Tools Integration for Supporting Software Measurement: A Systematic Literature Review

Vinícius Soares Fonseca Monalessa Perini Barcellos Ricardo de Almeida Falbo

Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo
Vitória, ES, Brazil
{vsfonseca, monalessa, falbo}@inf.ufes.br

Abstract. *Software measurement (SM) is a key area to support process quality improvement and project management. Due to the nature of the measurement activities, tool support is essential. Tools can be combined to support the SM process and provide necessary information for decision making. However, tools are usually developed without concern for integration. As a result, organizations have to deal with integration issues to enable communication between tools. Aiming at investigating studies in the literature that report initiatives involving tool integration for supporting SM, we performed a systematic literature review. Twelve initiatives were found. This paper presents the results of the systematic review and discusses the main findings.*

1. Introduction

Software measurement (SM) is a process applied by organizations in several contexts. For instance, in project management, software measurement is used to develop realistic plans, to monitor the progress of projects, to identify problems and to justify decisions [McGarry *et al*. 2012]. In process improvement initiatives, measurement supports analyzing process behavior, identifying needs for improvement and predicting if processes will be able to achieve the established goals [Florac and Carleton 1999].

Fenton and Pfleeger (1997) state that measuring software products, processes and projects is crucial for software organizations, because measures quantify attributes and allow people to get relevant information about the work done and to be done. In the context of software projects, developers can use measurement to verify requirements consistency and completeness, design quality, source code size, defects and test coverage, among others. Project managers, in turn, can use measurement to evaluate when the project will be finished and if the budget will be enough. Clients also can benefit from information provided by measurement. For instance, measures can be used to show if the final product is in conformance to the established standards and satisfies the agreed requirements.

The main purpose of measurement is to provide quantitative information to support decision making [Fenton and Neil 2000]. In this sense, measurement should be applied to several software processes (e.g., project management, requirement engineering, testing, etc.) in order to provide information needed to well-informed decision making at project and organizational levels.

Organizations use different tools to support software processes. For example, schedule and budget tools can be used to support project management activities, CASE tools support requirements engineering, and development environments support implementation and source code management. Despite these tools are not usually conceived to support software measurement, they can help collect and store useful data related to the supported processes (e.g., number of defects, time and cost spent on project activities, number of lines of code, test failure rate, etc.).

In order to provide consistent data and generate useful information for the software measurement process, tools should be integrated. However, this is not an easy task. In general, each tool runs independently and implements its own data and behavioral models, which are not shared between different tools, leading to several conflicts [Izza 2009].

Considering this scenario, we decided to investigate the literature by searching for initiatives involving tool integration to support software measurement. Aiming to reduce bias and ensure the study repeatability, in a previous work [Fonseca, Barcellos and Falbo 2015], we carried out a systematic mapping. Systematic mappings provide an overview of a research topic considering the evidences about that topic in the literature [Kitchenham and Charters 2007]. As pointed out by Kitchenham *et al.*(2011), a systematic mapping can be used as the starting point for undertaking systematic literature reviews, reducing the effort required to perform such subsequent studies. Systematic literature reviews allow a deep investigation concerning more specific research questions [Kitchenham and Charters 2007].

In line with Kitchenham *et al.*(2011), our systematic mapping results revealed some issues that we decided to explore in depth through a systematic literature review. In this paper we present and discuss the main results of the systematic literature review. The paper is organized as follows: Section 2 concerns the paper background, addressing software measurement and integration; Section 3 talks about secondary studies and describes the process followed in the performed study; Section 4 addresses the study itself, presenting the research protocol, the obtained results and some discussions about them; Section 5 discusses some related works; and Section 6 presents our final considerations.

2. Background

2.1. Software Measurement

Software measurement is the continuous process of defining, collecting, and analyzing data regarding software processes and products in order to understand and control them, as well as supply meaningful information to their improvement [Solingen and Berghout 1999]. It is a primary support process for managing projects, and it is also a key discipline in evaluating the quality of software products and the performance and capability of organizational software processes [ISO 2007].

Effective measurement helps software organizations succeed by enabling them to understand their capabilities, so that they can develop achievable plans for producing and delivering products and services. Measurement also helps

organizations to detect trends and anticipate problems, providing better costs control, reducing risks, improving quality, and ensuring that business goals are achievable [Florac and Carleton 1999].

There are some standards and methodologies devoted to assist organizations in defining their software measurement processes, such as ISO/IEC 15939 [ISO 2007], PSM (Practical Software Measurement) [McGarry *et al.* 2012] and IEEE Std. 1061 [IEEE 1998]. Although there are some differences among them, in general the software measurement process includes: measurement planning, measurement execution, and measurement evaluation [ISO 2007].

For performing software measurement, initially, an organization must plan it. Based on its goals, the organization has to define which entities (processes, products and so on) are to be considered for software measurement and which of their properties (size, cost, time, etc.) are to be measured. The organization has also to define which measures are to be used to quantify those properties. For each measure, an operational definition should be specified, indicating, among others, how the measure must be collected and analyzed. Once planned, measurement can start. Measurement execution involves collecting data for the defined measures, storing and analyzing them. Data analysis provides information to decision making, supporting the identification of appropriate actions. Finally, the measurement process and its products should be evaluated in order to identify potential improvements [Barcellos, Falbo and Rocha 2010].

In addition to standards and methodologies that address the software measurement process as a whole, there are some proposals that deal with more specific aspects of the measurement process. In this context, GQM (Goal Question Metric) [Basili, Rombach and Caldiera 2004] can be highlighted. It represents a systematic approach for tailoring and integrating goals to software processes, products and quality perspectives of interest, based upon project and organizational specific needs [Basili, Rombach and Caldiera 2004]. GQM considers three levels:

- **Conceptual Level (Goal):** A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. The objects of measurement are products (e.g., artifacts, specifications, programs), processes (e.g., designing, testing) and resources (e.g. software, hardware, personnel).
- **Operational Level (Question):** A set of questions is used to characterize the way assessment/achievement of a specific goal will be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.
- **Quantitative Level (Metric):** Measures are associated with each question in order to answer it in a quantitative way.

GQM levels are organized in a hierarchical structure starting with a goal. The goal is refined into several questions that usually break down the issue into its major components. Each question is then refined into metrics (measures). The same measure

can be used to answer different questions under the same goal [Basili, Rombach and Caldiera 2004].

While GQM model elaboration starts top-down, measurement data is interpreted bottom-up. As the measures are defined with an explicit goal in mind, the information provided by them should be interpreted and analyzed with respect to this goal, to conclude whether or not it is attained [Solingen and Berghout 1999]. Figure 1 illustrates the GQM hierarchical structure.

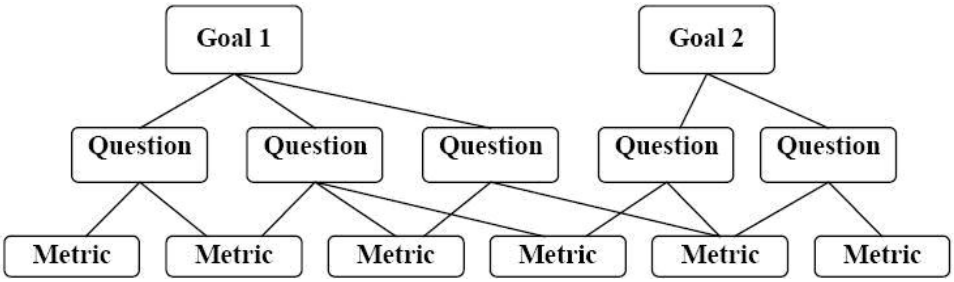

Figure 1. GQM model hierarchical structure [Basili, Rombach and Caldiera 2004].

Park *et al.* (1996) propose a variation of GQM introducing an "indicator" definition step, making it GQ(I)M. Indicators are measures directly used to monitor goal achievement [Barcellos et al., 2013]. They display one or more measurement results and are designed to communicate or explain the significance of those results against the established measurement goals. Seeing which measurement data has to be analyzed (i.e., data collected to which measure) and how they will be displayed help to point to and clarify exactly what someone must measure [Park, Goethert and Florac 1996]. Figure 2 illustrates GQ(I)M structure.

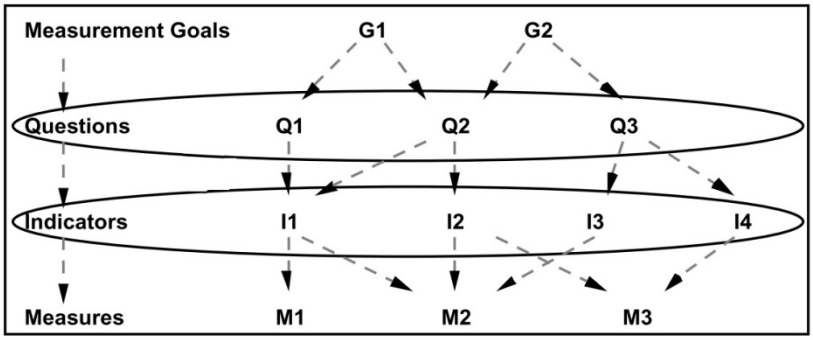

Figure 2. GQ(I)M structure. Adapted from [Park, Goethert and Florac 1996].

Figure 3 presents examples of GQM and GQ(I)M models (data plotted in the graph is hypothetical and merely illustrative). In (a), information to monitor the measurement goal is provided by the measure *Annual Cost with Rework.* However, in order to verify if the measurement goal was achieved, it is not enough to look at data collected for that measure. It is necessary to analyze the difference between the values related to a year and the previous one. In (b) the indicator directly used to monitor the measurement goal is explicitly defined (*Decreasing of Annual Cost with Rework)* and displayed in a graph in order to show whether the measurement

goal was achieved.

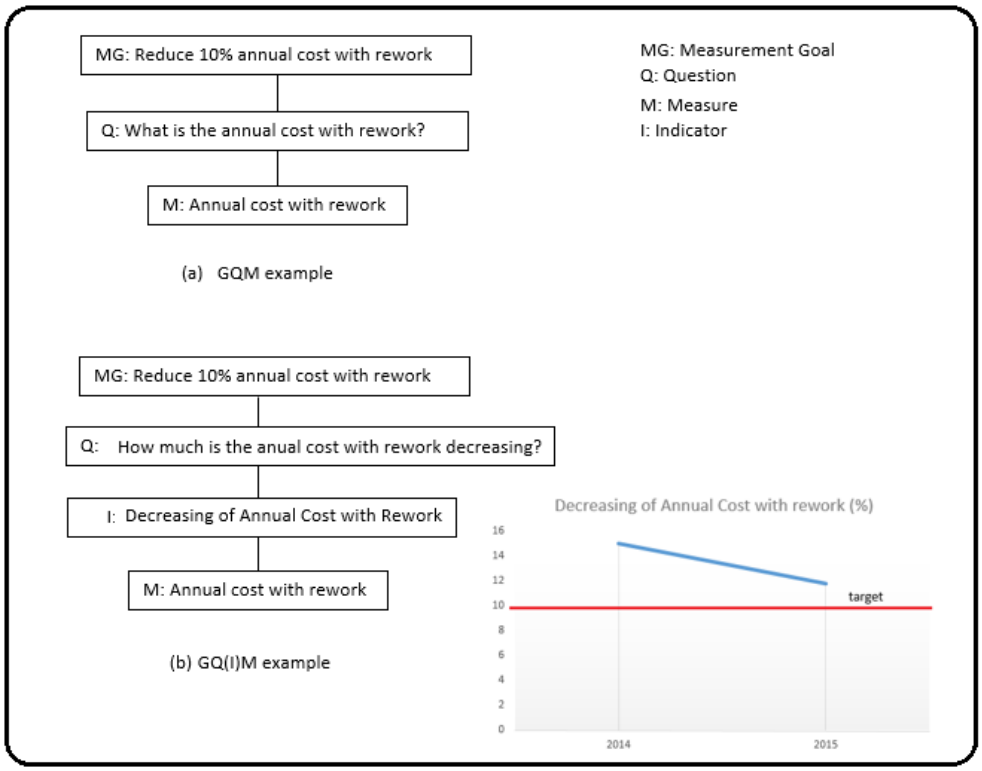

Figure 3. GQM and GQ(I)M examples.

2.2. Integration and Interoperability

Integration can be defined as the act of incorporating components into a complete set, conferring it some expected properties. The components are combined in a way to form a new system constituting a whole and creating synergy [Izza 2009].

Interoperability, in turn, can be understood as the ability of applications or application components to exchange data and services [Wegner 1996]. Interoperability provides two or more business entities (from the same organization or different organizations and irrespective of their location) with the ability of exchanging or sharing information (wherever it is and at any time) and of using functionality of one another in a distributed and heterogeneous environment. It preserves component systems as they are [Vernadat 2007].

Due to the interrelation between the terms integration and interoperability, they are often used in an indistinct way [Nardi, Falbo and Almeida 2013]. In this paper, the term integration is adopted in a broader sense, covering both integration and interoperability meaning.

For a single organization, integration means that it is necessary to create a coherent information system architecture in which the various administrative and business processes, information stores and systems are integrated so that they appear

seamless from the point of view of the individual user [Vernadat 2007]. In other words, it is necessary to define an integrated system as a collection of subsystems that interact to form a whole and whose properties emerge due to the interaction of its sub-systems [Pokraev 2009].

Integration can be extended to several organizations that integrate their applications because the emerging properties of the integrated system have value for them. Examples of such emerging properties are more efficient usage of the available resources, flexibility and adaptability of business processes, and increased market reach [Pokraev 2009].

Integration is a difficult and complex process [Themistocleous, Irani and Love 2004]. Organizations have been using an increasing number of applications to support their processes. In general, these applications are standalone software, defined in isolation, and operated autonomously supporting specific parts of the whole business process [Vernadat 2007]. They are based on different standards, computing languages, platforms and operating systems, which cause various integration problems. There is also the complexity of existing applications, which in many cases have fixed and rigid structures for messages, interfaces and databases. Moreover, there is a lack of documentation, especially as legacy systems have often emerged over the time without any strategy. Many legacy systems have existed in organizations for more than 25 years and their technical documentation was either not created or lost during the years [Themistocleous, Irani and Love 2004].

In sum, applications to be integrated often have not been designed to work together, i.e., they are heterogeneous, autonomous and distributed (HAD) applications. “Heterogeneous” means that each enterprise application implements its own data and process models. Heterogeneity exposes a particular difficulty relying on multiple technical, syntactical and semantic conflicts, which require a mediation process to deal with the differences. “Autonomous” means that applications may run independently of any other application. Autonomy poses a particular difficulty in interconnecting systems, requiring a solution that deals with asynchronous behavior during flow exchanges. “Distributed” means that applications locally implement their data model, which they generally do not share with other applications. Distribution mainly poses difficulties on transaction control [Izza 2009].

Integration can be performed considering different dimensions. Izza (2009) proposed a framework synthesizing integration approaches through four main dimensions: scope, viewpoint, layer and level. Figure 4 illustrates the integration dimensions.

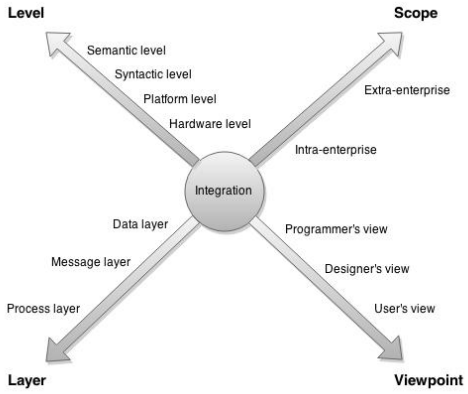

Figure 4. Integration dimensions [Izza 2009].

Scope dimension distinguishes two main approaches: intra-enterprise and inter-enterprise integration. Intra-enterprise integration concerns scenarios that imply internal enterprise applications. Extra-enterprise integration aims to connect applications from different partners [Izza 2009].

Regarding *viewpoint* dimension, three main viewpoints are considered: user's view (external), which concerns the different views from domain experts and business users; designer's view (conceptual), which concerns the different models used during information system design; and programmer's view (internal), which refers to information system implementation [Izza 2009].

Regarding *layers*, integration can address one or several information system layers. Data integration deals with moving or federating data between multiple data stores. Integration at this layer assumes bypassing the application logic and manipulating data directly in the database, through its native interface. Message or service integration addresses messages exchange between the integrated applications. Any tier of an application, such as GUI, application logic or database, can originate or consume the message. Process integration views enterprises as a set of interrelated processes and it is responsible for handling message flows, implementing rules and defining the overall process execution. It constitutes the most complex integration approach [Izza 2009].

With respect to integration *levels*, four main levels can be distinguished: hardware, platform, syntactical and semantic levels. Hardware level covers differences in computer hardware, networks, etc. Platform level encompasses differences in operating system, database platform, etc. Syntactical level addresses the way the data model and operation signatures are written down. Semantic level deals with the intended meaning of the concepts in a data schema or operation signature. Each level depends on the previous one, so it is not possible to consider semantics if syntax is not considered yet [Izza 2009].

Challenges in application integration arise, among others, from the fact that heterogeneous applications employ different data and behavioral models, leading to semantic conflicts. These conflicts occur whenever applications are built with

different conceptualizations, which can impact the integration of data, services and processes [Nardi, Falbo and Almeida 2013].

3. Research Overview

We are interested in investigating initiatives involving tool integration to support software measurement. Thus, we decided to search the literature for such initiatives and we started by doing a tertiary review.

A tertiary review is a study that investigates secondary studies regarding a research topic. Secondary studies, in turn, are studies based on analyzing research papers (referred as primary studies) [Kitchenham, Budgen and Brereton 2011]. Systematic literature reviews and mapping studies are examples of secondary studies. We did not find any secondary study about integrating tools to support software measurement. Hence, we decided to carry out such study. We started by performing a mapping study in which we investigate general aspects of initiatives involving tool integration to support software measurement [Fonseca, Barcellos and Falbo 2015].

A mapping study provides a broad overview of a research area in order to determine whether there is research evidence on a particular topic [Kitchenham and Charters 2007]. Mapping studies help identifying gaps in order to suggest areas for future research and provide a map that allows appropriately to position new research activities. Moreover, results of a mapping study may identify suitable areas for performing systematic reviews of the literature [Kitchenham, Budgen and Brereton 2011]. In this sense, the results obtained from the systematic mapping pointed out aspects that should be deeper investigated. Thus, we carried out a systematic literature review in order to investigate them.

Systematic literature reviews (SLR) are secondary studies used to find, critically evaluate and aggregate all relevant research papers on a specific research question or research topic [Kitchenham, Budgen and Brereton 2011]. SLRs allow identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest [Kitchenham and Charters 2007].

All the performed studies followed the approach defined in [Kitchenham and Charters 2007], which is composed of three main activities: *planning*, when the research protocol is defined with the purpose of supporting study repeatability as well as helping researchers to avoid bias when conducting the review; *conducting*, when the protocol is executed and data are extracted, analyzed and recorded; and *reporting*, when the results are recorded and made available to potential interested parties.

The following electronic databases were searched during the studies:

- IEEE Xplore (http://ieeexplore.ieee.org)
- ACM Digital Library (http://dl.acm.org)
- Springer Link (http://www.springerlink.com)
- Scopus (http://www.scopus.com),

- Science Direct (http://www.sciencedirect.com)
- Engineering Village (http://www.engineeringvillage.com)

Concerning the tertiary study, the search was done using a search string containing four groups of terms joined with the operator AND. The first group includes terms related to integration and interoperability. The second group includes terms related to software measurement. The third group includes terms related to tools and applications. The fourth group includes terms related to systematic mapping and SLRs. Within the groups, we used the OR operator to allow synonyms. The following search string was used:

("integration" OR "interoperability" OR "interoperable" OR "integrated") AND ("software measurement" OR "software process measurement" OR "software project measurement" OR "software engineering measurement" OR "software product measurement") AND ("tool" OR "application" OR "system" OR "framework" OR "suite" OR "toolkit") AND ("systematic literature review" OR "systematic review" OR "systematic mapping" OR "mapping study" OR "systematic literature mapping")

The search string was applied in three metadata fields (title, abstract and keywords) and 60 publications were returned. Then, we applied the following selection criterion: the publication addresses a systematic literature review or a mapping study about tool integration to support software measurement. However, none of the publications met the criterion. For instance, the publication [Mohammed and Mohammad, 2015] was returned from Springer Link database. It presented a systematic literature review, but it was not about tool integration to support software measurement.

After the tertiary study, we performed a mapping study [Fonseca, Barcellos and Falbo 2015]. 12 initiatives involving tool integration to support software measurement were found and their main characteristics were analyzed. The mapping study results provided a panorama regarding the research topic, showing when and where research in this topic has been published, the types of research done, and an overview of the initiatives. Aspects such as types of tools, categories of measures, integration layers and levels addressed, among others, were investigated during the mapping study.

As we argued before, a mapping study provides a broad view of a research area and its results may point issues that can be investigated in systematic literature reviews, since this kind of secondary study allows deeper investigation into the identified issues [Kitchenham et al., 2011]. In this sense, after the mapping study, we identified some issues we should investigate in deep:

(i) In the mapping, we identified the measurement activities supported by the initiatives. Now, we should investigate how the support is provided.

(ii) In the mapping we identified categories of measures addressed by the initiatives. Now, we should look at the measures addressed and also the processes that were measured in the initiative.

(iii) In the mapping we identified types of tools involved in the integration initiative. Now, we should investigate the tools involved, the measurement activities supported by each one of them, and how the tools support the activities.

(iv) In the mapping, we identified the integration layers and levels addressed. Now, we should also investigate the other integration dimensions and look in details at how the integration is performed.

Taking these issues into account, we established new research questions and carried out a SLR to answer them. We ran the same search string used in the mapping study and, although the period considered was a bit longer, no new papers regarding software measurement tool integration were found. Therefore, the selected publications were the same, but now the initiatives found in the mapping study were deeper analyzed during the SLR aiming to answer the new research questions. Next, the SLR is presented in details.

4. The Systematic Literature Review

4.1. Research Protocol

In this section we present the main parts of the research protocol used to perform the systematic literature review (SLR).

***SLR goal*:** the goal of this SLR is to investigate initiatives involving tool integration to support software measurement (SM).

***Research Questions*:** For achieving the SLR goal, we defined a main research question to be answered: *What are the tool integration initiatives aiming at supporting software measurement?* With this main question in mind, we defined four specific research questions (RQ) regarding three main aspects: Measurement, Tools and Integration.

RQ1 *(Measurement) - Which are the activities of the SM process (measurement planning, data collection, and data analysis) supported by the integrated set of tools and how is the support provided?*: The purpose of this question is to identify which measurement activities are supported by the initiatives in order to evaluate the coverage of the resulting set of integrated tools, as well as to explain how the support is provided. The activities considered are the two first activities established in [ISO 2007] (measurement planning and measurement execution). Measurement execution was split for allowing us to verify if the tools support both data collection (which involves data collection itself and data storage) and data analysis, or only one of them.

RQ2 *(Measurement) - Which are the measures considered in the integration initiative and what are the main software processes measured by them?:* This question aims at identifying which measures have been considered in the initiatives and the main software processes measured by them, allowing us to analyze how specific or comprehensive is the measurement scope, as well as the main processes focused by the integration initiative.

RQ3 *(Tools) - Which are the integrated tools and to which activities of the SM process are they related?:*The rationale of this question is to identify the tools being integrated in each initiative, and which measurement activities they support.

RQ4 *(Integration) - How is the tool integration performed and how can it be categorized according to the scope, viewpoint, layer and level dimensions?:* This question aims to describe and categorize each integration initiative considering the four dimensions proposed in Izza's framework [Izza 2009]: scope, viewpoint, layer and level.

***Search String*:** the following search string was applied to the digital libraries cited in the previous section. As it can be noticed, the search string resulted from excluding the fourth group of terms from the string used in the tertiary review.

("integration" OR "interoperability" OR "interoperable" OR "integrated") AND ("software measurement" OR "software process measurement" OR "software project measurement" OR "software engineering measurement" OR "software product measurement") AND ("tool" OR "application" OR "system" OR "framework" OR "suite" OR "toolkit")

In order to establish the search string, we selected some relevant papers during the informal literature review that preceded the SLR to be used as control publications, meaning that they should be selected by the search string used in the study. Thus, we defined and tested several different search strings until selecting the one to be used, which was the one that provided better results in terms of relevance and number of returned publications.

***Publications Selection*:** selection was performed in five steps:

Step 1 – Primary selection and cataloging: the search string was applied in the search mechanisms of the selected sources. Publication type was limited to papers from the Computer Science and Engineering area. At the end of this step, 948 publications were returned.

Step 2 – Duplicate removal: studies indexed by more than one digital library were identified and the duplications were removed. 85 publications were removed, resulting in 863 studies at the end of this step.

Step 3 – Selection of Relevant Publications –1st Filter: the title, abstract and keywords of the selected publications were analyzed considering the following inclusion (IC) and exclusion (EC) criteria: (IC1) the publication presents information regarding integration among tools, applications or systems that support software measurement; (EC1) the publication does not have an abstract; (EC2) the publication is published as an abstract; and (EC3) the publication is not a primary study. As a result of this step, 24 studies were selected (a reduction of approximately 97%).

Step 4 – Selection of Relevant Publications –2nd Filter: the full text of the publications selected in S3 was read with the purpose of identifying the ones that provide useful information. Thereby, the inclusion criterion IC1 was considered and also the following exclusion criteria: (EC4) the publication is

not written in English; (EC5) the publication full text is not available; and (EC6) the publication is a copy or an older version of an already considered publication. 8 studies were selected in this step.

Step 5 – Snowballing: as suggested in [Kitchenham and Charters 2007], the references of publications selected in the study must be analyzed and, if some of them seem to present evidence related to the research topic, it should be assessed by the selection criteria and included in the study. Thus, in this step, references of the publications selected in the previous step were investigated by applying the first and second filters. As a result, 4 new publications were selected.

Figure 5 illustrates the process followed to select publications, which resulted in 12 selected publications.

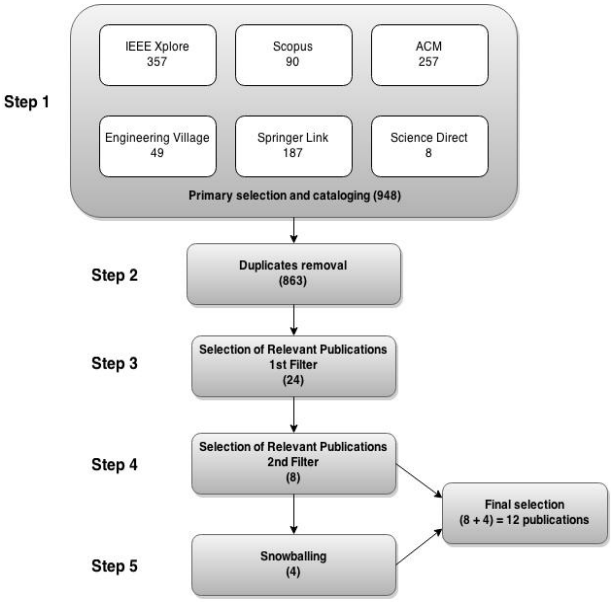

Figure 5. Publication Selection Process.

4.2. Data Synthesis

In this section we present the main results obtained considering each research question (RQ).

Main RQ - What are the tool integration initiatives aiming at supporting software measurement?

Table 1 presents the twelve initiatives identified in the SLR, answering the main research question.

Table 1. Tool Integration Initiatives that Support Software Measurement

Proposal	Year	Description
TAME [Basili and Rombach 1988]	1988	TAME (Tailoring A Measurement Environment) system is an Integrated Software Engineering Environment composed by several integrated components. TAME integrates three measurement tools that capture data from Ada source and generate measures.

Table 1. Tool Integration Initiatives that Support Software Measurement (cont.)

Proposal	Year	Description
Tool Support for SM [Tian, Troster and Palma 1997]	1997	This initiative uses a set of integrated tools in order to support software measurement and quality improvement. A tool that supports tree-modeling analysis (S-PLUS) is the central analysis tool. Other tools are used for data gathering, analysis and result presentation. The tools are connected to S-PLUS, either as information consumer or as information provider.
GQM Tool [Lavazza 2000]	2000	This proposal presents a GQM tool supporting measure definition, data collection, analysis and feedback. It has interface with a configuration management system and other measurement tools.
MetriFlame [Komi-Sirvio, Parviainen and Ronkainen 2001]	2001	MetriFlame is a measurement automation tool based on GQM that uses existing data recorded during software development process. It has components for collecting and converting measurement data from various tools, spreadsheets and databases.
DSS [Chulani*et al.* 2012]	2003	DSS is a Decision Support System developed at IBM for tracking and using software measures, aiming to enable executives to make better informed decisions in supporting their products. It captures (from different host systems) data regarding customer support, critical situations and customer satisfaction, and integrates these data into a data warehouse.
SM in a CI Environment [Moreira *et al.* 2010]	2010	This approach uses a Continuous Integration (CI) engine in order to automate measurement data extraction. It follows CMMI Measurement and Analysis process area practices and GQIM concepts for selecting relevant measures. Data collection is done by several tools.
SOFAS [Ghezzi and Gall 2011]	2011	SOFAS is a platform that offers software analysis services in order to allow interoperability among analysis tools. It is made up of three main constituents: Software Analysis Web Services, which provides a catalogue of services for data analysis; Software Analysis Broker, acting as the service manager and the interface between the services and the users; and Software Analysis Ontologies, which defines and represents the data consumed and produced by the different services.
Dione [Caglayan*et al.* 2012]	2012	Dione is a Java web application whose major functions are: i) build a measurement repository that contains product and process measures, as well as information about defective software components; ii) analyze trends in measures and issues using chart and report configurations; and iii) construct and calibrate customized defect prediction models to predict defect proneness of a software product version or release. It collects data from several tools and uses a smart client to connect with software development artifacts and automatically extract measures. It also supports integration with other tools through web services.
QualitySpy [Jureczko and Magott 2012]	2012	QualitySpy is a framework for monitoring the software development process. It collects raw data from several integrated tools, as well as from the source code, and provides analysis reports.
3C [Janus *et al.* 2012]	2012	3C Approach is an extension to the CI practice and addresses Continuous Measurement and Continuous Improvement as subsequent activities to Continuous Integration. Several Java tools and a version control system were integrated into the CI engine Cruise Control, allowing collection of measures related to source code and test coverage.
ASSIST [Keser, Iyidogan and Ozkan 2013]	2013	ASSIST is an integrated tool developed by a CMMI level 3 organization. It adopts GQ(I)M approach and is connected with commercial software suites for project management, issue tracking and enterprise resource planning (ERP).

Table 1. Tool Integration Initiatives that Support Software Measurement (cont.)

Proposal	Year	Description
DePress [Madeyski and Majchrzak 2014]	2014	DePress is an open source, extensible framework for software measurement and data integration, which can be used for prediction purposes (e.g. defect prediction, effort prediction) and software changes analysis (e.g., release notes, bug statistics). It supports the integrated use (through KNIME Framework) of the issue tracking systems JIRA and Bugzilla, the software configuration management systems Subversion and Git, and the measurement tools Judy, JaCoCo, EclipseMetrics, CheckStyle and PMD.

RQ1 (Measurement) - Which are the activities of the SM process (measurement planning, data collection, and data analysis) supported by the integrated set of tools and how is the support provided?

Measurement planning activity is supported by four of the twelve analyzed studies (TAME, GQM Tool, MetriFlame and ASSIST) and all of them use GQM or GQ(I)M. TAME provides GQM templates to goal definition and refinement into questions and measures. GQM Tool enables the edition of defined goals by means of predefined forms and verifies the structural consistency of plans (e.g., by checking whether each question is connected with a goal and refined into measures). ASSIST uses GQ(I)M and includes a pool of well-structured business goals-questions-indicators-measures that can be queried, viewed and examined. This pool allows the reuse of the same set of sound measurement constructs in goal setting and planning activities performed by project managers and upper level management. MetriFlame does not present details about GQM usage, but its authors state that the tool can manage GQM plans. In SM in a CI Environment, measurement planning is done by using GQ(I)M, however, it is done manually before the use of the integrated tools. Therefore, we considered that this initiative does not support measurement planning.

Unlikely measurement planning, data collection activity is supported in all studies. All initiatives support data collection by integrating tools that act as data input tools. Tool Support for SM, GQM Tool, MetriFlame, DSS, SM in a CI Environment, SOFAS, QualitySpy and 3C obtain measurement data only from external tools. TAME, Dione and ASSIST also provide a mechanism for data input as a result of the integrated measurement support.

Data analysis is also supported in all studies. The initiatives present analysis features varying from simple reports to sophisticated analysis tools. TAME, MetriFlame, QualitySpy, 3C and DePress present simple report tools, i.e., they include a module or a tool that generates limited and fixed graphs or reports for viewing measurement results. Tool Support for SM, GQM Tool, DSS, SM in a CI environment, SOFAS, Dione and ASSIST, in turn, present at least one sophisticated analysis tool, providing flexible and dynamic views, graphs and reports about the collected measurement data.

RQ2 (Measurement) - Which are the measures considered in the integration initiative, and what are the main software processes measured by them?

Table 2 enumerates the measures addressed in each proposal, and the main processes measured. Some proposals focus on a single process. Others focus on more

than one process. There are also initiatives in which the measured processes are not previously defined, and depend on data available from the integrated tools.

Most of the measures addressed by the initiatives are related to code. As a consequence, Coding is the software process focused by most of the proposals. Some of them address only measures related to code (e.g., SOFAS). Others have also measures related to other processes (e.g., TAME and SM in a CI Environment also has measures related to Testing), but the main measured process is Coding.

Some proposals split their focus between Coding and another process: ASSIST measures mainly Project Management and Coding, DePress focus on Coding and Configuration Management, and 3C focus on Coding and Testing.

MetriFlame addresses measures related to the software development process, however the authors do not specify which are the measures, since they depend on data available from the integrated tools. Consequently, it is not possible to identify which processes related to software development (e.g., Coding, Design) are measured, because this depends on the addressed measures. ASSIST also allows defining measures according to available data. In this sense, depending on the application context, it could measure other processes than the ones cited in Table 2.

Table 2. Measures and Main Measured Software Processes

Proposal	Measures	Main Software Processes
TAME	Lines of code, structural complexity measures, data binding measures, test coverage.	Coding
Tool Support for SM	Number of defect fixes, code complexity, internal measures, predictive modeling linking code measures, failure arrivals, execution time, time of failure instances, number of test runs, number of processed transactions, estimated reliability (number of successes over the number of test runs), predicted success rate, measures related to design, size, changes, defects, tests, transactions.	Coding
GQM Tool	Total number of failures; for each failure: priority, type, detection time, conclusion time; Total number of faults; for each fault: severity, type, phase when originated, detection time, correction time; cyclomatic number; number of classes; methods per class; LOC; lines of comments; size; average complexity and size by component	Coding
MetriFlame	It depends on the data available in the integrated tools, databases and spreadsheets.	It depends on the available data.
DSS	Nature of a problem, Severity, Problem resolution provided, Capability, Ease of use, Performance, Reliability, Ease of installation, Maintainability, Documentation, Service/Support, Overall Satisfaction.	Customer Management
SM in a CI Environment	Number of Lines of Code, Number of instructions, Number of methods, Number of fields, Code Source Cyclomatic Complexity, IL Cyclomatic Complexity, Type rank, Lack of cohesion of methods, Number of children, Depth of inheritance tree, Number of lines of comment, Percentage comment, Afferent coupling at type level, Efferent coupling at type level, Association between class, Percentage coverage (unit tests).	Coding

Table 2. Measures and Main Measured Software Processes (cont.)

Proposal	Measures	Main Software Processes
SOFAS	Fan-In and Fan-Out of classes, methods and packages; McCabe's cyclomatic complexity of classes, methods and packages; LOC of classes, methods and packages; Number of calls in the entire system; Height of inheritance tree of classes; Average hierarchy height; Average number of derived; Number of direct sub-classes of a classes; Number of methods overriding a method in any one of the super-classes of a class; Number of classes; Number of packages; Number of attributes (static and non) of classes and packages; Number of methods (static and non) of classes and Packages; Number of parameters of a method.	Coding
Dione	Cyclomatic complexity per month, LOC size of the project over time, Defect count and defect density per month, Average cyclomatic complexity per month.	Coding
QualitySpy	19 code-related measures calculated by CKJM extended tool.	Coding
3C	Number of tests, Test coverage, Test-Growth-Ratio, Number of broken builds, Total lines, Effective lines, Checkstyle violations, Findbugs priority 1/2/3 rule violations, PMD priority 1/2/3 rule violations.	Coding and Testing
ASSIST	Measures related to project (e.g., estimates and actual values), to product (e.g., Functional size, Code length, Technical properties), to development process (e.g., development team measures) and other measures defined during GQIM, according to the available data.	Project Management and Coding
DePress	Number of issues, Number of unique issues for each file, Defects post-release, CK Java Metrics, Code coverage, Time spent on assigned tasks.	Configuration Management and Coding

The addressed measures are used with several purposes. Many of them are used to support software process improvement. This occurs in TAME, GQM Tool, SM in a CI Environment, QualitySpy and ASSIST. In Tool Support for SM, measures enable software quality assessment based on reliability growth models. In DSS, measures related to customer are used to provide a customer view of the provided services to business executives responsible for multiple software products. SOFAS applies measures to analyze services quality. In Dione and DePress, measures are used to support software defect prediction. In 3C, measures support quality assurance of software products developed by using agile methods. Finally, MetriFlame uses measures to evaluate software development processes and products.

RQ3 (Tools) - Which are the integrated tools, and to which activities of the SM process are they related?

Table 3 presents the tools involved in each initiative and the corresponding measurement activity supported. After Table 3, the measurement support provided by the tools in each proposal is described.

Table 3. Measurement Tools and Software Measurement Activities Supported

Proposal	Measurement Tools and Software Measurement Activities Supported		
	Measurement Planning	Data Collection	Data Analysis
TAME	TAME	TAME, coverage tool, data bindings tool, code measurement tool	TAME
Tool Support for SM	-	IDSS, CMVC, TestLog, SlaveDriver, REFINE, W-Analyzer	S-PLUS, SMERFS, SAS, TreeBrowser

Table 3. Measurement Tools and Software Measurement Activities Supported (cont.)

Proposal	Measurement Tools and Software Measurement Activities Supported		
	Measurement Planning	Data Collection	Data Analysis
GQM Tool	GQM tool	GQM tool, Oracle, PCMS, Krakatau, Resource Standard Metrics	GQM tool, MS Access
MetriFlame	MetriFlame	MetriFlame, Lotus Notes, Paradox, dBASE, IBM DB/2, Informix, Interbase, MS Access, MS SQL Server, Oracle, Sybase, FoxPro, Microsoft Project, Microsoft Excel	MetriFlame
DSS	-	Legacy tools	Decision Support System
SM in a CI Environment	-	Cruise Control.Net, Nant, NUnit, PartCover, NDepend, MS Access	MS Excel, MS SQL Server Analysis Services
SOFAS	-	SOFAS, CVS, Subversion, Git, Bugzilla, Google Code, Trac, SourceForge	SOFAS
Dione	-	Dione, CVS, Subversion, Git, Mercurial, Clearcase, Bugzilla, Jira	Dione
QualitySpy	-	QualitySpy, CKJM extended, Selenium, Jira, Subversion, Hudson	QualitySpy
3C	-	Subversion, Findbugs, Checkstyle, PMD, Cobertura, Cruise Control, JUnit	Cockpit
ASSIST	ASSIST	ASSIST, ERP system, Issue Tracking tool, Project Management tool	ASSIST
DePress	-	Jira, Bugzilla, Subversion, GIT, Judy, JaCoCo, EclipseMetrics, CheckStyle, PMD, DePress	DePress, KNIME Report Designer

TAME: In this initiative, TAME is the main tool and supports the three measurement activities. TAME's architecture is made up of several components. GQM Model Selection and GQM Model Generation components support measurement planning by allowing the creation or reuse of GQM models. Measurement Scheduling, Measurement Tools and Data Entry and Validation components support data collection. They allow, respectively, scheduling automatic data collection, collecting process and product data automatically from three tools (coverage tool, data binding tool and code measurement tool), and entering data manually. GQM Analysis and Feedback and Report Generator components support data analysis. The first one allows analysis according to a specific GQM model and the second offers a variety of reports.

Tool Support for SM: This proposal does not support measurement planning. It uses IBM tools (IDSS - Integrated Development Support System and CMVC - Configuration Management/Version Control), and home-grown applications (TestLog and SlaveDriver) to support data collection from projects databases. REFINE, a reverse engineering toolkit, is used to calculate design and code complexity measures from source code and a tool called W-Analyzer computes product measures. Data analysis is supported by four tools: S-PLUS, a tree-modeling analysis tool that acts as a central analysis tool; SMERFS, which is used when additional analysis models are necessary; SAS, a commercial statistical package that is used for general statistical

modeling; and TreeBrowser, an internal tool that facilitates analysis tree exploration.

GQM Tool: The GQM tool supports measurement planning by allowing creating GQM plans. To aid data collection, PCMS, Krakatau and Resource Standard Metrics tools act as data providers, and MS Access and Oracle are used to store data. Data analysis is supported by the GQM tool, which borrows the computational power from the MS Access query system to run queries associated with GQM plan items and display the results.

MetriFlame: This proposal has the MetriFlame tool as the central tool. It supports GQM plans creation (measurement planning), and collects measurement data from various sources (data collection): Lotus Notes, Paradox, dBASE, IBM DB/2, Informix, Interbase, MS Access, MS SQL Server, Oracle, Sybase, FoxPro, MS Project and MS Excel. MetriFlame tool allows data analysis, supports data representation and results visualization, identifies trends and compares previous and latest results.

DSS: The Decision Support System consists of a data warehouse, an OLAP analytical engine and a web front-end. Data collection is made from three legacy system data sources (customer support, critical situations and customer satisfaction) that are integrated into the data warehouse. To support data analysis, the analytical engine analyzes data from the data warehouse, and the results are presented by the web front-end.

SM in a CI Environment: Several tools are used to support data collection: PartCover is used to provide test code coverage data, NDepend concerns software metrics, Cruise Control.Net acts as a CI engine connecting the other tools; Nant automates build tasks and provides build information; NUnit runs unit tests and report test results. To support data analysis, a data warehouse was created, and an ETL process consolidates data in a data warehouse cube, allowing OLAP operations.

SOFAS: SOFAS architecture is made up of three main constituents: Software Analysis Web Services (SA-WS), Software Analysis Broker (SA-B), and Software Analysis Ontologies (SA-Ontos). Both data collection and data analysis are supported by SA-WS, which collects data through services from CVS, Subversion, Git, Bugzilla, Google Code, Trac and SourceForge. To support data analysis, SA-B provides a services composer that allows data to flow between services, generating combined analysis results.

Dione: For supporting data collection, Dione uses a smart client technology to connect to version control systems (CVS, Subversion, Git, Mercurial, Clearcase) and bug repositories (Bugzilla, Jira). The smart clients automatically extract product and in-process measures data. Data analysis is supported by a web-based reporting module, which allows customized reports according to different stakeholders.

QualitySpy: QualitySpy has two main groups of features. The first group concerns data acquisition and supports data collection. This group is based on several connectors to collect data from Java classes, Subversion, Jira and Hudson. Measures from Java classes are calculated by a connector that wraps the CKJM extended tool. Measures from Jira are collected through the user interface using Selenium. Data

analysis is supported by the second group of features (reporting), which is operated through a web browser and provides reports that can be predefined or customized by the user.

3C: For data collection, Cruise Control is used to extract data from several tools: Subversion, for data related to source code; Findbugs for programming bugs; Checkstyle, for violations of coding standards; PMD, as a hybrid-version of the tools mentioned before; Cobertura, for test coverage; JUnit, for unit tests; and the Cruise Control itself for data regarding builds. A tool called Cockpit was developed to support data analysis. Measurement results are put into graphs that show the changes of measures over time.

ASSIST: ASSIST is composed of several modules. Measurement activities are mainly supported by the Measurement module, which comprises three sub-modules: Metrics, Services and Projects. Measurement planning is supported by the Metrics sub-module, which provides functionalities for defining business goals, questions, indicators and measures, and establishing relationships between them. Measurement planning is also supported by the Projects sub-module, which is used to create project measurement plans. Data collection, in turn, is aided by the Services sub-module, which has manual and automated data collection services to populate a measurement database. Automatic data collection features extracts data from project management, issue tracking and ERP commercial software suites. Data analysis is supported by the Metrics sub-module that allows creating/customizing reports for selected indicators and constructing pivot tables that supplement the indicators. The analysis results (graphical/tabular representations and interpretations or comments made by the user) are stored. Although the main module supporting the measurement process is Measurement, the Project Management module also aids measurement activities. It provides an infrastructure for project data collection, storage and use for project estimation.

DePress: DePress uses an open source framework called KNIME to support data collection and analysis. Data collection is aided by an extension set of KNIME plugins that retrieve data from software configuration management systems (Subversion and Git), issue tracking tools (Jira and Bugzilla) and metric readers (Judy, JaCoCo, EclipseMetrics, CheckStyle and PMD). Data can also be manually inputted. DePress supports data analysis with the help of KNIME Report Designer, which is based on a BIRT tool (Business Intelligence Reporting Tool).

RQ4 (Integration) - How is the tools integration performed and how can it be categorized according to the scope, viewpoint, layer and level dimensions?

Table 4 summarizes the integration approach adopted in each proposal. Next, we classify the initiatives according to the framework proposed by Izza [Izza 2009]. It is worth saying that some publications describe the integration approach in more details than others. Hence, information regarding the integration approaches is heterogeneous and limited to the publication content.

Table 4. Tools Integration Overview

Proposal	Description
TAME	Integration in TAME occurs in the Measurement Tools component. This component is integrated to three measurement tools that collect data from Ada source code and make them available for TAME in a relational database.
Tool Support for SM	The integration approach is based on adopting external rules for data contents and formats (on which all the parties involved have to agree on), using common tools for multiple purposes, and utility programs to convert data for interoperability of tools. There is a central analysis tool (S-PLUS) and the other integrated tools (internal and commercial off-the-shelf tools) support data capturing, analysis and result presentation. Defect data are stored in databases, test data in reports, and data related to source code measures are dynamically calculated. Utility programs are used to extract raw data from the data sources and convert them into a format suitable for analysis.
GQM Tool	In this initiative, integration between the main tool (GQM Tool) and the configuration management system (PCMS) is made through links created between the databases of the tools. Once links are established, data created by and stored in PCMS's database are automatically made available in GQM Tool's database. For integrating metric reader tools (Krakatau and Resource Standard Metrics) to GQM Tool, another strategy is used because the metric readers record measurement data in HTML files. A DTD (Document Type Definition) was defined and a translator to turn the native data format into XML was developed to each metric reader tool. GQM Tool was equipped with a parser that reads XML files and stores data into GQM Tool's database.
MetriFlame	MetriFlame integrates data from various data sources, such as version control systems and project management tools, with the help of a scheduler. The scheduler provides a set of tasks (e.g., retrieve data from external raw data sources, calculate metrics) to be chosen by the user and scheduled according to established conditions. When conditions are fulfilled, the scheduler activates MetriFlame to perform the scheduled tasks.
DSS	In DSS, data stored in three legacy data sources are integrated into a central data warehouse that is further used for a web front-end to analyze and display quality information about service and customer satisfaction. Details about how data is loaded into the data warehouse are not discussed in the publication.
SM in a CI Environment	In this proposal, the integration is done with the assistance of the CI engine Cruise Control.Net. It extracts data from other tools and consolidates them in XML files. A developed program extracts measures from the XML files, and stores them into a relational database. An ETL process consolidates data in the relational database into a data warehouse cube, enabling OLAP analysis over the cube.
SOFAS	In SOFAS there is a set of web services for software analysis that provide different kinds of analysis from data recorded in several tools. The integration approach is based on service workflows. Users can select services to compose workflows, and SOFAS turns the workflows into executable processes and runs them. The composition of services allows data to flow between services and generates a combined analysis result.
Dione	Dione integrates data by using smart clients that are small Java programs able to be executed directly from the Dione web interface. They gather measures from software artifacts (e.g., source code repositories and issue management systems), and send them to Dione's server. Integration with software quality applications is also supported through web services.
QualitySpy	QualitySpy uses four connectors to collect data from different sources. Three of them are used to collect raw data (data stored in a textual form without transformation) and one to calculate software measures from Java classes. Collected data are stored in a central repository and are made available to further investigation. The user can define measures on top of the raw data using a reporting module interface implemented as a light web client that communicates with the server using Representational State Transfer (REST) architecture.

Table 4. Tools Integration Overview (cont.)

Proposal	Description
QualitySpy	QualitySpy uses four connectors to collect data from different sources. Three of them are used to collect raw data (data stored in a textual form without transformation) and one to calculate software measures from Java classes. Collected data are stored in a central repository and are made available to further investigation. The user can define measures on top of the raw data using a reporting module interface implemented as a light web client that communicates with the server using Representational State Transfer (REST) architecture.
3C	In this proposal, the CI engine Cruise Control is responsible for the integration. It activates the measurement tools to collect test data (from JUnit and Cobertura) and source code data (from Findbugs, Checkstyle and PMD). Then, measurement results are put into graphs (Cockpit).
ASSIST	ASSIST is integrated with commercial project management, issue tracking and ERP software suites, which all rely on relational database. An integration strategy based on SQL (Structured Query Language) is used in order to spare code development/modification at commercial tools side. An SQL-like syntax and interpreter was developed so that complex and parameterized expressions can be written and users can define measurement constructs, queries, reports and data collection services via user interfaces.
DePress	DePress integrates with other tools through KNIME plugins structure. For each new tool to be integrated, a new plugin has to be implemented. To collect data, plugins can work in two modes: online and offline. Online mode means direct access to data through the API provided by the tools. Offline mode means that data is exported by tools and imported into DePress. DePress uses tabular format to exchange data. The plugins check only whether incoming data consists of the required columns or not.

Scope: Table 5 presents the study classification from the scope dimension perspective. The column Intra refers to proposals presenting an intra-enterprise scope, where only one organization is involved into the integration approach. Extra column refers to proposals involving more than one organization into the integration approach, presenting an extra-enterprise scope. Undefined column refers to proposals which it was not possible to identify whether only one or more organizations were involved in the integration initiative. 8 proposals are classified as intra-integration scope (TAME, Tool Support for SM, GQM Tool, MetriFlame, DSS, SM in a CI Environment, 3C, ASSIST)and 4 proposals are classified as undefined (SOFAS, Dione, QualitySpy, DePress).

Table 5. Integration Classification According to Scope Dimension

Proposal	Intra	Extra	Undefined
TAME	X		
Tool Support for SM	X		
GQM Tool	X		
MetriFlame	X		
DSS	X		
SM in CI Environment	X		
SOFAS			X
Dione			X
QualitySpy			X
3C	X		
ASSIST	X		
DePress			X

Viewpoint: The three views of this dimension are covered by all proposals. Programmer's view is covered by all proposals, since this view refers to the implementation of systems, and all initiatives are functional. Designer's view is addressed by all initiatives because it concerns design models representing the integration, and all initiatives have at least one design model describing the integration. In general, the design models are architectural models or conceptual models. Details about the models presented in each study are shown in Table 6. User's view refers to the integration from the users' perspective. All proposals provide features from the integration to the users.

Table 6. Integration Classification According to Designer's View (Viewpoint Dimension)

Proposal	Model Type	Model presented in the Publication
TAME	Architectural	Architectural design of the TAME system.
Tool Support for SM	Conceptual	An abstract model represents the information flow and connections among the integrated tools.
GQM Tool	Conceptual	Database schemas and a model addressing the integration of GQM tool with software configuration management and metrics tools.
MetriFlame	Conceptual	Models represent the integration environment and the scheduler.
DSS	Architectural	Design and architecture models of the integrated system.
SM in CI Environment	Conceptual	Data warehouse schema represents the integration scenario.
SOFAS	Architectural	SOFAS architecture model.
Dione	Architectural	Dione architecture model.
QualitySpy	Architectural	A model represents the QualitySpy high level architecture.
3C	Conceptual	A model addresses the message flow between tools.
ASSIST	Conceptual	A model represents the module responsible for data collection.
DePress	Conceptual	Models illustrate DePress integration with other tools and plugin workflow.

Layer: The classification regarding integration layer dimension is presented in Table 7. Process layer is not addressed, 7 proposals address only data layer (GQM Tool, MetriFlame, DSS, SM in a CI Environment, QualitySpy, ASSIST, DePress) and 4 proposals address only message layer (TAME, SOFAS, Dione, 3C). Tool Support for SM addresses both data and message layer. In this proposal, data is directly accessed in its source (data layer) and messages guide the information flow (message layer) from data capturing to analysis and presentation tools.

Level: Except for SOFAS, the higher level addressed by all proposals is the syntactical level. This level encompasses the way the data model and operation signatures are written down [Izza 2009]. Proposals classified in this level concern essentially in capturing data regardless the semantic of these data or of the services involved. SOFAS is the only proposal addressing the semantic level. It uses OWL ontologies to assign a clear semantic to data consumed and produced by the services.

Table 7. Integration Classification According to Layer Dimension

Proposal	Data	Message	Process
TAME		X	
Tool Support for SM	X	X	
GQM Tool	X		
MetriFlame	X		
DSS	X		
SM in CI Environment	X		
SOFAS		X	
Dione		X	
QualitySpy	X		
3C		X	
ASSIST	X		
DePress	X		

4.3. Discussion

This section provides some discussion about the data presented in the previous section.

Regarding measurement activities, measurement planning was supported by four studies while data collection and data analysis were supported by all of them. A possible explanation is that measurement planning is highly dependent on human judgment and not prone to automation [Komi-Sirvio, Parviainen and Ronkainen 2001].

We noticed that all proposals that support measurement planning activity (TAME, GQM Tool, MetriFlame, ASSIST) are based on GQM paradigm [Basili, Rombach and Caldiera 2004] or one of its variations. Since GQM has been successfully adopted in software measurement initiatives for years, its usage by the proposals that address measurement planning was expected.

Regarding data collection, data is collected in different ways in the proposals. There are smart client technologies (Dione), direct database access (GQM Tool, ASSIST), CI engines (SM in CI Environment, 3C), web services calls (SOFAS), plugins (DePress) and schedulers (MetriFlame). All proposals are focused in automated data collection. Proposals TAME, Dione and ASSIST also allow manual data input. Proposal MetriFlame argues that measurement process should be automated whenever possible and reasonable, turning data definition, collection, calculation and analysis easy and effortless as possible. In this way, the automation of measurement enhances visibility and leads to a greater awareness of the reasoning behind collecting measurement data and using it whiting organizations [Komi-Sirvio, Parviainen and Ronkainen 2001]. However, automating measurement process does not mean suppressing manual data collection. Therefore, we believe that a hybrid

approach (i.e., automated and manual) is preferred because it eases organizations to switch from a previous measurement process that is essentially based on manual data collection to an automated measurement process. It also allows the collection of measurement data that are not yet available in tools or are not prone to automated collection.

For data analysis, some studies focus on a specific perspective such as analysis of software reliability (Tool Support for SM), customer satisfaction (DSS) and defect prediction (Dione, DePress). However, most studies (TAME, GQM Tool, MetriFlame, SM in CI Environment, Dione, 3C) adopt a more general perspective, allowing to analyze whether the established goals have been achieved. It can be highlighted that all proposals addressing measurement planning adopt this general perspective, since they adopt GQM (a goal oriented paradigm).

Analyzing the integrated tools, there are proposals integrating commercial tools (e.g., Tool Support for SM, MetriFlame, ASSIST), open source tools (e.g., SM in a CI Environment, SOFAS, QualitySpy, 3C, DePress) and in-house developed tools (e.g., TAME, GQM Tool, ASSIST). Some proposals focus on integrating tools aiming to promote a more complete environment (with new measurement supporting features) in which external tools are used essentially to data collection (e.g., TAME, ASSIST). Others integrate tools without adding new functionalities, i.e., the initiative is basically the integration of existing (or developed) tools (e.g., Tool Support for SM, SM in a CI Environment, SOFAS, QualitySpy, 3C). 8 of 12 (75%) proposals adopt this last approach, including all initiatives that support measurement planning. We also noticed that in some initiatives (Tool Support for SM, GQM Tool, MetriFlame, SM in a CI Environment, SOFAS, ASSIST, DePress) supporting the measurement process was the main motivation for integrating the tools, while in others (TAME, DSS, Dione, QualitySpy, 3C) the measurement support was achieved as a consequence of the tool integration. For instance, in QualitySpy, tools are integrated to support monitoring the software development process and, as a consequence of the integration, software measurement was also supported.

There are a variety of tools that can be used to support measurement. This increases the relevance of integration in this domain, because organizations can choose the tools that are more suitable for their needs and work on their integration. Although there is diversity in tools being integrated, a predominance of code-related tools detaches. Several code measurement tools, issue tracking tools, and configuration management systems are integrated in the analyzed proposals. It might be a consequence of the fact that these types of tools are prone to automatic collection of measures. Nevertheless, some of them depend on others to provide information. For instance, since source code is usually stored in a configuration management system, the presence of a code measurement tool usually implies the presence of a configuration management system.

Considering that code-related tools were integrated in most proposals, it was expected that code measures (e.g., cyclomatic complexity, number of methods) would be addressed by most proposals. 10 of the 12 studies address them. Taking the types of integrated tools and measures into account, except MetriFlame and ASSIST, which have a more comprehensive scope, the integration initiatives usually address a

specific measurement scope (e.g., coding, customer support).

As a consequence of code-related tools predominance, Coding was the main measured process. Although other processes such as testing, configuration management and project management were also measured, coding was the process on which most of studies focused.

Analyzing the classification scheme according to Izza's framework [Izza 2009], some points can be highlighted. First, almost all studies have an intra-enterprise scope. Second, regarding the integration viewpoints, all of them were addressed by all the proposals, but in different ways. For programmer's view, every integration initiative is functional and it was implemented in an ad-hoc way, as described in RQ3.

Designer's view is covered, in general, by presenting either architectural or conceptual models. User's view is addressed by features available to the users due to the integration. Third, all proposals are classified in syntactical level, except one (SOFAS) that addresses the semantic level. Neglecting semantics during an integration initiative is a serious issue, since many semantic problems can occur, such as the ones called "false agreement", which are described in [Pokraev 2009] and include: the use of equivalent terms with different meaning; the use of equivalent terms with partially equivalent meaning; the use of different terms with equivalent meaning; and the use of different terms with a certain degree of equivalence. Ontologies can be used for addressing these problems, since they have been acknowledged as an important means for achieving semantic integration by providing formal specifications of shared conceptualizations [Nardi, Falbo and Almeida 2013]. Last, integration layer classification is diverse. 7 proposals address the data layer, 4 the message layer and 1 both the layers. Process layer is not addressed. We believe this is due to the fact that process layer integration (also referred as Business Process Integration) constitutes the most complex integration approach [Izza 2009]. It views an enterprise/organization as a set of interrelated business processes and not merely islands of information. Process integration deals with message flows, rules and process execution. Message layer is addressed, but only by few proposals. Message layer integration requires tool communication by means of message exchange between the tools. Tools providing service API (application programming interface) encourage message layer integration. However, if the integrated tools are not able to communicate by means of messages, integration in this layer demands extra effort, especially if tools were not developed by the group performing the integration (this is the case in most proposals). One alternative is to develop wrappers to expose tool features into services, allowing integration in this layer.

5. Related Work

Since secondary studies addressing tool integration to support SM were not found during the tertiary review, we searched for secondary studies analyzing tool integration without delimiting the domain. We found one study and, in this section, we compare some of its findings with the ones obtained in our SLR.

Nardi *et al*. (2013) conducted a secondary study investigating semantic integration initiatives and the use of ontologies in this context. The authors also

classified the identified integration initiatives according to Izza's framework [Izza 2009] layer dimension. When considering the layers in isolation or in tandem with other layers, 30% of the analyzed studies addresses the data layer, 75% addressed the message layer, and 42% addresses the process layer. In the SLR presented in this paper, when considering layers in isolation or in tandem with other layers, data layer is addressed by 66% of the studies, message layer by 42%, and process layer is not addressed. It is possible to notice that message layer is more addressed than data layer in [Nardi, Falbo and Almeida 2013] while the opposite occurs in the review discussed in this paper. We can speculate that one of the reasons of this difference is the fact that in [Nardi, Falbo and Almeida 2013] authors consider only tools integration initiatives that take semantic aspects into account. Since these initiatives cover more levels than most of the studies selected in our SLR, we can suppose that they are more complete and, as a consequence, prone to cover superior layers.

Nardi *et al.* (2013) explain that the role that functionalities (represented by the message layer) play in order to promote the link between data sources and business processes in addition to the increasing interest in service-oriented architectures (SOA) in the past decade justifies message layer being more addressed than data layer in their study. Analyzing the studies selected in our review, we noticed that, despite the consolidation of SOA, none of the studies have adopted a SOA approach.

6. Final Considerations

This paper presented a systematic literature review in which we analyzed twelve proposals involving tool integration for supporting software measurement. The review was motivated by issues we identified in a systematic mapping that preceded the review [Fonseca, Barcellos and Falbo 2015]. In the systematic literature review we investigated in deep some aspects related to measurement (addressed measures and supported measurement activities), tools (involved tools and provided support), and integration dimensions (according to Izza's framework [Izza 2009]).

Summarizing, the analyzed proposals address measurement execution (data collection and analysis), but most of them do not address measurement planning. Most proposals address code-related measures and focus on the Coding process. Coverage of integration scope, viewpoint and level dimensions is very similar among the proposals (most of them are intra-enterprise, cover all views and consider only syntactical aspects). Regarding the layer dimension, data integration is most common, although some proposals deal with integration in the message layer. Finally, only one proposal considers semantic aspects.

The results of the systematic review point out to some important issues in the context of tool integration to support measurement: (i) there is a limited support to measurement planning; (ii) the scope of measurement is not comprehensive (limited mainly to code-related measures and Coding process); (iii) semantics has not been a concern; (iv) service-oriented architectures have not been explored, resulting in limited integration in the message layer; (v) a holist view of the (software) process has not been considered, leading to the absence of integration in the process layer.

These issues reveal research opportunities. Tools integration initiatives in

software measurement domain should provide more general solutions, covering other measures than code-related measures and measuring other processes. Also, it is necessary to consider a holistic view of the measurement process and provide better support for measurement planning.

The integration between the measurement process and the measured processes (e.g., project management and quality assurance processes) should be addressed by performing integration at the process layer.

SOA is well recognized and widespread in nowadays. Thus, measurement related tools integration initiatives should consider this kind of architecture in order to enable integration in message layer, even when integrating legacy tools (i.e., not service-oriented tools).

Last but not least, it is necessary to consider semantic aspects when integrating tools to support software measurement. The vocabulary regarding software measurement is diverse. Although there are several standards devoted to address software measurement (e.g., [IEEE 1998], [ISO 2007], [McGarry *et al.* 2012]) the vocabulary used by them is not the same. Many times, the same concept is designated by different terms in different proposals. Others, the same term refers to different concepts. As a consequence, the vocabulary adopted by software organizations tends to also be diverse. To deal with these problems, it is important to establish a common conceptualization regarding the software measurement domain [Barcellos, Falbo and Rocha 2010]. Ontologies can be used for that purpose and can be used as a basis for integrating measurement tools, since they provide formal specifications of shared conceptualizations and have been acknowledged as an important means for achieving semantic integration [Nardi, Falbo and Almeida 2013].

In this sense, we consider that the road ahead in the area must focus on ontology-based approaches to guide tool integration initiatives, considering semantic aspects and covering both message and process layers.

7. References

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2010). Establishing a well-founded conceptualization about software measurement in high maturity levels. In Proc. of the *7th International Conference on the Quality of Information and Communications Technology*, p. 467–472.

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2013). A Strategy for Preparing Software Organizations for Statistical Process Control. *Journal of the Brazilian Computer Society* 2013 (19), p. 445–473.

Basili, V. R. and Rombach, H.D. (1988). The TAME project: towards improvement-oriented software environments. *IEEE Trans. Softw. Eng*., v. 14, n. 6, p. 758–773.

Basili, V. R., Rombach, H.D., Caldiera, G. (2004). Goal Question Metric paradigm. *Encyclopedia of Software Engineering*, 2 Volume Set, John Wiley & Sons, Inc.

Caglayan, B., Misirli, A. T., Calikli, G., Bener, A., Aytac, T. and Turhan B. (2012). Dione: an integrated measurement and defect prediction solution. In *Proc. of the*

ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12, p. 1–4.

Chulani, S., Ray, B., Santhanam, P. and Leszkowicz, R. (2013). Metrics for managing customer view of software quality. In Proc. of the *5th International Workshop on Enterprise Networking and Computing in Healthcare Industry* (IEEE Cat. No.03EX717), p. 189–198.

Fenton, N. E. and Neil, M. (2000). Software Metrics: a Roadmap. In *Proc. of the 22nd International Conference on Software Engineering*, San Francisco Bay, p. 359-370.

Fenton, N. E. and Pfleeger, S. L. (1997). Software Metrics: A Rigorous and Pratical Approach, *PWS Publishing Company*.

Florac, W. A. and Carleton, A. D. (1999). Measuring the software process: statistical process control for software process improvement. *Addison Wesley*, Boston, USA.

Fonseca, V. S., Barcellos, M. P. and Falbo, R. A. (2015). Integration of Software Measurement Supporting Tools: A Mapping Study. In *Twenty-Seventh International Conference on Software Engineering and Knowledge Engineering* (SEKE 2015), p. 516–521.

Ghezzi, G. and Gall H. C. (2011). SOFAS: A Lightweight Architecture for Software Analysis as a Service. In *Proc. of the Ninth Working IEEE/IFIP Conference on Software Architecture*, p. 93–102.

IEEE Std 1061. (1998). IEEE Standard for a Software Quality Metrics Methodology.

ISO/IEC 15939. (2007). Systems and Software Engineering – Measurement Process.

Izza S. (2009). Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterp. Inf. Syst.*, v. 3, n. 1, p. 1–57.

Janus, A., Dumke, R., Schmietendorf, A. and Jager, J. (2012). The 3C approach for agile quality assurance. In *Proc. of the 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)*, p. 9–13.

Jureczko, M. and Magott, J. (2012). QualitySpy: a framework for monitoring software development processes. In *Journal of Theoretical and Applied Computer Science*, v. 6, n. 1, p. 35–45.

Keser, B., Iyidogan, T. and Ozkan, B. (2013). ASSIST: an integrated measurement tool. In *Proc. of the Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, p. 237–242.

Kitchenham, B. A., Budgen, D. and Brereton, O. P. (2011). Using mapping studies as the basis for further research: a participant-observer case study. *Journal of Information and Software Technology*, v. 53, p. 638-651.

Kitchenham B. A. and Charters S. (2007). Guidelines for performing systematic literature reviews in software engineering. TR EBSE-2007-01. *School of Computer Science and Mathematics*, Keele University,

Komi-Sirvio, S., Parviainen, P. and Ronkainen, J. (2001). Measurement automation: methodological background and practical solutions a multiple case study. In *Proc. of the 7th International Software Metrics Symposium*, p. 306–316.

Lavazza, L. (2000). Providing automated support for the GQM measurement process. *IEEE Softw.*, v. 17, n. 3, p. 56–62.

Madeyski, L. and Majchrzak, M. (2014). Software measurement and defect prediction with DePress extensible framework. *Foundations of Computing and Decision Sciences*, v. 39, n. 4, p. 249–270.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. and HallF. (2002). Practical Software Measurement: Objective information for decision makers. *Addison Wesley*, Boston, USA.

Mohammed M., Mohammad A. (2015). UML model refactoring: a systematic literature review. In *Empirical Software Engineering*, v. 20, Issue 1, p. 206-251.

Moreira, G. de S. P., Mellado, R. P., Montini, D. A., Dias, L. A. V. and da Cunha, A. M. (2010). Software product measurement and analysis in a continuous integration environment. In *Proc. of the 7th International Conference on Information Technology: New Generations*, p. 1177–1182.

Nardi, J. C., Falbo, R. A. and Almeida, J. P. A. (2013). A panorama of the semantic EAI initiatives and the adoption of ontologies by these initiatives. In *Proc. of the IWEI 2013*, LNBIP 144, Lecture Notes in Business Information Processing, p. 198–211.

Park, R. E., Goethert, W. B. and Florac, W. A. (1996). Goal-Driven Software Measurement — A Guidebook. Carnegie Mellon University, Pittsburgh, PA, CMU/SEI-96-HB-002.

Pokraev, S. (2009). Model-Driven Semantic Integration of Service-Oriented Applications. PhD Thesis, University of Twente.

Solingen, R. and Berghout, E. (1999). The Goal/Question/Metric Method: a Practical Guide for Quality Improvement of Software Development. *McGraw-Hill*.

Themistocleous, M., Irani, Z., Love, P. E. D. (2004). Evaluating the integration of supply chain information systems: A case study. *European Journal of Operational Research*, n. 159, p. 393–405.

Tian, J., Troster, J. and Palma, J. (1997). Tool support for software measurement, analysis and improvement. *J. Syst. Softw.*, v. 39, n. 2, p. 165–178.

Vernadat, F. B. (2007). Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control*, v. 31, p. 137–145.

Wegner, P. (1996). Interoperability. In *ACM Computing Survey*, v. 28, n. 1, p. 285–287.