An MDA proposal to integrate the measurement lifecycle Into the process lifecycle

A. Meidan[a], J. A. García-García[a], I. Ramos[a], David Lizcano[b], M.J. Escalona[a]

[a] *University of Seville*,
Avenida Reina Mercedes s/n, Seville, 41012, Spain.

[b] *School of Computer Science, Madrid Open University (UDIMA),*
Collado Villalba, 28400, Madrid, Spain.

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 to 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 the SDP lifecycle, (ii) Measurement definition metamodel (MDMM); which support 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 (Fenton, 1991). 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 (Cugola & Ghezzi, 1998; Fuggetta, 2000; B. Kitchenham & Pfleeger, 1996).

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 (Lennselius et al., 1987). Moreover, software process measurement provides support for better understanding, evaluation, and control of the development process, project, and the resulting product (Ebert et al., 2007). Measurement also enables organizations to have insight into its processes, predict, and improve its quality and performance, which give organizations a better position to make appropriate and informed decisions as early as possible during the development process (Abreu Fernando Brito & Carapuça, 1994; García et al., 2006).

In the last decades, the SDP has evolved to meet the market needs and to keep abreast of modern technologies and infrastructures that have influenced the product development and its use. These changes in the development

processes have increased the importance of the measurement (Bourgault et al., 2002) and caused changes in the measurement process and the used measures (Tihinen et al., 2012).

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) (Bang et al., 2013). 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 (Tihinen et al., 2012).

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

This work aims to use the Model-Driven Engineering (MDE) paradigm (Schmidt, 2006) to integrate the measurement process into the process lifecycle in a way that allows the definition and modeling of the process measures explicitly and operationally during the process modeling phase. It also aims to use the MDE transformations to derive the measurable process execution model from the process definition model. The result of this work is a theoretical solution guided by models to improve the measurement of the software processes, as well as a software tool to support the application of this theoretical solution in practical environments.

The remainder of this paper is organized as follows: The next 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, describe 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 the existing proposals, modeling languages, and tools related to this work. Also, 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) (A. Meidan et al., 2016), this study aims to investigate to what extent the existing BPMSs support the process lifecycle. Also, provide a guide for the organizations to plan and perform a comparative on the existing BPMSs. Which allows them to discover which BPMS best meets their process management needs.

One of the findings of this study indicates a lack of the definition and integration of the Process Performance Indicators (PPI) into the process model, also, in linking the PPIs with the service level agreements.

This finding prompted the authors to perform the second study, a Systematic Mapping Study (Ayman Meidan et al., 2018) that focuses on the measurement of the software development process and its execution projects, mainly to give insight on the measurement related to the “Project” and “Process” entities.

These previous two studies reveal the lack of support for the measurement in the existing process modeling and management tools and proposals. The survey demonstrates the existing weakness in the definition of the measurements and its integration into the process lifecycle; the majority of the investigated proposals do not support the definition and integration of the process measurement, this integration promotes the process monitoring and improvement.

Furthermore, the mapping study demonstrates the scarcity of research on defining the measurements 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 (Deming, 1986), as well as considering the measurement issue 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 reveal 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, for that it has gained significant interest from both researchers and practitioners. There are many proposals in the literature related to the measurement definition, modeling, and execution. This section focuses on the model-based proposals.

In (Bendraou et al., 2006) 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 (Mora et al., 2009; Mora, Garcia, et al., 2008; Mora, Piattini, et al., 2008) the authors propose a measurement framework based on Model Driven Architecture (MDA) (Singh & Sood, 2009) 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 which allows the users to define software measurement models based on software measurement ontology. This work focuses mainly on measuring model elements based on its 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 (Larrucea & Iturbe, 2010) authors present an approach to combine different metamodels (e.g., SMM (GROUP, 2009) and SPEM 2.0 (OMG, 2002)) 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 but does not allow the process modeler to model the measures in an explicit and operational form within the process model.

In (Freire et al., 2011) the authors present a model-driven approach for the definition, execution, and monitoring of SDPs, it supports the automatic collection of quantitative measures during project execution. The authors define 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 also 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 (Del-Río-Ortega et al., 2013) provides 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 definition of context data to be collected with the measurement value.

In the proposal (Garcia-Garcia, 2015; García García et al., 2015) 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, but 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.

On the other side, the industrial standard BPMN 2.0 (Allweyer, 2015) does not define the measures as a process element. The commercial implementations of this standard (e.g., Bonita BPM (Bonitasoft, 2016)) allows the modeler

to define an attribute to be measured but does not define the measures as an element to allow the modeler to include it in an explicit and operational way. SPEM 2.0 (OMG, 2002) defines the measure as a process element but in basic and abstract form, wherefore, the process modeling tools which use SPEM 2.0 metamodel (e.g., EPF(Eclipse, 2017) and RMC (IBM, 2017)) does 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 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

Lifecycle can be defined as a series of activities grouped in a set of phases- each with a specific focus - performed to achieve 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 (Hill et al., 2006) 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 (W.M.P. van der Aalst, 2004; Wil M. P. van der Aalst, 2004) 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)(Habra et al., 2008). 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. (Jacquet & Abran, 1997) 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 many several forms (e.g., characterizing and predicting purposes).

In a similar form, authors in (Y. Zhang & Sheth, 2006) 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 (Del-Río-Ortega et al., 2009) 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*eport*. In this phase, the analysis results are summarized and reported to the users.

Furthermore, the recent version of the standard ISO 15939-2017 (*ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process*, 2017) 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, such as: 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 (McGarry, 2002), and ISO 15939) to guide the selection and definition of the measurement concepts 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 (B. A. Kitchenham et al., 2001). 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 the collection, storing, and analysis of 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 the integration of the measurement into the software process (Barcellos et al., 2013).

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 (Del-Río-Ortega et al., 2009; Habra et al., 2008; Jacquet & Abran, 1997). Defining the measurement concepts in such form supports the integration of the measurement into the SDP. The research 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 its integration into the process lifecycle? (iii) How to consolidate the existing measurement selection and definition methods to support the operational definition of measurement concepts.

The second problem is related to the integration of measurement issues (e.g., concepts, artifacts, and activities) into the process lifecycle. To address this problem, the research focus 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) (Bandara et al., 2005) 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 its lifecycle).

The second objective is **defining** the measurement lifecycle, and **integrating** it into the process lifecycle, for this purpose we propose three metamodels. The first 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. The second is the ***Measurement Execution Metamodel*** which supports the measurement during the process execution phase (e.g., collecting and validating the measures data). The third metamodel is the **monitoring metamodel** which supports the monitoring and reporting of the measurement data. And finally, ***merging*** the measurement metamodels with the process metamodels to complete the integration.

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 readability of the paper we have excluded the measurement execution metamodel, monitoring metamodel, and the MDE transformation rules.

3.2 Influences

This section outlines the previous works and the main technological and conceptual aspects that influenced the development of this work.

3.2.1 The conclusions obtained from the previous research.

The results obtained from the survey and the mapping studies (presented in section 2.1) have demonstrated the lack of supporting the process measurement in the existing process management tools. This issue could be divided into two aspects:

On the one hand, the conceptual aspect, this aspect is related to the definition of the measurement concepts in the form that supports the measurement objectives and the alignment with the business needs. Existing process definition languages (e.g., SPEM and BPMN) only allow defining the measures in a simple and generic form (e.g., measure name, description, and value); this definition lack of defining important measurement aspects such as the information needs to be satisfied by this measure, the validation rules, and the context data. For that, it is essential to enrich the measurement definition to comprise all the necessary data to support the measurement configuration, collection, validation, analysis.

On the other hand, the integration aspect, this aspect is related to integrating the measurement into the process lifecycle in the form that uses the measurement definition to (i) Configure the measurements in the process deployment phase, (ii) Collect the process execution data, and (iii) Analyzing and reporting the measurements data in the process monitoring and improvement phases. Integrating the measurement in such a form allows achieving the measurement goals and satisfying the organization's needs.

3.2.2 Model-Driven Engineering

Using model-based approaches is a growing trend when developing software processes modeling languages (García-Borgoñón et al., 2014). Due to the high-level abstraction and code reuse (or regeneration) that provide, it seems appropriate to consider this approach when developing a solution. Adopting the model-driven approach reduces the development time, enhance the quality of the final code, and also facilitate and improve the process of applying changes or maintenance (Czarnecki & Helsen, 2003; Kleppe et al., 2003).

Model-Driven Engineering paradigm has emerged to address the complexity of the software systems to express the concepts of the problem's domain adequately. In this line, the basic principle of MDE is *«Everything is a model»* (Bézivin, 2005). MDE seems appropriate to achieve the objectives of this work because it uses models to represent the information of a given domain. In the context of this work, using the models allows the formalization of the measurement information. Also because this paradigm applies transformations which are a possible tool to describe,

perform, and automate the transformations between the models. Furthermore, and as mentioned earlier, using model-based approaches is the current trend in the development of the software processes modeling languages.

3.2.3 Product Lifecycle Management for Business-Software (PLM4BS) framework

The PLM4BS is a model-driven based framework (García-Borgoñon et al., 2013; Garcia-Garcia et al., 2017) 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.

PLM4BS 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*.

PLM4BS framework has been developed in the same research group in which this work was developed; the Web Engineering and Testing Early (IWT2) research group has started the development of PLM4BS to support the evolution of the NDT (Navigational Development Techniques) (Escalona & Aragon, 2008; Escalona, 2004) methodology which was developed and used by the group to support the software development lifecycle. For this reason, this work is influenced and motivated by this framework.

As mentioned early, PLM4BS defines the process lifecycle in four phases. Currently, the framework provides support for the first two phases (modeling and execution & orchestration) only. 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 PLM4BS (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) in the way that supports 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 Process continuous improvement.

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 4.1**,** the proposed lifecycle defines all the activities related to the measurement process:

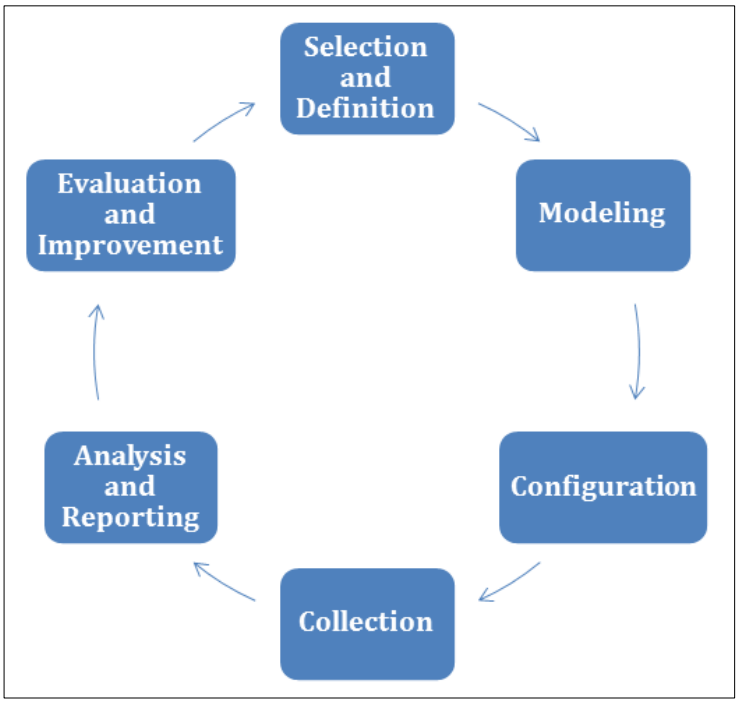

Figure 4.1. Measurement process lifecycle.

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

Modeling: In this phase, the measurement definition resulted from the previous phase is represented in a formal and operational form, and also 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) (Del-Río-Ortega et al., 2013; Popova & Sharpanskykh, 2010) in this phase.

Configuration: 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 also reporting the measurement data.

Collection: 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: In this phase, measurement data is analyzed and reported according to the measurement definition; the resulting information is generated, formed, and communicated to the pre-defined roles as indicated in the measurement definition.

Evaluation and improvement: 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., (Fenton & Pfleeger, 1996; Habra et al., 2008; B. Kitchenham et al., 1995)). Also, the industry standard ISO/IEC/IEEE 15288:2015 (Martin, 2015) 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* that focuses on evaluating the measurement principle in itself; to ensure that the process is characterizing what it intended to measure (Habra et al., 2008).

4.1.2 Integrating the process and measurement lifecycles

The measurement process is closely related to the SDP since the measurement process provides support for the software process in various phases throughout its lifecycle such as design and simulation, monitoring and improvement(Mora et al., 2009). 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. (Daskalantonakis, 1992; Dekkers & McQuaid, 2002).

The potential benefits of this integration include: (**i**) The integration of the measurement process into the development process establishes a connection (Figure 4.2) between the two parts of the development process (that is, the management process and the production process (McLeod, Raymond and Schell, 1990; Ruiz-gonzález & Canfora, 2004)). 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 life cycle. (**iv**) Promotes objective communication between the stakeholders by using common concepts and terminology. (**v**) Defining how 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. ("Tell me how you will evaluate me to tell you how I will behave. (Eliyahu & Goldratt, 1990)").

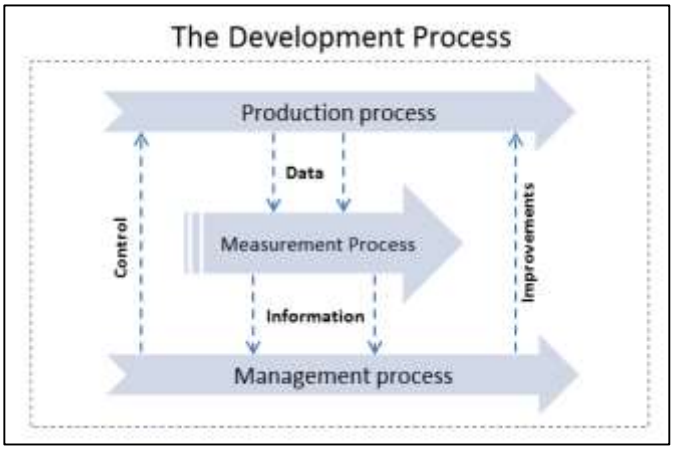

Figure 4.2. The measurement process connects the two parts of the SDP.

This integration could be done by introducing the measurement activities into its corresponding phases in the software process lifecycle and allowing the transition of the artifacts between the two lifecycles. Figure 4.3 shows 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 phase, process engineers define the process's main objectives, activities, roles, and outputs. The measurement team - in collaboration with the process engineers- can use these details to (**i**) Define 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 objectives. (**iii**) Demonstrate the management's commitment to the measurement processes which is an essential success factor for the measurement process (Tahir et al., 2016).

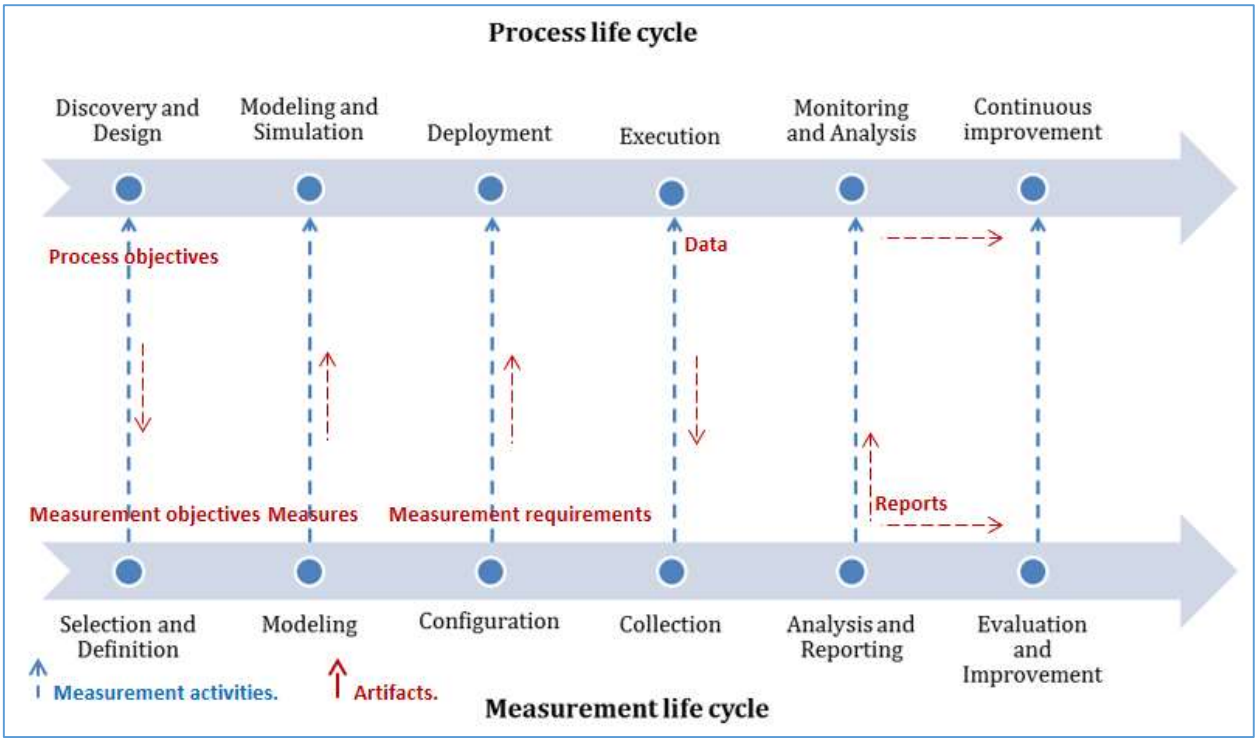

Figure 4.3. Integrate the lifecycle of measurement into the lifecycle of the process.

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 be used to 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 (Briand et al., 2002; Kasunic, 2006). The integration of measurement concepts in the process requires the clarification of the following details (Barcellos et al., 2013): (**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 and indicators, the measure, and stakeholder) in the form that facilitates its traceability and prioritization.

Moreover, in this phase, the defined measures could be used to support the simulation of the process execution. This simulation evaluates several aspects of the process for different purposes such as possible improvements, changes (Magennis, 2015; Sánchez González et al., 2010; H. Zhang et al., 2008), and assessment (Ruiz et al., 2002). 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 reports 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 (Sánchez González et al., 2010), the predefined analysis models and decision criteria are applied to support the management team to analyze the process performance(Perez-Alvarez et al., 2016), the quantitative management (Hikichi et al., 2006; X. Wang et al., 2017), 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 (Kuwaiti & Kay, 2000; Nissen, 1998).

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, it is essential to select and define these concepts in a complete and operational form as described in (Park et al., 1996).

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 (*ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process*, 2017).

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 (Card & Jones, 2003). The MIM shown in figure 4.4 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 (Staron et al., 2016), first one (base) is used to obtain the data, second level (derived)is used to prepare the data to 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 (Ordonez & Haddad, 2008), 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 (Y. Wang et al., 2002)(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 the management in the analysis and decision making (*ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process*, 2017). It applies the evaluation/estimation models (calculations or algorithm) to the measures, then, display and communicate the

results to the stakeholders. The *decision criteria* (e.g., patterns, thresholds, or target values (Staron et al., 2014)) provide support for interpreting the indicator value and also to suggest actions based on the indicator results.

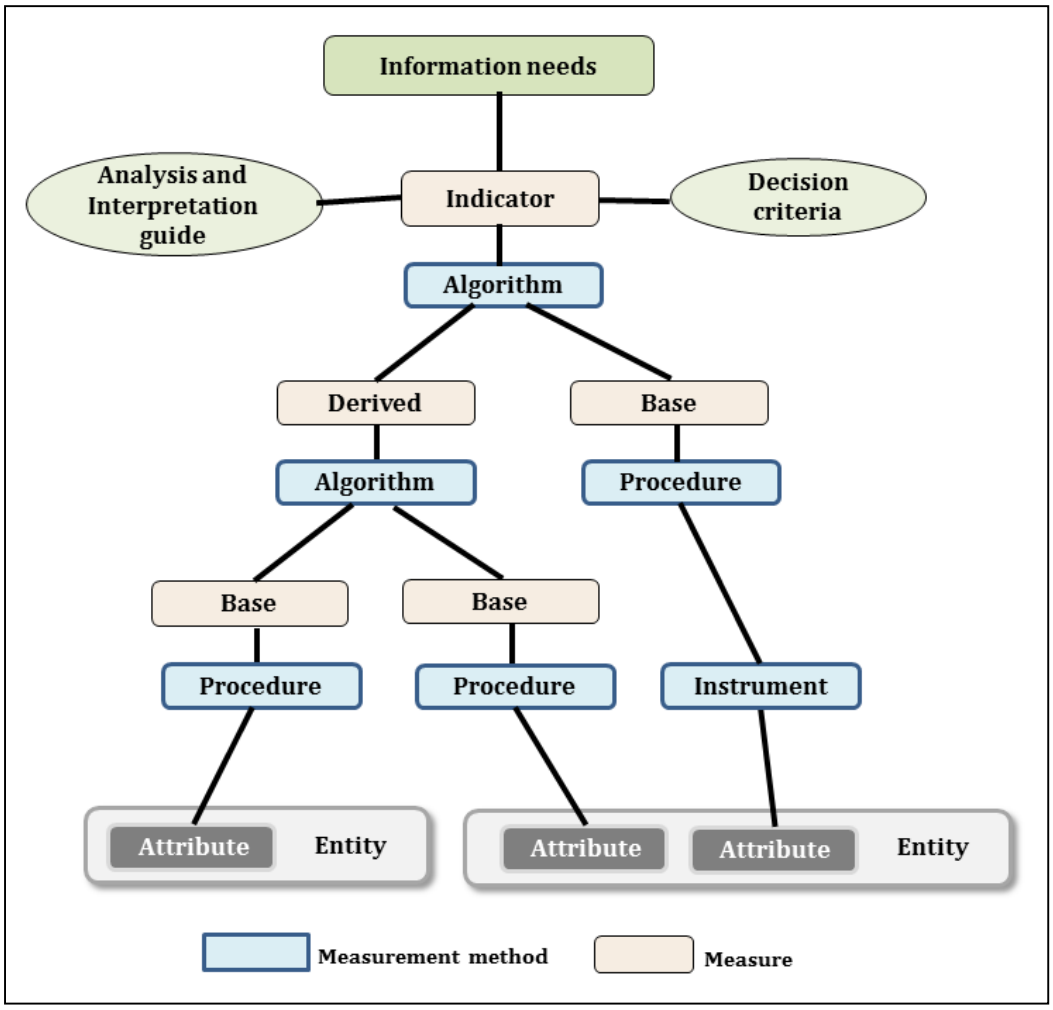

Figure 4.4. Measurement Information model.

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 (Staron et al., 2011). *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 or characteristic 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 (Berander & Jönsson, 2006), **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. This could include *Thresholds* (upper limit, center limit, low limit), and *color scale* with the traffic light metaphor(Pandazo et al., 2010; Staron et al., 2014). **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 (Dekkers & McQuaid, 2002). **Priority:** Define the priority of the indicator (Berander & Jönsson, 2006). **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 (Montgomery, 2009) 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 + derived_m3)* base_m7.

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 (Habra et al., 2008). The most common scale types (*ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process*, 2017) 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 (B. Kitchenham et al., 1995). *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 (Daskalantonakis, 1992; *ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process*, 2017). 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

The following scenario –based on (Staron & Meding, 2009)- illustrates how to use the proposed concepts in practice. Project management needs to know the cost situation of the project (e.g., the ratio between allocated and used budgets).

In this case, the *information needs* is to understand the cost situation of the project; this need is fulfilled by *the indicator* "cost situation" which informs the management about the project cost. This indicator defines the following analysis model and decision criteria to satisfy the management requirements.

The analysis guide or model defines three levels for the cost situation indicator. The indicator could have a "Red-unacceptable" level defined when the cost of the project exceeds the budget and a "Green-acceptable" level when the cost is up to 90% of the budget, leaving the 10% remaining to be the "Yellow-warning" level of the indicator.

The decision criteria associated with the indicator define the actions that must be taken when a specific criterion occurs: "Red-unacceptable". Call for meeting with project management, "Green-acceptable". Inform management, or "Yellow-warning". Inform management and call for meeting with the project team.

This indicator uses a *derived measure* to evaluate the cost situation by applying the *Algorithm* (e.g., calculation) *measurement method* 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 defined *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. Also defines the necessary (**ii**) MDE transformation rules to derive and automatically generate the necessary artifacts throughout the measurement process lifecycle.

Figure 4.5 shows the proposed metamodels, the relationships between them, and also 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 allows the integration of 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 transformations 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.

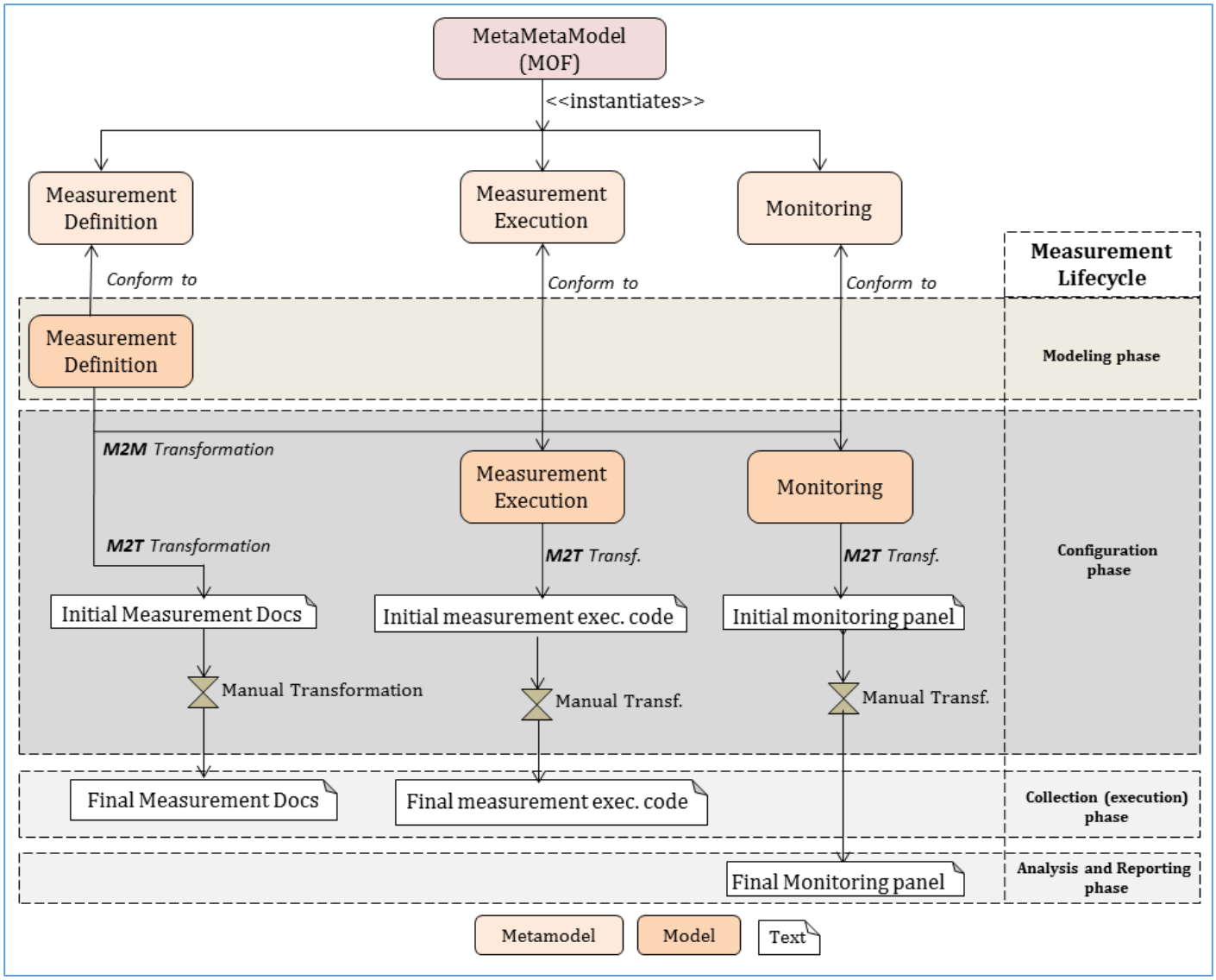

Figure 4.5 The MDE solution (metamodels and transformation rules).

Figure 4.5 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 also, preserves the relationship between these goals and its related measurement data. **Measurement documentation**. These documents provide the specifications of each measurement concept defined in the MDM. These documents will be derived from the MDM using (M2T) transformations.

As mentioned before, this paper only covers the measurement definition metamodel, the rest of the metamodels and the transformation rules will be introduced in future papers to improve the readability of this paper. The next section introduces the proposed Measurement definition metamodel.

The measurement definition metamodel (MDMM) aims to support the measurement modeling phase. Its 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 4.6 the proposed measurement language is described in form of 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 ISO/IEC 19507 (*ISO/IEC 19507:2012.Information technology - Object Constraint Language (OCL)*, 2012). 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. 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 value of the measure, this measurement method should define at least one procedure. The OCL expression which implements this restriction is:

```
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. The following OCL expression implements this restriction:

```
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* define the necessary details to support and guide the analysis and interpretation of the indicator results, where the attribute *Decision criteria* specify actions to be taken based on specific indicator results, and the attribute *Presentation guide* provide 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 using the following OCL expression:

```
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 *MeasurementMethod* 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, where 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) –in the measurement modeling phase- to describe the measurement concepts, the output of this phase is a measurement definition model, this model contain 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 4.7, 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, next section describes the developed tool which allows the practical use of this proposal.

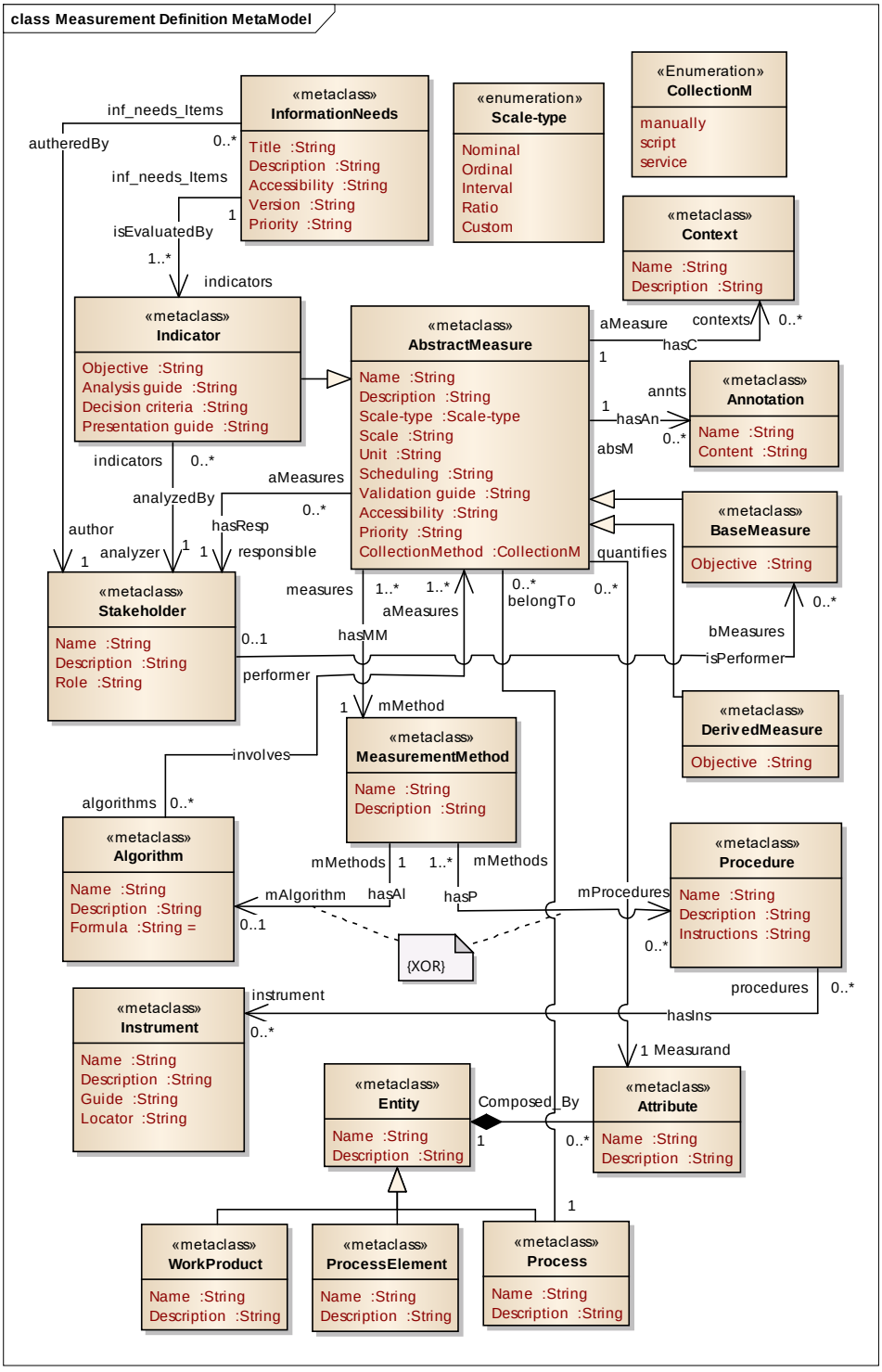

Figure 4.6. The measurement definition metamodel (MDMM).

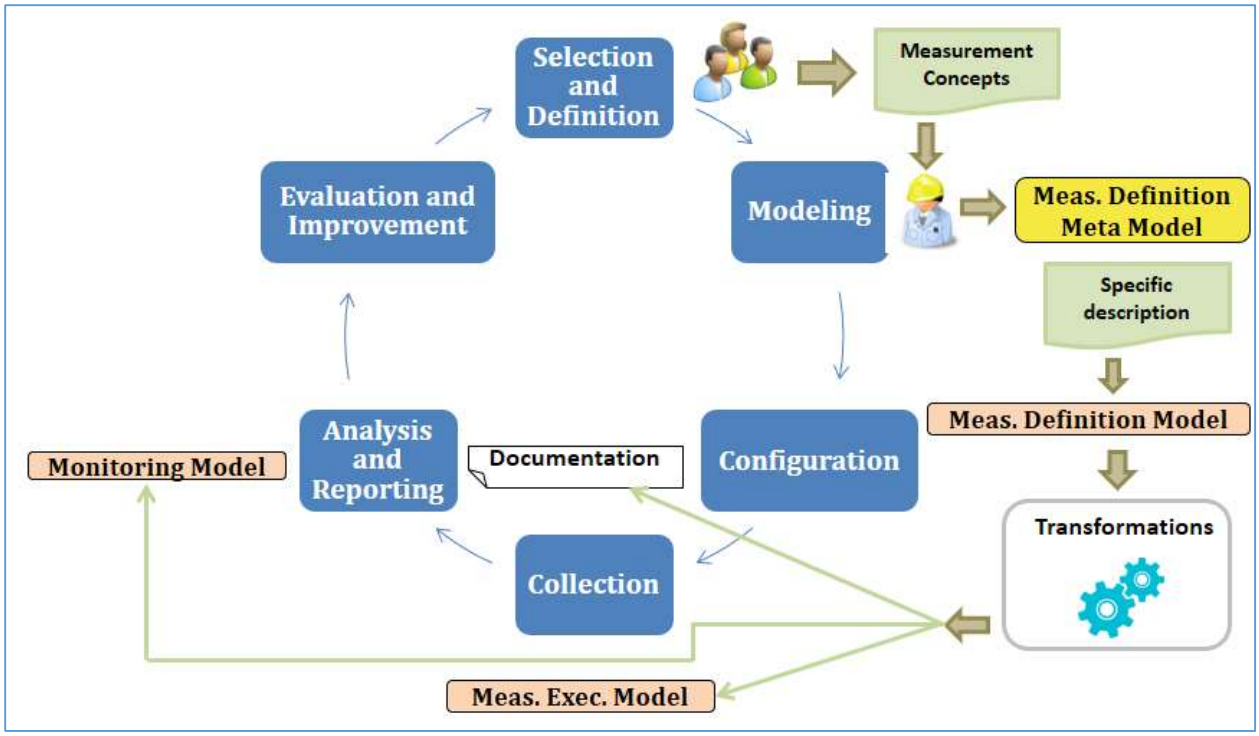

Figure 4.7. The role of the measurement definition metamodel in the proposed solution.

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$_4$BS process modeling tool (see section 3). PLM$_4$BS is based on Enterprise Architect (EA) (Sparx Systems, 2018)) as a modeling CASE tool. Therefore, the development of our MMT consists of integrating the proposed measurement definition language into the PLM$_4$BS 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$_4$BS 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 (Object Management Group, 2015). This profile allows the instantiation of the MDMM using a visual notation that can be used by CASE tools (e.g., NDT-Tool (Escalona et al., 2003), IBM Rational Software Architect Designer (IBM Corporation, 2018), 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$_4$BS 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$_4$BS). 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 4.9 and 4.10 show the measurement toolbox defined for our MMT.

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$_4$BS). This section describes the application and evaluation of our MMT in a real project.

6.1 The Application of the proposal

We have applied and evaluated our proposal in a real project related to the health industry; the IDE$_4$ICDS project (Integrated 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 (Audet et al., 1990), together with 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 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$_4$ICDS 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, 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.

And in the modeling phase, stakeholders describe the different perspectives of the clinical guide in a formal language; the objective is ensuring 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 6.1 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 (6.2 and figure 6.3) demonstrate parts of the formal description of the measurement concepts defined for this clinical guide using our MMT.

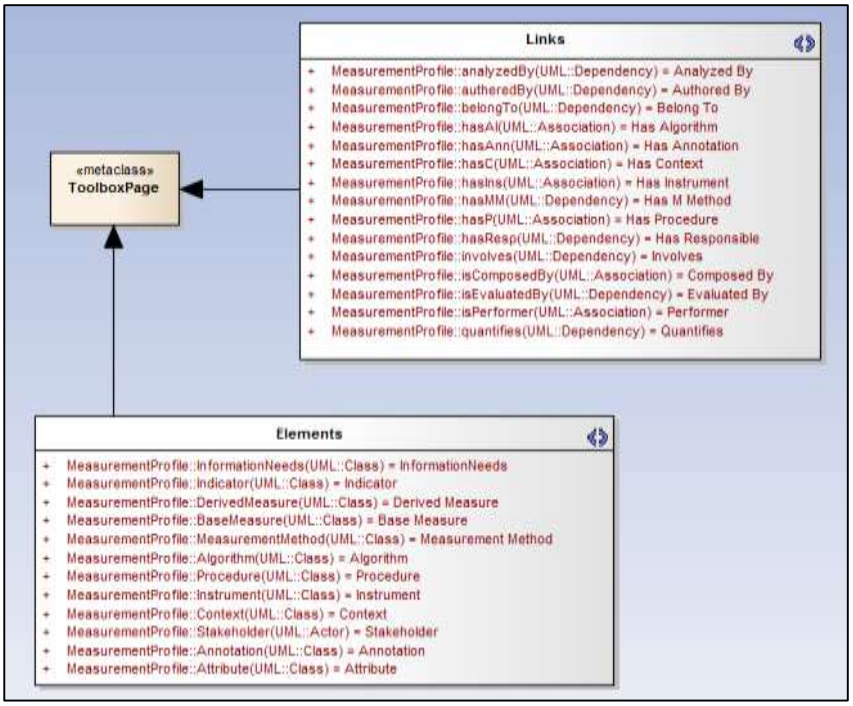

Figure 4.9. The elements and relations of the MMT toolbox.

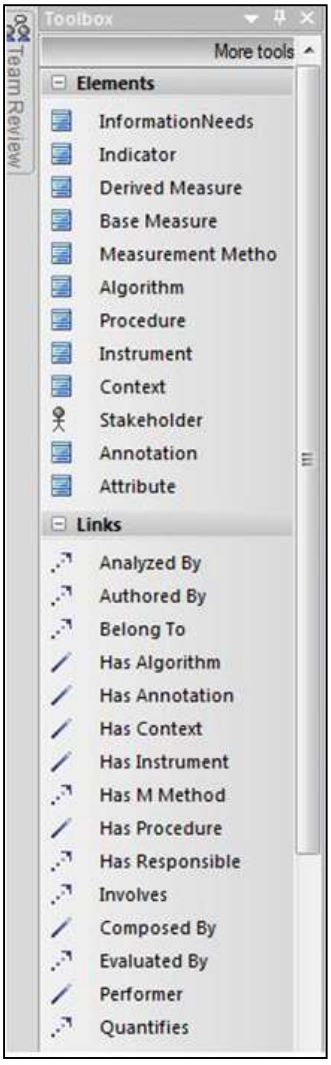

Figure 4.10.
Measurement toolbox.

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 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.

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 its relationships, which has supported the communication between the project stakeholders. Moreover, the proposed measurement language has supported the operational definition of the measurement concepts.

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.

As well as, 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.

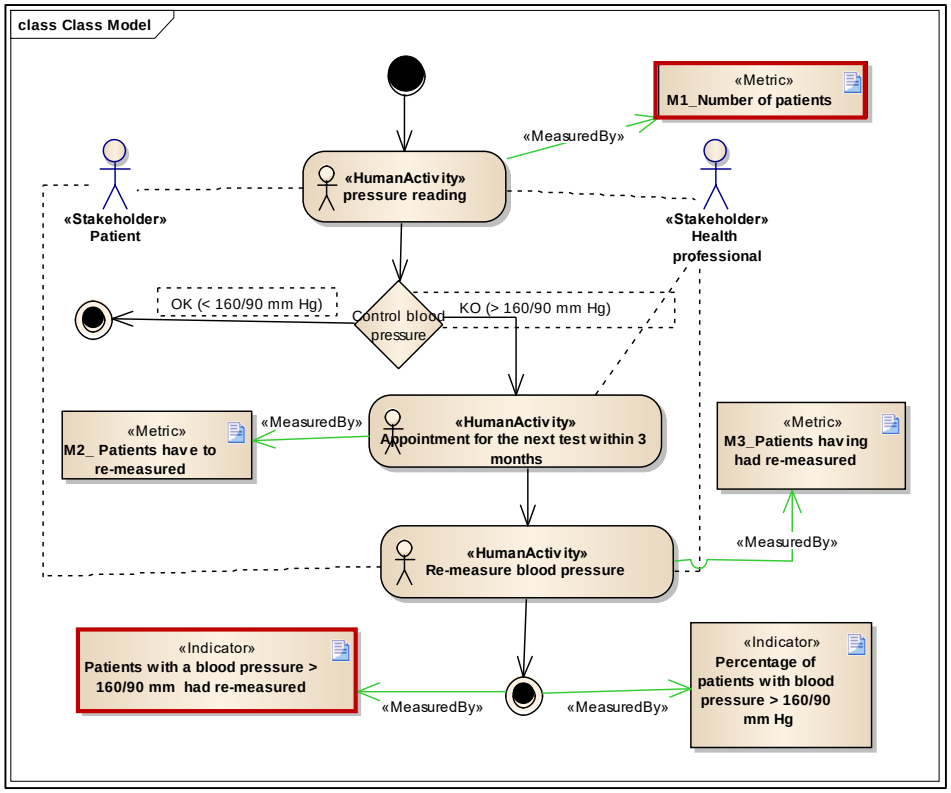

Figure 6.1. Part of a clinical guide model.

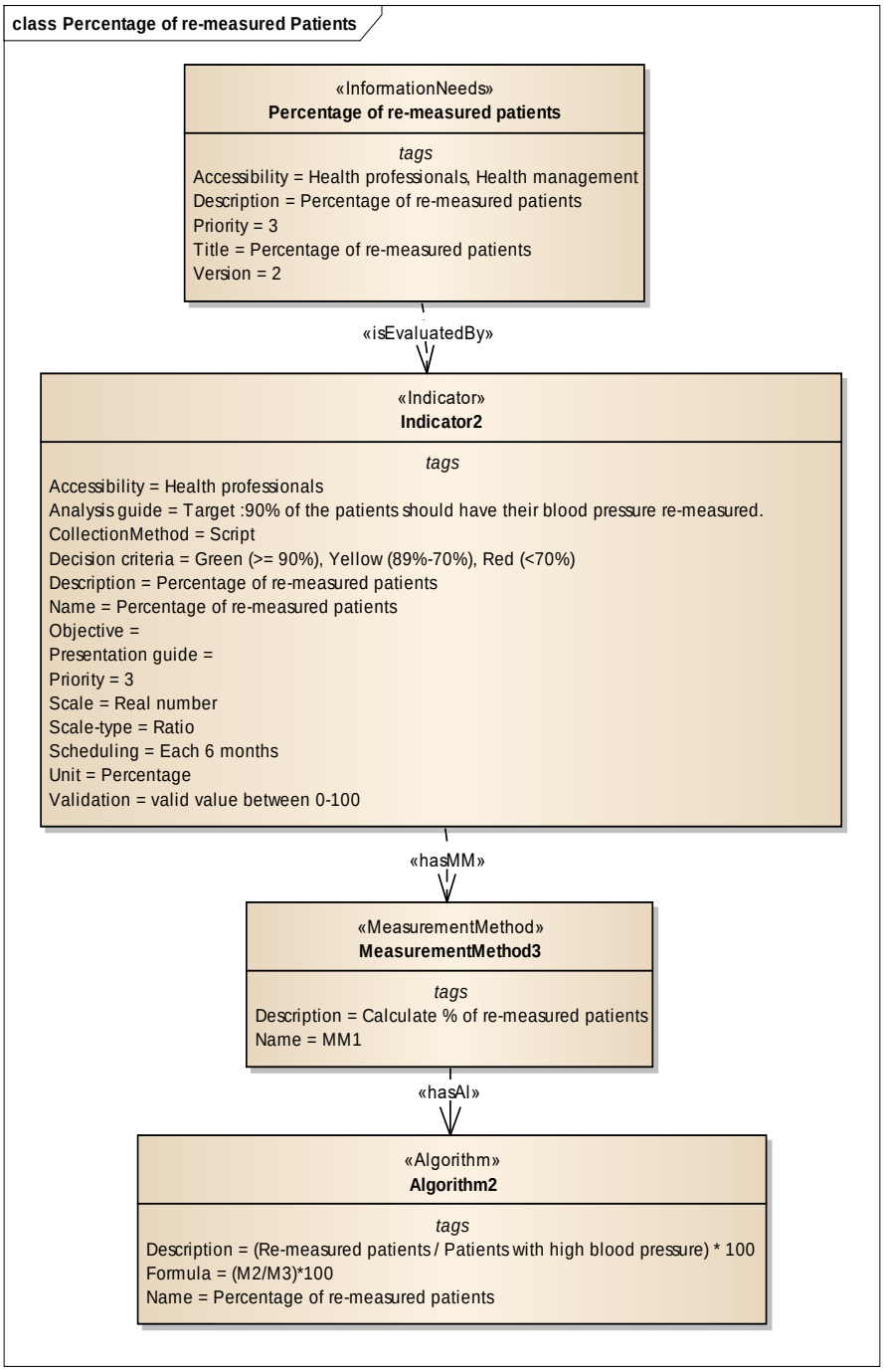

Figure 6.2. Part of the definition model of the measurement concepts (1)

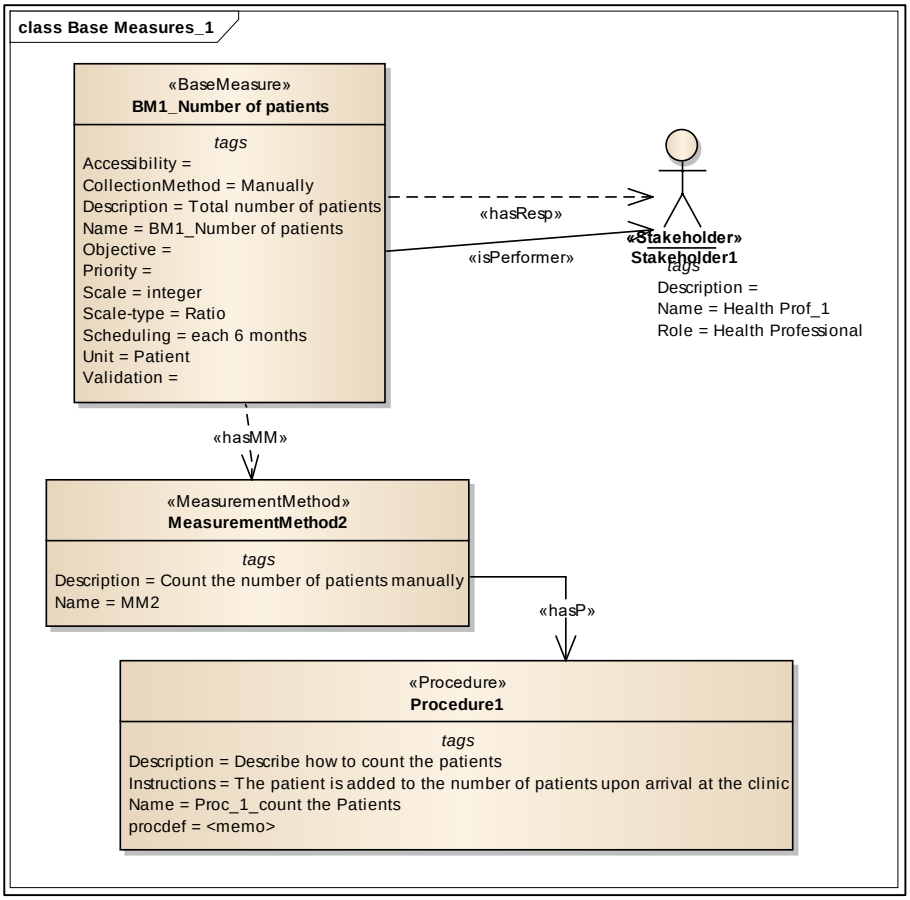

Figure 6.3. Part of the definition model of the measurement concepts (2).

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 proposed 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 also to 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 the engineers to define operationally the established measurement goals and the necessary measurement concepts to achieve these goals. 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$_4$BS).

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 (see section 4.2.3), 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 has shown that the proposal is useful and provide an important support to the project team in defining and integrating the measurement issues into the clinical guide lifecycle.

In this work, we have presented 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.

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

Abreu Fernando Brito, & Carapuça, R. (1994). Object-oriented software engineering: Measuring and controlling the development process. *Proceedings of the 4th International Conference on Software Quality*, *186*, 1–8.

Allweyer, T. (2015). *Business Process Model and Notation (BPMN) Version 2.0*. https://doi.org/10.1007/s11576-008-0096-z

Audet, A.-M., Greenfield, S., & Field, M. (1990). Medical practice guidelines: current activities and future directions. *Annals of Internal Medicine*, *113*(9), 709–714.

Bandara, W., Gable, G. G., & Rosemann, M. (2005). Factors and measures of business process modelling: model building through a multiple case study. *European Journal of Information Systems*, *14*(4), 347–360. https://doi.org/10.1057/palgrave.ejis.3000546

Bang, S. K., Chung, S., Choh, Y., & Dupuis, M. (2013). A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps. *Proceedings of the 2nd Annual Conference on Research in Information Technology*, 61–62.

Barcellos, M. P., de Almeida Falbo, R., & Rocha, A. R. (2013). A strategy for preparing software organizations for statistical process control. *Journal of the Brazilian Computer Society*, *19*(4), 445–473. https://doi.org/10.1007/s13173-013-0106-x

Bendraou, R., Gervais, M.-P., & Blanc, X. (2006). UML4SPM: An executable software process modeling language providing high-level abstractions. *Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International*, 297–306.

Berander, P., & Jönsson, P. (2006). A goal question metric based approach for efficient measurement framework definition. *Proceedings of the 2006 ACM/IEEE International Symposium on International Symposium on Empirical Software Engineering - ISESE '06*, 316–325. https://doi.org/10.1145/1159733.1159781

Bézivin, J. (2005). On the unification power of models. *Software & Systems Modeling*, *4*(2), 171–188. https://doi.org/10.1007/s10270-005-0079-0

Bonitasoft. (2016). *Bonitasoft*. http://www.bonitasoft.com/

Bourgault, M., Lefebvre, E., Lefebvre, L. A., Pellerin, R., & Elia, E. (2002). Discussion of metrics for distributed project management: Preliminary findings. *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference On*, 10--pp.

Briand, L. C., Morasca, S., & Basili, V. R. (2002). An operational process for goal-driven definition of measures. *IEEE Transactions on Software Engineering*, *28*(12), 1106–1125. https://doi.org/10.1109/TSE.2002.1158285

Card, D. N., & Jones, C. L. (2003). Status report: practical software measurement. *Third International Conference on Quality Software, 2003. Proceedings.*, 315–320. https://doi.org/10.1109/QSIC.2003.1319116

Cugola, G., & Ghezzi, C. (1998). Software Processes: a Retrospective and a Path to the Future. *Software Process: Improvement and Practice*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.2499&rep=rep1&type=pdf

Czarnecki, K., & Helsen, S. (2003). Classification of Model Transformation Approaches. *2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, 1–17. https://doi.org/10.1147/sj.453.0621

Daskalantonakis, M. K. (1992). A practical view of software measurement and implementation experiences within Motorola. *IEEE Transactions on Software Engineering*, *18*(11), 998–1010. https://doi.org/10.1109/32.177369

Dekkers, C. A., & McQuaid, P. A. (2002). The dangers of using software metrics to (mis)manage. *IT Professional*, *4*(2), 24–30. https://doi.org/10.1109/MITP.2002.1000457

Del-Río-Ortega, A., Resinas, M., Cabanillas, C., & Ruiz-Cortés, A. (2013). On the definition and design-time analysis of process performance indicators. *Information Systems*, *38*(4), 470–490.

Del-Río-Ortega, A., Resinas, M., & Ruiz-Cortés, A. (2009). Towards modelling and tracing key performance indicators in business processes. *II Taller Sobre Procesos de Negocio e Ingeniería de Servicios, PNIS*.

Deming, W. E. (1986). Out of the crisis, Massachusetts Institute of Technology. *Center for Advanced Engineering Study, Cambridge, MA*, *510*.

Ebert, C., Dumke, R., Bundschuh, M., & Schmietendorf, A. (2007). Best practices in software measurement: How to use metrics to improve project and process performance. In *Best Practices in Software Measurement: How to use Metrics to Improve Project and Process Performance*. https://doi.org/10.1007/b138013

Eclipse. (2017). *Eclipse Process Framework Project | projects.eclipse.org*. https://projects.eclipse.org/projects/technology.epf

Eliyahu, & Goldratt, M. (1990). *The haystack syndrome: sifting information out of the data ocean*. North River Press.

Escalona, M. ., & Aragon, G. (2008). NDT. A Model-Driven Approach for Web Requirements. *IEEE Transactions on Software Engineering*, *34*(3), 377–390. https://doi.org/10.1109/TSE.2008.27

Escalona, M. J. (2004). *Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software*. University of Seville.

Escalona, M. J., Torres, J., Mejías, M., & Reina, A. (2003). NDT-Tool: A Case Tool to Deal with Requirements in Web Information Systems. In J. M. C. Lovelle, B. M. G. Rodríguez, J. E. L. Gayo, M. del Puerto Paule Ruiz, & L. J. Aguilar (Eds.), *Web Engineering* (pp. 212–213). Springer Berlin Heidelberg.

Fenton, N. E. (1991). *Software metrics : a rigorous approach*. Chapman & Hall.

Fenton, N. E., & Pfleeger, S. L. (1996). *Software metrics: a rigorous and practical approach* (Second Edi). International Thomson Computer Press.

Freire, M. A., Aleixo, F. A., Kulesza, U., Aranha, E., & Coelho, R. (2011). Automatic Deployment and Monitoring of Software Processes: A Model-Driven Approach. *SEKE*, 42–47.

Fuggetta, A. (2000). Software process: a roadmap. *Proceedings of the Conference on the Future of Software Engineering*, 25–34.

García-Borgoñón, L., Barcelona, M. A., García-García, J. A., Alba, M., & Escalona, M. J. (2014). Software process

modeling languages: A systematic literature review. *Information and Software Technology*, *56*(2), 103–116. https://doi.org/10.1016/j.infsof.2013.10.001

García-Borgoñon, L., García-García, J. A., Alba, M., & Escalona, M. J. (2013). Software Process Management: A Model-Based Approach. In *Building Sustainable Information Systems* (pp. 167–178). Springer US. https://doi.org/10.1007/978-1-4614-7540-8_13

Garcia-Garcia, J. A. (2015). *Una propuesta para el uso del paradigma guiado por modelos (MDE) para la definición y ejecución de procesos de negocios* [Sevilla]. https://idus.us.es/xmlui/handle/11441/26740

Garcia-Garcia, J. A., Meidan, A., Vázquez Carreño, A., & Mejias Risoto, M. (2017). *A Model-Driven Proposal to Execute and Orchestrate Processes: PLM4BS* (pp. 211–225). Springer, Cham. https://doi.org/10.1007/978-3-319-67383-7_16

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

García García, J. A., Escalona, M. J., Martínez-García, A., Parra, C., & Wojdyński, T. (2015). *Clinical Process Management: A model-driven & tool-based proposal*.

GROUP, O. M. (2009). *Software Metrics Metamodel*. http://www.omg.org/spec/SMM/1.0/Beta1/PDF/

Habra, N., Abran, A., Lopez, M., & Sellami, A. (2008). A framework for the design and verification of software measurement methods. *Journal of Systems and Software*, *81*(5), 633–648.

Hikichi, K., Fushida, K., Iida, H., & Matsumoto, K. (2006). *A Software Process Tailoring System Focusing to Quantitative Management Plans* (pp. 441–446). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11767718_41

Hill, J. B., Sinur, J., Flint, D., & Melenovsky, M. J. (2006). Gartner's position on business process management. *Gartner Research G*, *136533*.

IBM. (2017). *IBM - Rational Method Composer*. http://www-03.ibm.com/software/products/en/rmc

IBM Corporation. (2018). *IBM Rational Software Architect Designer*. https://www.ibm.com/us-en/marketplace/rational-software-architect-designer/details?mhq=Rational Software Modeler&mhsrc=ibmsearch_p

ISO/IEC/IEEE 12207-2017-International Standard - Systems and software engineering -- Software life cycle processes. (2017). https://www.iso.org/standard/63712.html

ISO/IEC/IEEE 15288-Systems and software engineering System life cycle processes (Vol. 15288). (2015). www.iso.org

ISO/IEC/IEEE 15939-2017 International Standard - Systems and software engineering--Measurement process. (2017). 1–49. https://doi.org/10.1109/IEEESTD.2017.7907158

ISO/IEC 19507:2012.Information technology - Object Constraint Language (OCL). (2012). https://www.iso.org/standard/57306.html

Jacquet, J.-P., & Abran, A. (1997). From software metrics to software measurement methods: a process model. *Proceedings of IEEE International Symposium on Software Engineering Standards*, 128–135. https://doi.org/10.1109/SESS.1997.595954

Kasunic, M. (2006). *The State of Software Measurement Practice: Results of 2006 Survey* (Issue CMU/SEI-2006-TR-009). http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8095

Kitchenham, B. A., Hughes, R. T., & Linkman, S. G. (2001). Modeling software measurement data. *IEEE Transactions on Software Engineering*, *27*(9), 788–804.

Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: The elusive target. *IEEE Software*, *13*(1), 12.

Kitchenham, B., Pfleeger, S. L., & Fenton, N. (1995). Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, *21*(12), 929–944.

Kleppe, A., Warmer, J., & Bast, W. (2003). MDA Explained: The Model Driven Architecture: Practice and Promise. In *AddisonWesley Professional* (Vol. 83). https://doi.org/10.1016/S0031-9406(05)65759-8

Kuwaiti, M. E., & Kay, J. M. (2000). The role of performance measurement in business process re-engineering. *International Journal of Operations & Production Management*, *20*(12), 1411–1426. https://doi.org/10.1108/01443570010353086

Larrucea, X., & Iturbe, E. (2010). A Metamodel Integration for Metrics and Processes Correlation. *ICSOFT*, 63–68.

Lennselius, B., Wohlin, C., & Vrana, C. (1987). Software metrics: fault content estimation and software process control. *Microprocessors and Microsystems*. http://www.sciencedirect.com/science/article/pii/0141933187905242

Magennis, T. (2015). The Economic Impact of Software Development Process Choice -- Cycle-Time Analysis and Monte Carlo Simulation Results. *48th Hawaii International Conference on System Sciences*, 5055–5064. https://doi.org/10.1109/HICSS.2015.599

Martin, J. N. (2015). Architecture Definition -- A New Process in the ISO International Systems Engineering Standard. *INCOSE International Symposium*, *25*(1), 463–472. https://doi.org/10.1002/j.2334-5837.2015.00075.x

McGarry, J. (2002). *Practical software measurement: objective information for decision makers*. Addison-Wesley Professional.

McLeod, Raymond and Schell, G. P. (1990). *Management information systems*. Upper Saddle River, N.J. : Pearson/Prentice Hall.

Meidan, A., García-García, J. A., Escalona, M. J., & Ramos, I. (2016). A survey on business processes management suites. *Computer Standards & Interfaces*. https://doi.org/10.1016/j.csi.2016.06.003

Meidan, Ayman, García-García, J. A., Ramos, I., & Escalona, M. J. (2018). Measuring Software Process: A Systematic Mapping Study. *ACM Computing Surveys*, *51*(3), 58:1--58:32. https://doi.org/10.1145/3186888

Montgomery, D. (2009). Introduction to statistical quality control. In *John Wiley & Sons Inc.* https://doi.org/10.1002/1521-3773(20010316)40:6<9823::AID-ANIE9823>3.3.CO;2-C

Mora, B., Garcia, F., Ruiz, F., & Piattini, M. (2009). Model-driven software measurement framework: A case study. *Quality Software, 2009. QSIC'09. 9th International Conference On*, 239–248.

Mora, B., Garcia, F., Ruiz, F., Piattini, M., Boronat, A., Gomez, A., Carsi, J. A., & Ramos, I. (2008). Software generic measurement framework based on MDA. *IEEE Latin America Transactions*, *6*(4), 363–370.

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

Nissen, M. E. (1998). Redesigning Reengineering through Measurement-Driven Inference. *MIS Quarterly*, *22*(4), 509. https://doi.org/10.2307/249553

Object Management Group. (2015). *Model Driven Architecture*. http://www.omg.org/mda/specs.htm

OMG. (2002). *SPEM 2.0 Software & Systems Process Engineering Metamodel specification*. http://www.omg.org/spec/SPEM/

Ordonez, M. J., & Haddad, H. M. (2008). The State of Metrics in Software Industry. *Fifth International Conference on Information Technology: New Generations (Itng 2008)*, 453–458. https://doi.org/10.1109/ITNG.2008.106

Pandazo, K., Shollo, A., Staron, M., & Meding, W. (2010). Presenting software metrics indicators: a case study. *Proceedings of the 20th International Conference on Software Product and Process Measurement (MENSURA)*, *20*(1).

Park, R. E., Goethert, W. B., & Florac, W. A. (1996). *Goal-Driven Software Measurement. A Guidebook.*

Perez-Alvarez, J. M., Gomez-Lopez, M. T., Parody, L., & Gasca, R. M. (2016). Process Instance Query Language to Include Process Performance Indicators in DMN. *IEEE 20th International Enterprise Distributed Object*

Computing Workshop (EDOCW), 1–8. https://doi.org/10.1109/EDOCW.2016.7584381

Popova, V., & Sharpanskykh, A. (2010). Modeling organizational performance indicators. *Information Systems*, *35*(4), 505–527. https://doi.org/10.1016/j.is.2009.12.001

Ruiz-gonzález, F., & Canfora, G. (2004). Software Process : Characteristics , Technology and Environments. *CEPIS-UPGRADE*, *V*(5), 5–10.

Ruiz, M., Ramos, I., & Toro, M. (2002). A Dynamic Integrated Framework for Software Process Improvement. *Software Quality Journal*, *10*(2), 181–194. https://doi.org/10.1023/A:1020580008694

Sánchez González, L., García Rubio, F., Ruiz González, F., & Piattini Velthuis, M. (2010). Measurement in business processes: a systematic review. *Business Process Management Journal*, *16*(1), 114–134. https://doi.org/10.1108/14637151011017976

Schmidt, D. C. (2006). Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, *39*(2), 25.

Singh, Y., & Sood, M. (2009). Model Driven Architecture: A Perspective. *Advance Computing Conference, 2009. IACC 2009. IEEE International*, *March*, 1644–1652. https://doi.org/10.1109/IADCC.2009.4809264

Sparx Systems. (2018). *Full Lifecycle Modeling for Business, Software and Systems*. https://sparxsystems.com/products/ea/

Staron, M., & Meding, W. (2009). Using models to develop measurement systems: a method and its industrial use. *Software Process and Product Measurement*, 212–226. https://doi.org/10.1007/978-3-642-05415-0_16

Staron, M., Meding, W., Hansson, J., Höglund, C., Niesel, K., & Bergmann, V. (2014). Dashboards for Continuous Monitoring of Quality for Software Product under Development. In *Relating System Quality and Software Architecture* (pp. 209–229). Elsevier. https://doi.org/10.1016/B978-0-12-417009-4.00008-9

Staron, M., Meding, W., Karlsson, G., & Nilsson, C. (2011). Developing measurement systems: an industrial case study. *Journal of Software Maintenance and Evolution: Research and Practice*, *23*(2), 89–107. https://doi.org/10.1002/smr.470

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

Tahir, T., Rasool, G., & Gencel, C. (2016). A systematic literature review on software measurement programs. *Information and Software Technology*, *73*, 101–121. https://doi.org/10.1016/j.infsof.2016.01.014

Tihinen, M., Kommeren, R., Systems, D., Rotherham, J., & Office, P. M. (2012). Metrics and Measurements in Global Software Development. *International Journal on Advances in Software*, *5*(3), 278–292.

van der Aalst, W.M.P. (2004). Business process management: a personal view. *Business Process Management Journal*, *10*(2), bpmj.2004.15710baa.001. https://doi.org/10.1108/bpmj.2004.15710baa.001

van der Aalst, Wil M. P. (2004). *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management* (pp. 1–65). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-27755-2_1

Wang, X., Ren, A., & Liu, X. (2017). *Researching on quantitative project management plan and implementation method*. 020176. https://doi.org/10.1063/1.4992993

Wang, Y., He, Q., Kliewer, C., Khoo, T., Chiew, V., Nikoforuk, W., & Chen, L. (2002). *Product and Process Metrics: A Software Engineering Measurement Expert System* (pp. 337–350). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36209-6_29

Zhang, H., Keung, J., Kitchenham, B., & Jeffery, R. (2008). Semi-quantitative Modeling for Managing Software Development Processes. *19th Australian Conference on Software Engineering (Aswec 2008)*, 66–75. https://doi.org/10.1109/ASWEC.2008.4483194

Zhang, Y., & Sheth, D. (2006). Mining software repositories for model-driven development. *IEEE Software*, *23*(1), 82–90. https://doi.org/10.1109/MS.2006.23