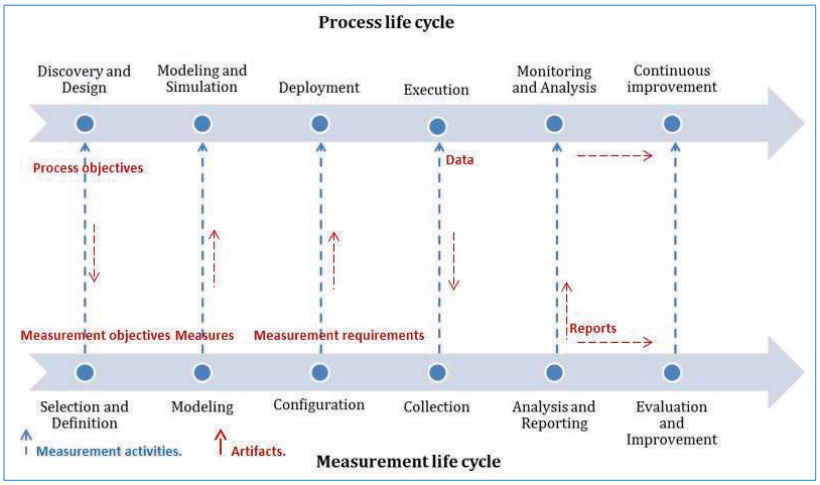

Figure 3 The integration of the measurement lifecycle into the process lifecycle.

objectives. (**iii**) Demonstrate the management's commitment to the measurement processes, which is an essential success factor for the measurement process [52].

The main output of the measurement activities in this phase is a complete and operational definition of the measurement concepts (detailed in Section 4.2). These defined concepts will guide the measurement activities during the next phases of the measurement process.

***Process modeling and simulation*:** The measurement concepts defined in the previous phase are formally defined and integrated into the process model. This formal definition promotes the success of the measurement process [53, 54]. The integration of measurement concepts in the process requires the clarification of the following details [40]: (**i**) What data should be collected (e.g., entity or process element, and attribute). (**ii**) When the data should be obtained (e.g., event or frequency), and (**iii**) The human role responsible for collecting and analyzing the measurement data.

Furthermore, it is necessary to establish the link between the measurement concepts (e.g., measure) and the process element (e.g., entity and attribute), as well as define the interrelations between the different measurement concepts (for example, the information needs, indicators, measures, and human roles) in the form that facilitates its traceability and prioritization.

Moreover, in this phase, the defined measures could support the simulation of the process execution. This simulation evaluates several aspects of

the process for different purposes, such as possible improvements, changes [55–57], and assessment [58]. Besides, the defined measurement concepts could be used to build prediction models to estimate process characteristics (such as resources, performance, and time) and product characteristics like (product size and quality).

***Process deployment*:** In this phase, engineers consider the measurement definition to perform the necessary configuration for collecting, validating, and storing the process execution data; this configuration include: prepare questionnaires and forms to obtain the data, create connections to services and data sources, create a database to store the measurement data, and also perform the required developments for data reporting and visualization.

***Process execution*:** In this phase, the data related to the process execution (e.g., resources, performance, and process outputs) is gathered, validated, and stored to be available for monitoring, control, and improvement purposes.

***Process monitoring and analysis*:** The data collected during the process execution is monitored and analyzed to support process management and control. The following activities will be performed according to the measurement concepts defined in the early stages of the measurement process: (**i**) Provide the information needs, measures, and indicators for the predetermined audience in a periodic manner, (**ii**) Monitor and analyze the measurement data, and (**iii**) Visualize and communicate the data in the form that support the management and decision-making process.

Furthermore, in this stage, the predefined targets of the indicators are compared with the actual values [55], the predefined analysis models and decision criteria are applied to support the management team to analyze the process performance [59], the quantitative management [60, 61], and in-process control.

***Process evaluation and improvement*:** The measurement data could be used in this phase to perform post-mortem analysis and compare the performance and results of the measurement process. Moreover, the measures and indicators can be used in this phase to improve, redesign, and re-engineer the process [62].

4.2 Measurement Concepts

This section presents the measurement concepts, its operational definition and also highlights the relationships among them. These measurement concepts

are identified and operationally defined by the measurement team in the first phase of the measurement process (Measurement Selection and Definition).

In this phase, the Measurement team – with the collaboration of the process engineers – uses the measurement methods to derive the necessary measurement concepts to achieve the measurement goals. These measurement concepts will be used as a guide for the measurement process throughout the rest of its lifecycle. Therefore, selecting and defining these concepts in a complete and operational form is essential, as described in [63].

4.2.1 Measurement information model (MIM)

We propose a Measurement Information Model (MIM) to represents the identified measurement concepts and their relationships. This information model is based on the information model presented in the ISO standard 15939 [37]. The original model was modified to support the objectives of this work (e.g., the integration of the measurement concepts into the process lifecycle) and support the use of the MDE paradigm (e.g., models and transformations).

The proposed information model defines the measurement concepts and also describes the relationships between the information needs (measurement goals) and the necessary objective data (measures) to be collected to satisfy these needs [64]. The MIM shown in Figure 4 demonstrates the proposed measurement concepts and their relationships from the high-level "information need" down to the measurable attributes. The main measurement concepts of this MIM are described below:

***Information needs*:** Represents the required information to track an objective (e.g., improvement or performance target) or constraint (e.g., schedule, effort, or budget).

***Measure*:** It is a value (number or symbol) assigned by mapping rules to characterize some attributes of an entity. Measures could be classified into three types or levels [65]; the first one (base) is used to obtain the data, the second level (derived) is used to prepare the data for the analysis, and the third level (indicator) is used in the analysis that satisfies the measurement requirements or needs: (**i**) *Base measure*: characterize and quantify the extent to which the entity possesses a certain attribute [66], defined procedures are used to determine this degree (e.g., counting the number of defects detected in a specific process phase), (**ii**) *Derived measure*: represents a relationship or algorithm/function between multiple measures [67] (e.g., productivity = size/duration), and (**iii**) *Indicator*: is a measure that provides an estimation or evaluation (using a model and decision criteria) to support

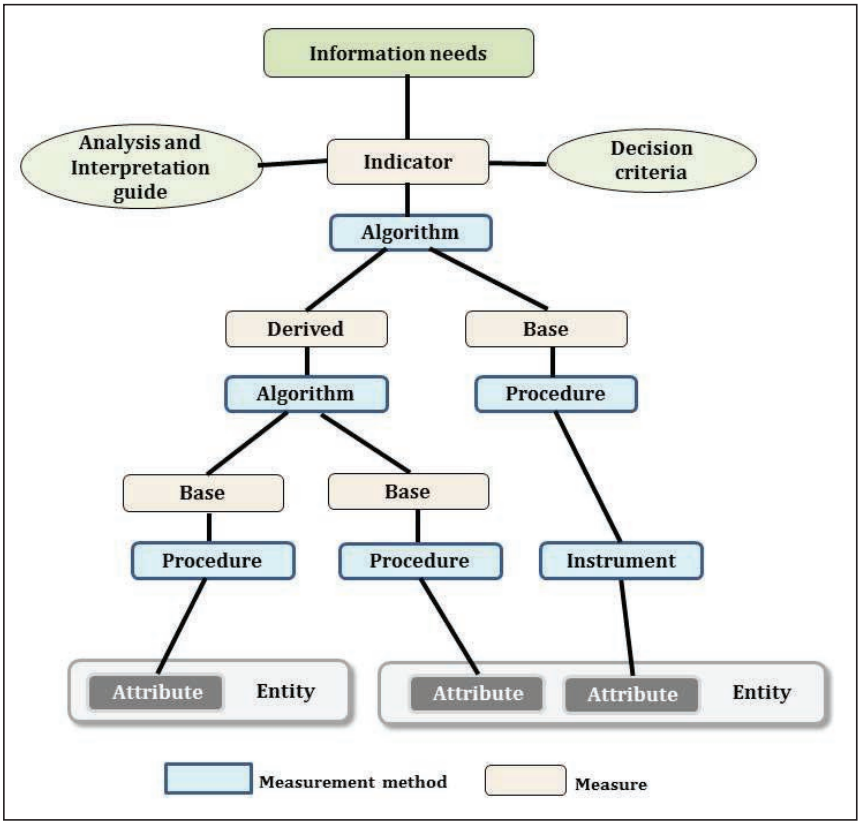

Figure 4 Measurement Information model.

the management in the analysis and decision making [37]. It applies the evaluation/estimation models (calculations or algorithm) to the measures, then displays and communicates the results to the stakeholders. The *decision criteria* (e.g., patterns, thresholds, or target values [68]) provide support for interpreting the indicator value and suggesting actions based on the indicator results.

***Measurement method*:** Provide an operational description of how the measurement value will be obtained (counting rules) by describing the measurement procedure and instrument for the base measures and the algorithm for the derived measures and the indicators, and it involves: (**i**) *Measurement procedure*: Define the steps that should be followed to quantify an attribute. E.g., counting defects or lines of code. (**ii**)

Measurement instrument: Define how the measurement method is implemented to obtain the measurement value [69]. *Examples*: Software program to count the line of code, Person, or program who gets data from a data source (web page, Excel, database, etc.), Questionnaire, and Checklist. (**iii**) *Measurement algorithm*: Define the required operations to obtain the measurement value. E.g., Formula.

Attribute: is a property of an entity, such as the size of a program, the size of the requirement list, the productivity of a team, and the time required to achieve a milestone.

Entity: is an object or event (e.g., process, resource, project, or product). Its attributes should be measured to achieve the measurement objectives.

4.2.2 The operational definition of the measurement concepts

This section introduces the operational definition of the measurement concepts described in the proposed MIM. Next, we describe the proposed aspects that define these concepts in an operational form.

Information needs: ID: Unique identifier, **Title:** Define the subject of the item, **Description:** Provide details to support the understandability and describe the necessity of this item, **Author:** Refer to the role or unit that proposed and following the item, **Priority:** Define the priority of the item [70], **Accessibility:** Define who can access the item, **Version:** Provide traceability information about the item.

Indicator: ID: Unique identifier, **Title:** Define the subject of the item, **Description:** Provide details to support the understandability and describe the necessity of this item, **Information needs ID:** Refer to the information need to be satisfied by this indicator, **Objective:** Define the indicator in natural language (e.g., describe relations). *Examples*: Display Earned value over time, Show the Defect density over time, Show Schedule deviation rate for each phase. Display downtime for each release, Show the mean and standard deviation of all projects productivity values, display process center and limits using defect density values over time, Show the performed activities concerning the planned activities. **Measurement method:** Define mathematical operations and expressions to be used (if necessary) to obtain the indicator results. Examples: Indicator = measure1, Indicator = average (measure_1, measure_2..., measure_n), Indicator = Effort_prod1+ Effort_prod2, Indicator = actual cost/planned cost. **Analysis and interpretation guide:** Provide the necessary details to support and guide the analysis and interpretation

of the indicator results. Could define *Thresholds* (upper limit, center limit, low limit) and *color scale* with the traffic light metaphor [68, 71]. **Decision criteria:** Define actions to be taken based on specific indicator results, **Interpretation:** Provide support to interpret and understand the indicator results (e.g., if there are two consecutive points out of the low or upper limit, then this is a deviation trend, and management actions are needed to investigate this deviation.). **Analyst:** Assign responsibility (role or unit) for analyzing the indicator results. **Responsible:** Assign responsibility (e.g., project manager, product manager) for monitoring the indicator results (the audiences). **Accessibility:** define the role or unit which can access the indicator results [50]. **Priority:** Define the priority of the indicator [70]. **Scheduling:** Define when the indicator is evaluated, analyzed, and reported. **Presentation guide:** Provide a guide to visualize and communicate the indicator results (e.g., XmR chart [72] is recommended to represent data over time (e.g., daily, weekly, or monthly).

Derived measure: **ID:** Unique identifier, **Title:** Define the subject of the item, **Description:** Provide details to support the understandability and describe the necessity of this item, **Measurement method**: Define how the measurement value is calculated. Use an algorithm to combine other measures (based and derived measures). E.g., value = base_m1 + base_m3.

Base measure: **ID:** Unique identifier, **Title:** Define the subject of the item, **Description**: Provide details to support the understandability and describe the necessity of this item, **Entity:** Define the measured entity (e.g., phase, activity, work product, or team), **Attribute:** Define the measured attribute (e.g., cost, effort, size, or progress), **Scale-type:** The scale-type determine the type of operations and transformations that could be applied to the measured value [31]. The most common scale types [37] are nominal, ordinal, interval, and ratio. **Scale:** Define the type of measurement value (e.g., Integers from zero to infinity, positive number, decimals, or label such as experienced, not experienced), **Unit:** A measurement unit determines how the attribute is measured [48]. *Examples*: the size could be measured by the units: number of lines of code, function point, implemented functions/requirements or number of implemented classes, Program correctness, or test case could be measured by the unit: Fault rate. And the effort could be measured using the unit: work hours. **Performer:** Assign responsibility to a role or unit for obtaining the measurement value, **Scheduling:** Define when the measurement value is obtained (collection interval) (e.g., (every

week), or when an event occurs (e.g., activity complete)), **Measurement Method:** Describe how the measurement value is collected or obtained; by defining the measurement procedure and instrument, **Validation guide:** Define how the collected data could be validated for correctness and consistency (e.g., describes the valid data, the range of possible data, or expected values). **Context data:** The context data includes the necessary information to verify, interpret, or evaluate the measurement value [37, 73]. Examples of the context data and its categories: The measured entity/attribute: E.g., when measuring the program size (LOC), it is essential to indicate the programing language used to implement the file. The measurement performer: E.g., name and role. And the environment: E.g., measurement date and time, data source.

Measurement Method: ID: Unique identifier, **Title:** Define the subject of the item, **Description:** Provide details to support the understandability and describe the necessity of this item, **Measurement procedure:** Define steps to obtain the measurement value, **Instrument:** Define how the procedure is implemented, **Algorithm:** Define a formula to calculate the measurement value.

4.2.3 Example of using the proposed measurement concepts in the practice

Based on [74], the following scenario illustrates how to use the proposed concepts in practice. In this scenario, the project manager needs to know the cost situation of the project (e.g., the ratio between allocated and used budgets) in a specific stage of the project.

BPMN 2.0 does not define any element representing the measurement concepts (e.g., information needs, indicator, measure, measurement method).

In the case of **SPEM 2.0,** it defines the "Metric" element. This element allows engineers to define only the measure concept in a simple form: metric name, description, purpose, and constraint.

On the other hand, integrating **our proposal** (e.g., using UML-profile) to a process definition/modeling language (e.g., PLM_4BS) allows engineers to define the measurement concepts and their relationships completely and operationally. In this scenario, our proposal allows the engineers to define the following measurement concepts:

– **Information needs**: "understand the cost situation of the project." This information need is fulfilled by the following measurement concept:

- **Indicator**: "*cost situation,*" which informs the management about the project cost. This indicator defines the following analysis model and decision criteria to satisfy this management requirement.
  - The analysis guide defines three levels for the indicator "cost situation":
    - "Red-unacceptable": means that the cost of the project exceeds the budget.
    - "Green-acceptable," which means the cost is up to 90% of the budget.
    - And "Yellow-warning" if the indicator value is $> 90\%$ and $< 100\%$.
  - The decision criteria associated with this indicator defines the actions that must be taken when a specific criterion occurs:
    - "Red-unacceptable." Call for meeting with project management.
    - "Green-acceptable." Inform management.
    - "Yellow-warning." Inform management and call for a meeting with the project team.

This indicator uses a **derived measure** to evaluate the cost situation by applying the **measurement method Algorithm**, which divides the **base measure** "current cost" by another **base measure,** "planned cost."

While the values of the base measures (the current cost and the planned cost) are obtained using the **measurement methods: procedures** and **instruments**.

4.3 MDE Solution to Support the Measurement Lifecycle and Its Integration Into the Process Lifecycle

The previous section has discussed the integration of the measurement lifecycle into the process lifecycle. This integration implies merging the measurement activities and concepts with the process activities. This section proposes an MDE solution to support this integration. This solution defines (**i**) Metamodels to support the different phases of the measurement lifecycle and its integration into the process lifecycle. It also defines the necessary (**ii**) MDE transformation rules to derive and automatically generate the necessary artifacts throughout the measurement process lifecycle.

Figure 5 shows the proposed metamodels, the relationships between them, and the necessary transformation rules to derive the necessary artifacts to support the different phases of the measurement process.

The first metamodel is the **measurement definition** metamodel (MDMM). This metamodel supports the formal definition of the measurement concepts during the measurement modeling phase. This metamodel will be defined in the next section. The second metamodel (The **measurement execution** metamodel) presents the necessary concepts to integrate the measurement issues into the process execution. This metamodel provides essential information to perform the measurement activities throughout the process execution phase. The third is the **monitoring metamodel**. This metamodel supports the analysis and reporting phase of the measurement lifecycle.

We define two types of transformation rules: (**i**) Model-to-Model (M2M) transformations, this type of transformation uses one (or more) source model to generate different kinds of model(s) in different languages and on different levels of abstraction. We use it to generate the measurement execution model and the monitoring model from the measurement definition model. We also use the (**ii**) Model-to-Text transformations (M2T) to generate the measurement documentation from the measurement definition model, to generate the necessary code to execute the measurement activities from the measurement, and to generate the necessary code for the monitoring panel from the monitoring model.

Figure 5 also shows how the models created with conformance to the first metamodel (i.e., the MDMM) will be used to derive the execution and monitoring models and the measurement documentation. We describe below how these artifacts will be generated:

The measurement **execution model**. This model uses the measurement specifications – defined in the Measurement Definition Model (MDM) – to identify the necessary measurement concepts that achieve the measurement goals (established in the MDM) during the process execution. This model supports the measurement collection phase by defining the required elements to collect, obtain, validate, and store the measurement concepts specified in the MDM. **The monitoring model**. This model is derived from the MDM to define the necessary concepts to monitor the measurement goals. This model uses the dashboard concept as a container for all the measurement goals (i.e., the information needs defined in the MDM) and preserves the relationship between these goals and its related measurement data. **Measurement documentation**. These documents provide the specifications of each measurement

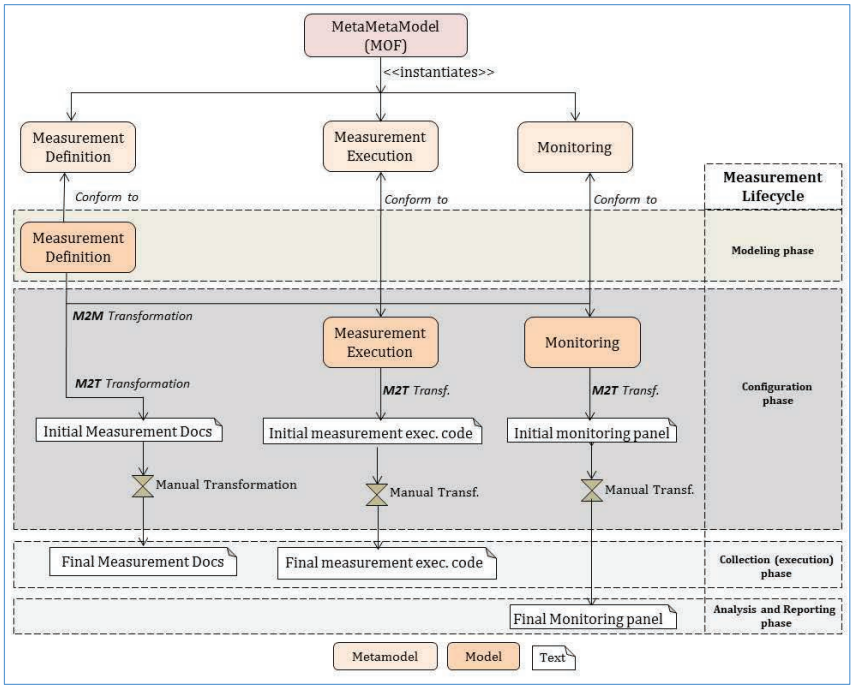

Figure 5 The MDE solution (metamodels and transformation rules).

concept defined in the MDM. These documents will be derived from the MDM using (M2T) transformations.

As mentioned above, this document only covers the measurement definition metamodel. To improve the readability of this paper, the rest of the metamodels and transformation rules will be introduced in future articles. The following section presents the proposed measurement definition metamodel.

The measurement definition metamodel (MDMM) aims to support the measurement modeling phase. It allows the engineers to define the established measurement goals and the necessary measurement concepts to achieve these goals. These goals and concepts were identified in the first phase of the measurement process (Selection and Definition).

As shown in Figure 6, the proposed measurement language is described in a MOF metamodel and presented by the UML class diagram notation. Moreover, we have defined the necessary semantic constraints – as recommended by the Object Management Group (OMG) – using the OCL language

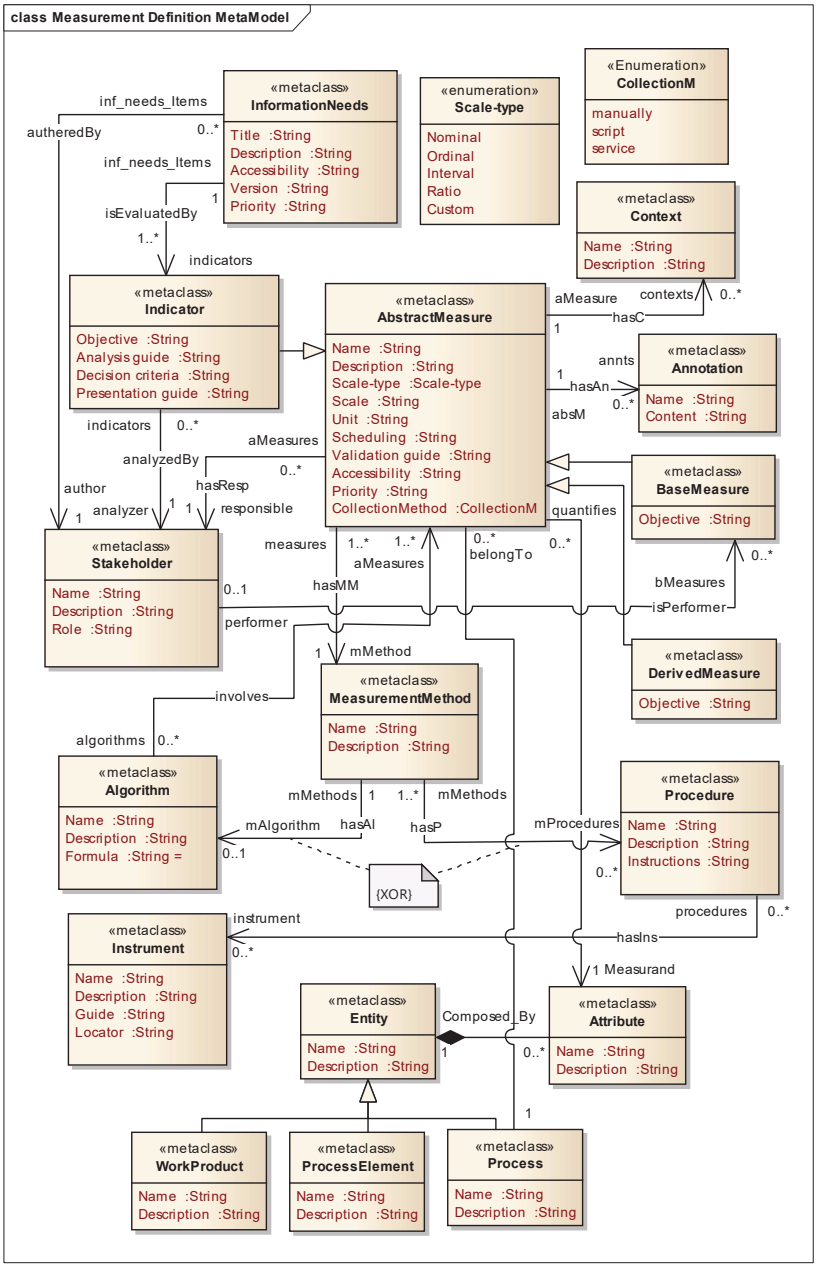

Figure 6 The measurement definition metamodel (MDMM).

ISO/IEC 19507 [75]. We describe below the main elements of the proposed metamodel.

The ***AbstractMeasure*** is the main metaclass in the proposed language. It represents a generalization of the three types of measures (base measure, derived measure, and indicator). This metaclass defines the common attributes (detailed in Section 4.2.2) and relations of these metaclasses. The associations of this metaclass allow the definition of the stakeholder role, the measurement context, the measured attribute, the process to which the measure belongs, and the annotation, which allows adding custom attributes and notes to the measure. It also has an association with the *MeasurementMethod* metaclass to define how the measurement value is obtained.

The ***BaseMeasure*** metaclass represents the measure that quantifies a specific attribute of an entity (e.g., process, process element, or work product). The associations of this metaclass allow the definition of the human role responsible for performing the measurement and a measurement method to obtain the measure's value. This measurement method should define at least one procedure. Expression 1 shows this OCL restriction:

Expression 1. OCL constraint on the metaclass "BaseMeasure".

context BaseMeasure

inv measureHasProcedure : **self**.mMethod.mProcedures->**size()**>=1

The ***DerivedMeasure*** metaclass represents a relationship or algorithm/function between multiple measures (i.e., base measures or derived measures). It has an association with the *MeasurementMethod* metaclass. This association allows the definition of an algorithm to obtain the value of this measure. This restriction is described by expression 2.

Expression 2. OCL constraint on the metaclass "DerivedMeasure".

context DerivedMeasure

inv measureHasAlgorithm : **self**.mMethod.mAlgorithm->**size()**>=1

The ***Indicator*** metaclass allows the evaluation of the measurement objective based on defined analysis rules, also suggests actions based on defined decision criteria. The attribute *Analysis guide* defines the necessary details to support and guide the analysis and interpretation of the indicator results. The attribute *Decision criteria* specify actions to be taken based on specific indicator results. The attribute *Presentation guide* provides a guide about how to visualize and report the indicator results. Moreover, this metaclass defines

associations to specify the analyzer role and the *InformationNeeds* evaluated by this indicator.

InformationNeeds metaclass represents the required information to track a goal or constraint. This metaclass defines associations to specify the author's role and the indicators that evaluate the information needs element.

The metaclass ***MeasurementMethod*** defines how the measurement value is obtained. Its associations allow the specification of the measures, algorithms, and procedures related to this element. It is not allowed to associate procedure and algorithm with the same *MeasurementMethod*. This restriction is defined in the metamodel using the UML logical operator «*XOR*» associated with the *mProcedures* and *mAlgorithm*. This metaclass should be associated with at least one *mProcedures* or one *mAlgorithm.* This restriction is implemented as shown in expression 3.

Expression 3. OCL constraint on the metaclass "MeasurementMethod".

context MeasurementMethod

inv hasProcedureOrAlgorithm :

(**self**.mProcedures->**size()**>=1) **or** (**self**.mAlgorithm->**size()**>=1)

Procedure metaclass defines how the attribute of the entity is characterized. The associations related to this element specify the *Measurement-Method* associated with the procedure and the instruments which implement the procedure (if it exists).

The ***Algorithm*** metaclass defines a relation between measures to calculate the derived measures or indicators. The associations of this element allow specifying the measurement method and the measures related to the algorithm element.

The ***Stakeholder*** metaclass represents the human roles (e.g., performer, responsible, and author) involved in the measurement activities. And, the ***Instrument*** metaclass represents the necessary instruments to obtain the measurement value. Moreover, the ***Context*** metaclass represents the necessary information to verify, interpret, or evaluate the measurement value. The ***Annotation*** metaclass allows the user to add more notes or attributes to define the measure element.

Finally, the enumeration ***Scale-type*** classifies the measurement scale type, and the ***CollectionM*** enumeration defines the possible values of the collection methods

The process engineers will use this metamodel (MDMM) to describe the measurement concepts in the measurement modeling phase. The output

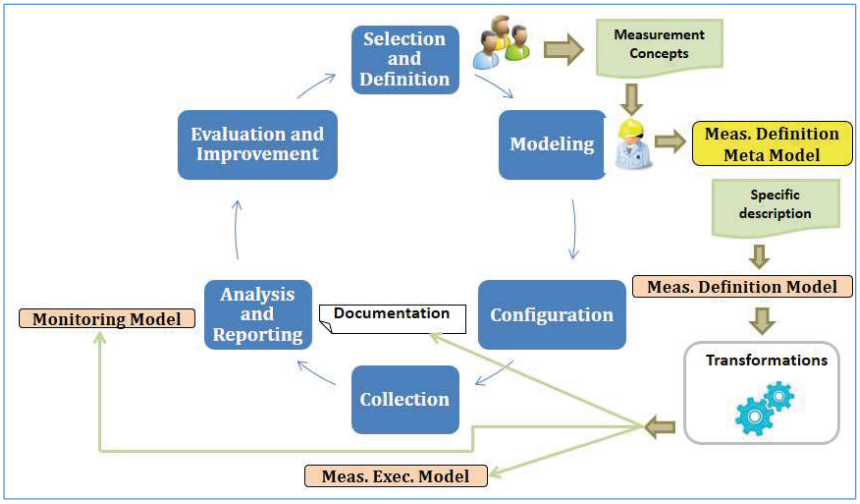

Figure 7 The role of the measurement definition metamodel in the proposed solution.

of this phase is a measurement definition model; this model contains the formal description of the measurement concepts and relations. This data is needed to support the rest of the measurement lifecycle phases. As shown in Figure 7, the measurement definition model will be used to automatically generate – using MDE transformations – the necessary artifacts (execution and monitoring models and the measurement documentation) to support the measurement process and its integration into the process lifecycle.

After defining the MDMM and describing its role in supporting the measurement process lifecycle, the next section describes the developed tool that allows this proposal's practical use.

5 Measurement Modeling Tool

The previous section has presented the proposed metamodel to model the measurement concepts of the software processes. However, to support the practical use of this proposal, it is necessary to develop a tool that allows the engineers to model the measurement concepts. This section presents our Measurement Modeling Tool (MMT).

As mentioned before, this work is influenced by our previous works; This proposal aims to support the modeling and integration of the measurements in our PLM_4BS process modeling tool (see Section 3). PLM_4BS is based on Enterprise Architect (EA) [76]) as a modeling CASE tool. Therefore, the

development of our MMT consists of integrating the proposed measurement definition language into the PLM_4BS process modeling tool (i.e., integrating our measurement definition language into the EA). This integration consists of two steps: (i) Develop a domain-specific modeling notation (i.e., UML profile) which allows the practical use of our MDMM, and (ii) Integrate this notation into the PLM_4BS process modeling language.

5.1 Develop The Specific Language (Measurement UML Profiles)

UML profile provides a usable, expressive, and flexible mechanism to adapt a theoretically defined metamodel with specific constructs for a particular domain [77]. This profile allows the instantiation of the MDMM using a visual notation that can be used by CASE tools (e.g., NDT-Tool [78], IBM Rational Software Architect Designer [79], and Enterprise Architect. The UML extension protocol is based on three basic mechanisms: «Stereotype,» «Tagged value,» and «Constraint.»

The UML profile that implements our MDMM defines a stereotype for each metaclass of the MDMM and includes the required tagged values in each stereotype to represent the attributes of each metaclass in the metamodel. It also adapts the semantic constraints of the metamodel to restrict the behavior of the UML metaclass used.

5.2 Integrating the Measurement Definition Profile Into PLM_4BS Process Modeling Tool

To allow the practical use of the solution proposed in this work, we need to integrate the proposed metamodel into our process modeling tool (PLM_4BS). As mentioned earlier, this tool is based on Enterprise Architect (EA) CASE tool; EA supports the creation of visual instances of the metamodels that describe the process (e.g., software process, clinical guides). To perform this integration, we need to add the UML profile developed in the previous step to the EA.

One of the advantages that EA provides is its extension capacity, which allows the development of Add-Ins. For this purpose, EA provides a Model Driven Generation (MDG) Technology, which allows the development of custom packages and deploys them in the EA project, providing a solution tailored to specific domains or environments.

We have used the MDG technology to develop an Add-In to allow the instantiation of the MDMM proposed in this work. Figures 8 and 9 show the measurement toolbox defined for our MMT.

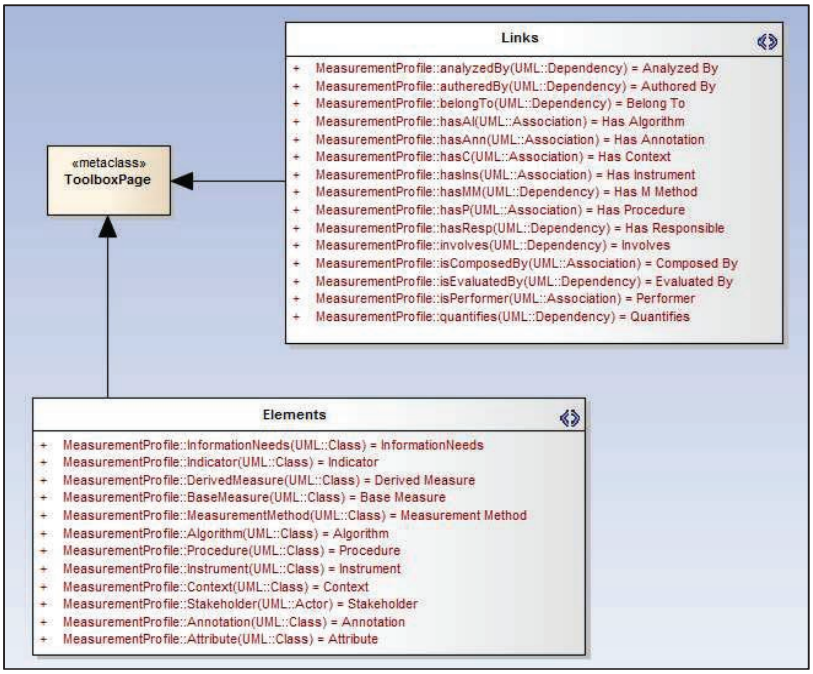

Figure 8 The elements and relations of the MMT toolbox.

6 The Application of the Proposed Solution

The previous section has described the development of our Measurement Modeling Tool and its integration into our process modeling tool (PLM_4BS). This section describes the application and evaluation of our MMT in a real project.

6.1 The Application of the Proposal

On the one hand, we have applied our proposal to several real cases to validate it and ensure that its supports the measurement definition and modeling. For example, due to space limitations, Figure 10 shows part of the measurement model that defines the measurement concepts related to the example described in Section 4.2.3. This figure demonstrates some measurement concepts such as information needs, indicator, measurement method. It also shows the relationships between these concepts.

On the other hand, we have applied and evaluated our proposal in a real project related to the health industry; the IDE_4ICDS project (Integrated

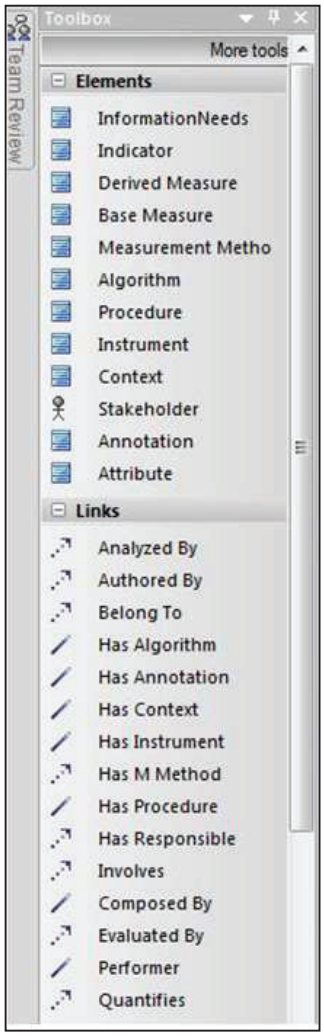

Figure 9 Measurement toolbox.

Development Environment for Improving Clinical Decision Support based on Clinical Guides). This project aims to develop the IDE$_4$ICDS platform, which establishes a real working philosophy oriented to clinical guides [80] and effective, systematic, and automatic mechanisms within the health sector organizations. This approach allows the representation, maintenance, execution of the clinical guides, also capturing feedback about its use to improve

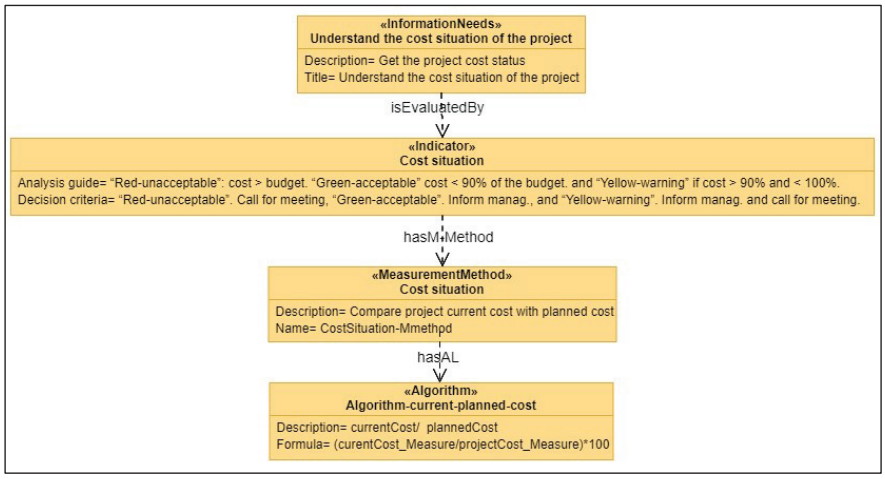

Figure 10 Part of the model that defines the concepts described in Section 4.2.3.

the quality of the health care received by the patient; all using software tools that enable these tasks as well as the interoperability between systems to transfer and share clinical knowledge.

The main objective of the IDE_4ICDS platform is implementing the clinical guides lifecycle, which is similar to the software process lifecycle (i.e., design, modeling, deploying, executing, monitoring, etc.)

The previous description of the project and its objectives highlights the role that the proposal presented in this work can play in achieving these objectives. Our proposal has contributed to monitoring the status of the clinical guides by defining the measurement concepts which evaluate and monitor the execution of the clinical guides. We describe below how the proposed solution was applied during the clinical guide lifecycle:

In the clinical guide design phase, the stakeholders (e.g., health professionals and process engineers) define the clinical guide objectives and requirements (e.g., identifying the biomedical best practices and references, the technical requirements to execute the process, etc.). Integrating the activities of our measurement *selection and definition* phase (see Section 4.1) has allowed the stakeholder to define the measurement objectives (based on the clinical guide goals). Furthermore, it has supported the team in identifying the main measurement concepts that satisfy these objectives. Furthermore, these activities supported them in applying measurement methods to select and define these concepts in an operational manner.

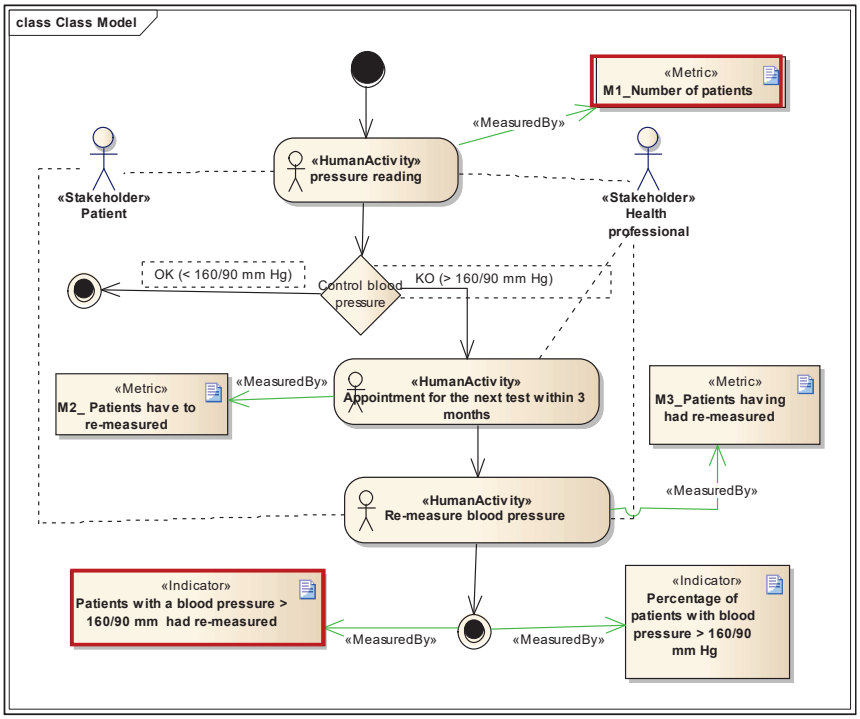

Figure 11 Part of a clinical guide model.

And in the modeling phase, stakeholders describe the different perspectives of the clinical guide in a formal language; the objective is to ensure a common understanding of the clinical guide perspectives between the various stakeholders. The MDMM and the UML profile proposed in this work allow stakeholders to describe formally the measurement objectives and concepts defined in the previous step.

Figure 11 shows part of a clinical guide modeled using the PLM$_4$BS framework, also shows the defined measures and indicators integrated into the clinical guide model. And Figures 12 and 13 demonstrate parts of the formal description of the measurement concepts defined for this clinical guide using our MMT.

6.2 The Results of Applying the Proposed Solution

The proposed solution has provided the support needed by the project team to carry out the measurement activities during the project; the proposed

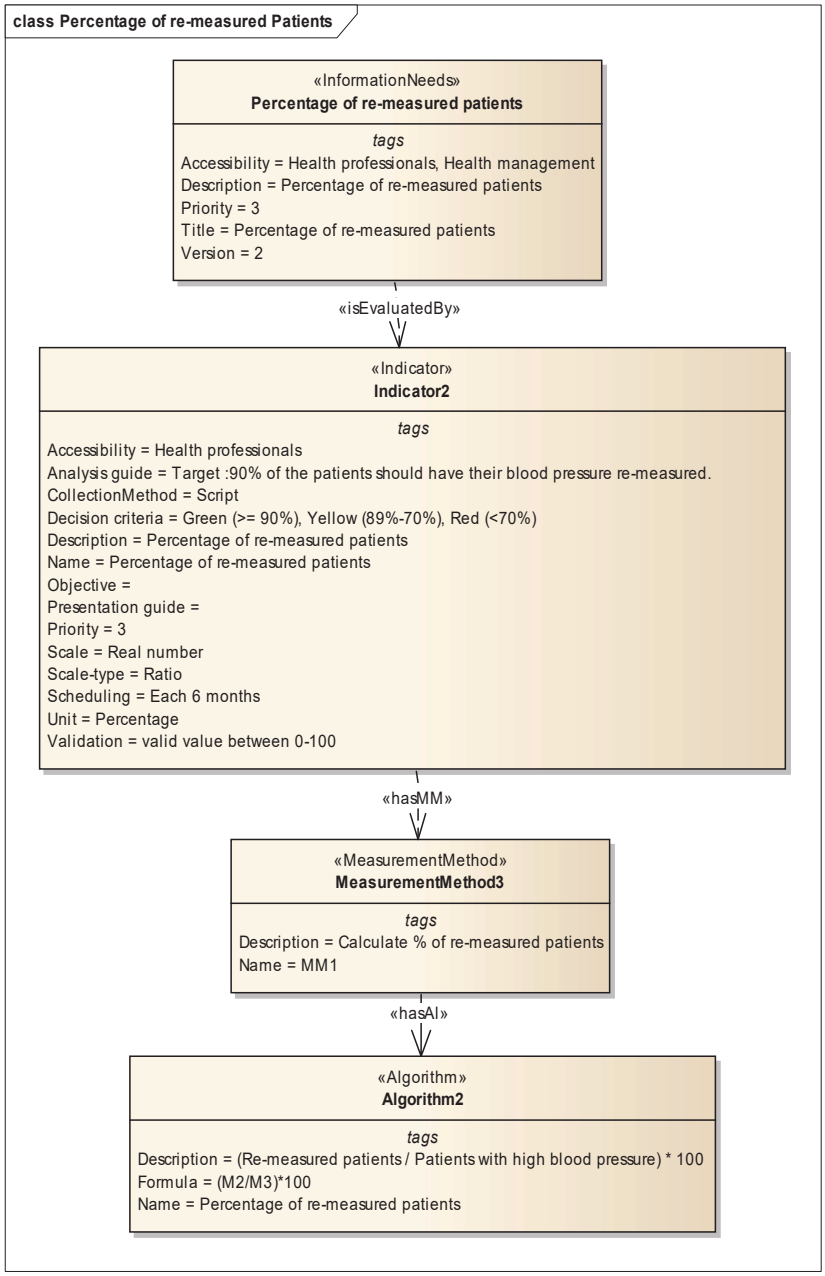

Figure 12 Part of the definition model of the measurement concepts (1).

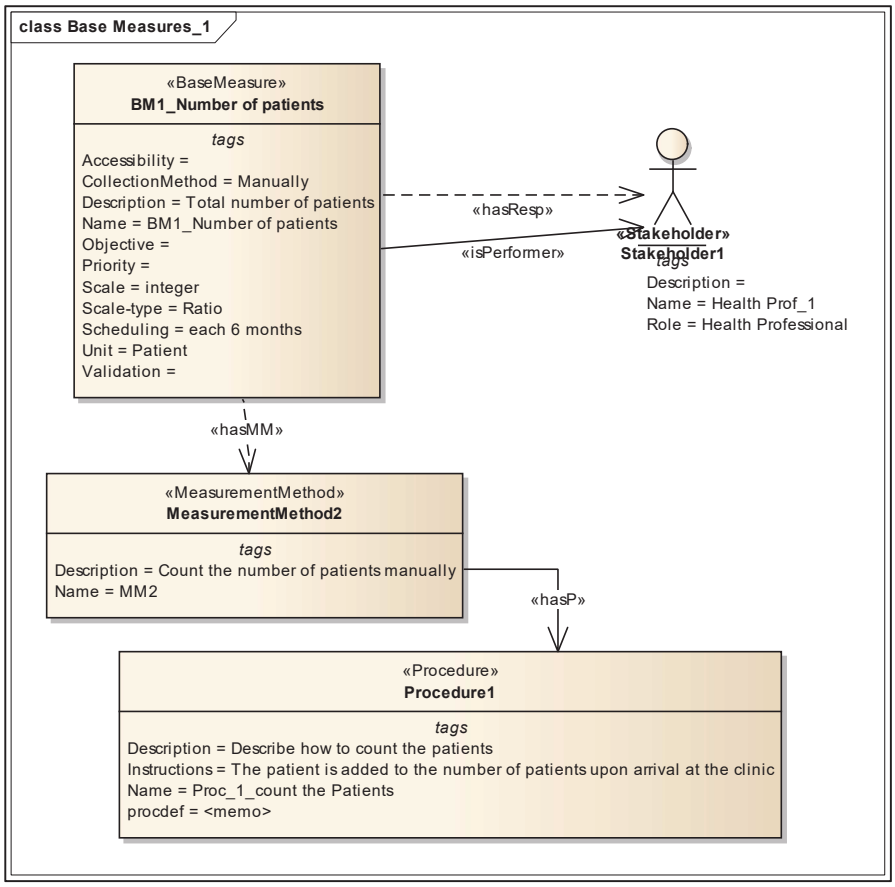

Figure 13 Part of the definition of the measurement concepts (2).

lifecycle has been used to plan and identify the required measurement activities, and the proposed measurement modeling language (MDMM) has been used to describe the measurement objectives and concepts formally and operationally. The feedback of the project team has highlighted the following benefits as the main contributions of applying the solution:

6.2.1 The proposed measurement concepts and information model

Have contributed to the unification of the measurement vocabulary used in the project and connected them coherently, also ensured the traceability

between these concepts. Using the proposed measurement concepts and information model has promoted a clear and common understanding of the measurement goals and concepts and their relationships, which has supported the communication between the project stakeholders. Moreover, the proposed measurement language has supported the operational definition of the measurement concepts.

6.2.2 The proposed measurement lifecycle

Has provided a clear and comprehensive guide to the project team; it has defined and consolidated the measurement activities which should be performed during the clinical guide lifecycle. Furthermore, this lifecycle has supported the project team in planning and performing the measurement activities by defining why? When? And how? These activities should be conducted.

6.2.3 The formal definition of the measurement concepts

Using the proposed MDMM has supported the communication between the different roles in the project and reduced the errors, time, and costs.

Besides this, since the proposed solution addresses the international standards (e.g., ISO 1593-2017) and the best practices available in the literature, it has supported the project team to follow and comply with the measurement standards.

7 Conclusions and Future Work

In this paper, we have outlined several issues related to the process measurement domain; the main problems discussed in this research are (i) the operational definition of the measurement concepts and (ii) the integration of the measurement issues (e.g., concepts, artifacts, and activities) into the process lifecycle.

To address these problems, we have presented a measurement process lifecycle; this lifecycle defines all the necessary concepts and artifacts to define and achieve the measurement objectives. The proposed lifecycle also describes the necessary activities to achieve the measurement goals in each phase of the process lifecycle (e.g., design, modeling, execution). After defining the measurement lifecycle, we have described how this lifecycle can be integrated into the process lifecycle. As we have explained, this integration describes (i) how the measurement activities will be performed throughout

the different phases of the process lifecycle and (ii) how the artifacts will be exchanged between the activities of both lifecycles.

Furthermore, we have proposed a measurement information model to define the measurement concepts and describe the relationships between the measurement goals and the necessary objective data (measures) to satisfy these goals.

Besides, we have proposed our measurement definition metamodel, which supports the modeling phase of the measurement process lifecycle. This metamodel allows engineers to operationally define the measurement objectives and the measurement concepts necessary to achieve these objectives. To allow the practical use of this metamodel, we have developed a measurement modeling tool and integrated this tool into our process modeling tool (PLM_4BS).

Moreover, we have validated our proposal in several ways; on the one hand, we have applied our proposal (the MIM, measurement concepts, and its operational definition) to many scenarios from the literature and from real cases. As an example – and to increase the readability of this paper – we have included only one scenario (in Sections 4.2.3 and 6.1). This has served as a proof-of-concept that shows the applicability of our proposal.

On the other hand, the proposal has been applied to several industry experiences. As an example, we have included the experience of applying it to a project related to the health sector. The results obtained from this experience have shown that the proposal is useful and provides important support to the project team in defining and integrating the measurement issues into the clinical guide lifecycle.

This work presents our proposal to support the measurement definition and modeling phases of the measurement process lifecycle. As future work, we plan to provide support for the rest of the measurement process lifecycle. Currently, we are developing and testing the measurement execution metamodel to support the measurement collection phase and the measurement monitoring metamodel to support the analysis and reporting phase.

The measurement execution and monitoring models will be derived automatically from the measurement definition model presented in this work. Moreover, we are developing the necessary transformations to derive the artifacts (models and documentations) which support the measurement process lifecycle.

We are also studying how the integration of the measurement model into the process model affects the maintenance and the evolution of the process model.

Acknowledgements

This research has been supported by the NICO project (PID2019-105455GB-C31) of the Spanish Ministry of Science, Economy, and University and by NDT4.0 (US-1251532) of the Andalusian Regional Ministry of Economy and Knowledge.

References

[1] N. E. Fenton, *Software metrics: a rigorous approach*. Chapman & Hall, 1991.

[2] B. Kitchenham and S. L. Pfleeger, "Software quality: The elusive target," *IEEE Softw.*, vol. 13, no. 1, p. 12, 1996.

[3] A. Fuggetta, "Software process: a roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 25–34.

[4] G. Cugola and C. Ghezzi, "Software Processes: a Retrospective and a Path to the Future," *Softw. Process Improv. Pract.*, 1998, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.2499&rep=rep1&type=pdf.

[5] F. García et al., "Towards a consistent terminology for software measurement," *Inf. Softw. Technol.*, vol. 48, no. 8, pp. 631–644, 2006.

[6] M. Bourgault, E. Lefebvre, L. A. Lefebvre, R. Pellerin, and E. Elia, "Discussion of metrics for distributed project management: Preliminary findings," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, 2002, pp. 10–pp.

[7] M. Tihinen, R. Kommeren, D. Systems, J. Rotherham, and P. M. Office, "Metrics and Measurements in Global Software Development," *Int. J. Adv. Softw.*, vol. 5, no. 3, pp. 278–292, 2012.

[8] S. K. Bang, S. Chung, Y. Choh, and M. Dupuis, "A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps," in *Proceedings of the 2nd annual conference on Research in information technology*, 2013, pp. 61–62.

[9] D. C. Schmidt, "Model-driven engineering," *Comput. Comput. Soc.*, vol. 39, no. 2, p. 25, 2006.

[10] A. Meidan, J. A. García-García, M. J. Escalona, and I. Ramos, "A survey on business processes management suites," *Comput. Stand. Interfaces*, 2016, doi: 10.1016/j.csi.2016.06.003.

[11] A. Meidan, J. A. García-García, I. Ramos, and M. J. Escalona, "Measuring Software Process: A Systematic Mapping Study," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 58:1–58:32, 2018, doi: 10.1145/3186888.

[12] R. Bendraou, M.-P. Gervais, and X. Blanc, "UML4SPM: An executable software process modeling language providing high-level abstractions," in *Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International*, 2006, pp. 297–306.

[13] B. Mora, M. Piattini, F. Ruiz, and F. Garcia, "Smml: Software measurement modeling language," in *Proceedings of the 8th Workshop on Domain-Specific Modeling (DSM'2008)*, 2008.

[14] B. Mora, F. Garcia, F. Ruiz, and M. Piattini, "Model-driven software measurement framework: A case study," in *Quality Software, 2009. QSIC'09. 9th International Conference on*, 2009, pp. 239–248.

[15] B. Mora et al., "Software generic measurement framework based on MDA," *IEEE Lat. Am. Trans.*, vol. 6, no. 4, pp. 363–370, 2008.

[16] Y. Singh and M. Sood, "Model Driven Architecture: A Perspective," *Adv. Comput. Conf. 2009. IACC 2009. IEEE Int.*, no. March, pp. 1644–1652, 2009, doi: 10.1109/IADCC.2009.4809264.

[17] X. Larrucea and E. Iturbe, "A Metamodel Integration for Metrics and Processes Correlation.," in *ICSOFT*, 2010, pp. 63–68.

[18] O. M. GROUP, "Software Metrics Metamodel," 2009. [Online]. Available: http://www.omg.org/spec/SMM/1.0/Beta1/PDF/.

[19] OMG, "SPEM 2.0 Software & Systems Process Engineering Metamodel specification," 2002. [Online]. Available: http://www.omg.org/spec/SPEM/.

[20] M. A. Freire, F. A. Aleixo, U. Kulesza, E. Aranha, and R. Coelho, "Automatic Deployment and Monitoring of Software Processes: A Model-Driven Approach.," in *SEKE*, 2011, pp. 42–47.

[21] A. Del-Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés, "On the definition and design-time analysis of process performance indicators," *Inf. Syst.*, vol. 38, no. 4, pp. 470–490, 2013.

[22] J. A. García García, M. J. Escalona, A. Martínez-García, C. Parra, and T. Wojdyñski, "Clinical Process Management: A model-driven & tool-based proposal," 2015.

[23] J. A. Garcia-Garcia, "Una propuesta para el uso del paradigma guiado por modelos (MDE) para la definición y ejecución de procesos de negocios," Sevilla, 2015.

[24] T. Allweyer, "Business Process Model and Notation (BPMN) Version 2.0," 2015, doi: 10.1007/s11576-008-0096-z.

[25] Bonitasoft, "Bonitasoft," 2016. http://www.bonitasoft.com/.
[26] Eclipse, "Eclipse Process Framework Project |projects.eclipse.org," 2017. https://projects.eclipse.org/projects/technology.epf.
[27] IBM, "IBM – Rational Method Composer," 2017. http://www-03.ibm.com/software/products/en/rmc.
[28] J. B. Hill, J. Sinur, D. Flint, and M. J. Melenovsky, "Gartner's position on business process management," *Gart. Res. G*, vol. 136533, 2006.
[29] W. M. P. van der Aalst, "Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management," Springer, Berlin, Heidelberg, 2004, pp. 1–65.
[30] W. M. P. van der Aalst, "Business process management: a personal view," *Bus. Process Manag. J.*, vol. 10, no. 2, p. bpmj.2004.15710 baa.001, Apr. 2004, doi: 10.1108/bpmj.2004.15710baa.001.
[31] N. Habra, A. Abran, M. Lopez, and A. Sellami, "A framework for the design and verification of software measurement methods," *J. Syst. Softw.*, vol. 81, no. 5, pp. 633–648, 2008.
[32] ISO/IEC/IEEE 15288-Systems and software engineering System life cycle processes, vol. 15288. 2015.
[33] ISO/IEC/IEEE 12207-2017-International Standard – Systems and software engineering – Software life cycle processes. 2017.
[34] J.-P. Jacquet and A. Abran, "From software metrics to software measurement methods: a process model," in *Proceedings of IEEE International Symposium on Software Engineering Standards*, 1997, pp. 128–135, doi: 10.1109/SESS.1997.595954.
[35] Y. Zhang and D. Sheth, "Mining software repositories for model-driven development," *IEEE Softw.*, vol. 23, no. 1, pp. 82–90, 2006, doi: 10.1109/MS.2006.23.
[36] A. Del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "Towards modelling and tracing key performance indicators in business processes," *II Taller sobre Procesos Neg. e Ing. Serv. PNIS*, 2009.
[37] "ISO/IEC/IEEE 15939-2017 International Standard – Systems and software engineering–Measurement process," pp. 1–49, 2017, doi: 10.1109/IEEESTD.2017.7907158.
[38] J. McGarry, Practical software measurement: objective information for decision makers. Addison-Wesley Professional, 2002.
[39] B. A. Kitchenham, R. T. Hughes, and S. G. Linkman, "Modeling software measurement data," *IEEE Trans. Softw. Eng.*, vol. 27, no. 9, pp. 788–804, 2001.

[40] M. P. Barcellos, R. de Almeida Falbo, and A. R. Rocha, "A strategy for preparing software organizations for statistical process control," *J. Brazilian Comput. Soc.*, vol. 19, no. 4, pp. 445–473, Nov. 2013, doi: 10.1007/s13173-013-0106-x.

[41] W. Bandara, G. G. Gable, and M. Rosemann, "Factors and measures of business process modelling: model building through a multiple case study," *Eur. J. Inf. Syst.*, vol. 14, no. 4, pp. 347–360, Dec. 2005, doi: 10.1057/palgrave.ejis.3000546.

[42] L. García-Borgoñon, J. A. García-García, M. Alba, and M. J. Escalona, "Software Process Management: A Model-Based Approach," in *Building Sustainable Information Systems*, Boston, MA: Springer US, 2013, pp. 167–178.

[43] J. A. Garcia-Garcia, A. Meidan, A. Vázquez Carreño, and M. Mejias Risoto, "A Model-Driven Proposal to Execute and Orchestrate Processes: PLM4BS," Springer, Cham, 2017, pp. 211–225.

[44] M. J. Escalona, "Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software," University of Seville, 2004.

[45] M. Escalona and G. Aragon, "NDT. A Model-Driven Approach for Web Requirements," *IEEE Trans. Softw. Eng.*, vol. 34, no. 3, pp. 377–390, May 2008, doi: 10.1109/TSE.2008.27.

[46] V. Popova and A. Sharpanskykh, "Modeling organizational performance indicators," *Inf. Syst.*, vol. 35, no. 4, pp. 505–527, 2010, doi: 10.1016/j.is.2009.12.001.

[47] N. E. Fenton and S. L. Pfleeger, *Software metrics: a rigorous and practical approach*, Second Edi. London: International Thomson Computer Press, 1996.

[48] B. Kitchenham, S. L. Pfleeger, and N. Fenton, "Towards a framework for software measurement validation," *IEEE Trans. Softw. Eng.*, vol. 21, no. 12, pp. 929–944, 1995.

[49] J. N. Martin, "Architecture Definition – A New Process in the ISO International Systems Engineering Standard," *INCOSE Int. Symp.*, vol. 25, no. 1, pp. 463–472, Oct. 2015, doi: 10.1002/j.2334-5837.2015.00075.x.

[50] C. A. Dekkers and P. A. McQuaid, "The dangers of using software metrics to (mis)manage," *IT Prof.*, vol. 4, no. 2, pp. 24–30, Mar. 2002, doi: 10.1109/MITP.2002.1000457.

[51] F. Ruiz-gonzález and G. Canfora, "Software Process: Characteristics, Technology and Environments," *CEPIS-UPGRADE*, vol. V, no. 5, pp. 5–10, 2004.

[52] T. Tahir, G. Rasool, and C. Gencel, "A systematic literature review on software measurement programs," *Inf. Softw. Technol.*, vol. 73, pp. 101–121, 2016, doi: 10.1016/j.infsof.2016.01.014.

[53] L. C. Briand, S. Morasca, and V. R. Basili, "An operational process for goal-driven definition of measures," *IEEE Trans. Softw. Eng.*, vol. 28, no. 12, pp. 1106–1125, Dec. 2002, doi: 10.1109/TSE.2002.1158285.

[54] M. Kasunic, "The State of Software Measurement Practice: Results of 2006 Survey," Pittsburgh, PA, 2006. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8095.

[55] L. Sánchez González, F. García Rubio, F. Ruiz González, and M. Piattini Velthuis, "Measurement in business processes: a systematic review," *Bus. Process Manag. J.*, vol. 16, no. 1, pp. 114–134, Feb. 2010, doi: 10.1108/14637151011017976.

[56] H. Zhang, J. Keung, B. Kitchenham, and R. Jeffery, "Semi-quantitative Modeling for Managing Software Development Processes," in *19th Australian Conference on Software Engineering (aswec 2008)*, Mar. 2008, pp. 66–75, doi: 10.1109/ASWEC.2008.4483194.

[57] T. Magennis, "The Economic Impact of Software Development Process Choice – Cycle-Time Analysis and Monte Carlo Simulation Results," in *48th Hawaii International Conference on System Sciences*, Jan. 2015, pp. 5055–5064, doi: 10.1109/HICSS.2015.599.

[58] M. Ruiz, I. Ramos, and M. Toro, "A Dynamic Integrated Framework for Software Process Improvement," *Softw. Qual. J.*, vol. 10, no. 2, pp. 181–194, 2002, doi: 10.1023/A:1020580008694.

[59] J. M. Perez-Alvarez, M. T. Gomez-Lopez, L. Parody, and R. M. Gasca, "Process Instance Query Language to Include Process Performance Indicators in DMN," in *IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, Sep. 2016, pp. 1–8, doi: 10.1109/EDOCW.2016.7584381.

[60] K. Hikichi, K. Fushida, H. Iida, and K. Matsumoto, "A Software Process Tailoring System Focusing to Quantitative Management Plans," Springer, Berlin, Heidelberg, 2006, pp. 441–446.

[61] X. Wang, A. Ren, and X. Liu, "Researching on quantitative project management plan and implementation method," 2017, p. 020176, doi: 10.1063/1.4992993.

[62] M. E. Kuwaiti and J. M. Kay, "The role of performance measurement in business process re-engineering," *Int. J. Oper. Prod. Manag.*, vol. 20, no. 12, pp. 1411–1426, Dec. 2000, doi: 10.1108/01443570010353086.

[63] R. E. Park, W. B. Goethert, and W. A. Florac, "Goal-Driven Software Measurement. A Guidebook.," 1996.

[64] D. N. Card and C. L. Jones, "Status report: practical software measurement," in *Third International Conference on Quality Software, 2003. Proceedings.*, 2003, pp. 315–320, doi: 10.1109/QSIC.2003.1319116.

[65] M. Staron, W. Meding, K. Niesel, and A. Abran, "A Key Performance Indicator Quality Model and Its Industrial Evaluation," in *Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, Oct. 2016, pp. 170–179, doi: 10.1109/IWSM-Mensura.2016.033.

[66] M. J. Ordonez and H. M. Haddad, "The State of Metrics in Software Industry," in *Fifth International Conference on Information Technology: New Generations (itng 2008)*, Apr. 2008, pp. 453–458, doi: 10.1109/ITNG.2008.106.

[67] Y. Wang *et al.*, "Product and Process Metrics: A Software Engineering Measurement Expert System," Springer, Berlin, Heidelberg, 2002, pp. 337–350.

[68] M. Staron, W. Meding, J. Hansson, C. Höglund, K. Niesel, and V. Bergmann, "Dashboards for Continuous Monitoring of Quality for Software Product under Development," in *Relating System Quality and Software Architecture*, Elsevier, 2014, pp. 209–229.

[69] M. Staron, W. Meding, G. Karlsson, and C. Nilsson, "Developing measurement systems: an industrial case study," *J. Softw. Maint. Evol. Res. Pract.*, vol. 23, no. 2, pp. 89–107, 2011, doi: 10.1002/smr.470.

[70] P. Berander and P. Jönsson, "A goal question metric based approach for efficient measurement framework definition," *Proc. 2006 ACM/IEEE Int. Symp. Int. Symp. Empir. Softw. Eng. – ISESE '06*, pp. 316–325, 2006, doi: 10.1145/1159733.1159781.

[71] K. Pandazo, A. Shollo, M. Staron, and W. Meding, "Presenting software metrics indicators: a case study," in *Proceedings of the 20th International Conference on Software Product and Process Measurement (MENSURA)*, 2010, vol. 20, no. 1.

[72] D. Montgomery, Introduction to statistical quality control. 2009.

[73] M. K. Daskalantonakis, "A practical view of software measurement and implementation experiences within Motorola," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 998–1010, 1992, doi: 10.1109/32.177369.

[74] M. Staron and W. Meding, "Using models to develop measurement systems: a method and its industrial use," *Softw. Process Prod. Meas.*, pp. 212–226, 2009, doi: 10.1007/978-3-642-05415-0_16.
[75] ISO/IEC 19507:2012.Information technology – Object Constraint Language (OCL). 2012.
[76] Sparx Systems, "Full Lifecycle Modeling for Business, Software and Systems," 2018. https://sparxsystems.com/products/ea/.
[77] Object Management Group, "Model Driven Architecture," 2015. [Online]. Available: http://www.omg.org/mda/specs.htm.
[78] M. J. Escalona, J. Torres, M. Mejías, and A. Reina, "NDT-Tool: A Case Tool to Deal with Requirements in Web Information Systems," in *Web Engineering*, 2003, pp. 212–213.
[79] IBM Corporation, "IBM Rational Software Architect Designer," 2018. https://www.ibm.com/us-en/marketplace/rational-software-architect-designer/details?mhq=RationalSoftwareModeler&mhsrc=ibmsearch_p.
[80] A.-M. Audet, S. Greenfield, and M. Field, "Medical practice guidelines: current activities and future directions," *Ann. Intern. Med.*, vol. 113, no. 9, pp. 709–714, 1990.

Biographies

Ayman Meidan is a researcher at the University of Seville, Spain. In 2019 he obtained his PhD in Computer Science by the University of Seville, Spain. His current research interests include the areas of Software Engineering, Business & Software Process Management, Model-Driven Engineering and Quality Assurance. Since 2014, he has participated as a researcher in several technological transfer and R&D projects.

J. A. García-García is professor and researcher at the University of Seville, Spain. In 2015 he obtained his PhD in Computer Science by the University of Seville, Spain. Since 2008, he has participated in R&D projects as a researcher. His current research interests include the areas of Software Engineering, Business Process Management (BPM), Model-Driven Engineering and Quality Assurance. He manages several technological transfer projects with companies and he participates as a member committee in several international conferences and journals.

Isabel Ramos Roman PhD in Industrial Engineer University of Seville (US). During many years she has lead the Foundation for Research and Development of Information Technologies in Andalusia (FIDETIA) and has been the director of the Department of Languages and Information Systems of the University of Seville and she is currently a professor in the department. He directs several research projects in the private and public sphere in the field of Information Technology. The research lines in which she is currently working on are centered in management and process improvements, in Experimentation in Software Engineering, in the application of testing techniques in

the development of software processes improvement and in business model. She is author and coauthor of a great number of national and international publications.

David Lizcano holds a Ph.D. in Computer Science from the UPM (2010), and a M.Sc. degree in Research in Complex Software Development (2008) also from UPM. He held a research grant from the European Social Fund under their Research Personnel Training program, the Extraordinary Graduation Prize for best academic record UPM and the National Accenture Prize for the Best Final-Year Computing Project. He is Professor and Senior Researcher at the Madrid Open University (UDIMA). He is currently involved in several national and European funded projects related to EUP, Web Engineering, Paradigms of Programming and HCI. He has published more than 25 papers in prestigious international journals and attended more than 70 international conferences.

María José Escalona received her PhD in Computer Science from the University of Seville, Spain in 2004. Currently, she is a Full Professor in the Department of Computer Languages and Systems at the University of

Seville. She manages the web engineering and early testing research group. Her current research interests include the areas of requirement engineering, web system development, model-driven engineering, early testing and quality assurance. She also collaborates with public companies like the Andalusian Regional Ministry of Culture and Andalusian Health Service in quality assurance issues.