Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2012, Article ID 949746, 28 pages
doi:10.1155/2012/949746

Research Article

Specifying Process Views for a Measurement, Evaluation, and Improvement Strategy

Pablo Becker,[1] Philip Lew,[2] and Luis Olsina[1]

[1] *GIDIS_Web, Engineering School, Universidad Nacional de La Pampa, General Pico, Argentina*
[2] *School of Software, Beihang University, Beijing, China*

Correspondence should be addressed to Luis Olsina, olsinal@ing.unlpam.edu.ar

Received 24 August 2011; Revised 12 November 2011; Accepted 8 December 2011

Academic Editor: Osamu Mizuno

Any organization that develops software strives to improve the quality of its products. To do this first requires an understanding of the quality of the current product version. Then, by iteratively making changes, the software can be improved with subsequent versions. But this must be done in a systematic and methodical way, and, for this purpose, we have developed a specific strategy called SIQinU (*Strategy for understanding and Improving Quality in Use*). SIQinU recognizes problems of quality in use through evaluation of a real system-in-use situation and proposes product improvements by understanding and making changes to the product's attributes. Then, reevaluating quality in use of the new version, improvement gains can be gauged along with the changes that led to those improvements. SIQinU aligns with GOCAME (*Goal-Oriented Context-Aware Measurement and Evaluation*), a multipurpose generic strategy previously developed for measurement and evaluation, which utilizes a conceptual framework (with ontological base), a process, and methods and tools. Since defining SIQinU relies on numerous phase and activity definitions, in this paper, we model different process views, for example, taking into account activities, interdependencies, artifacts, and roles, while illustrating them with excerpts from a real-case study.

1. Introduction

Even though software product launches now may consist of "continuous beta," users expect more and better functionality, combined with increased quality from the user's perception. Methodically improving the perceived quality, that is, its quality in use (QinU) particularly for web applications (WebApps), is not an easy job. WebApps—a kind of software applications—are no longer simple websites conveying information. Rather, they have become fully functional software applications often with complex business logic and sometimes critical to operating the business. Users, in addition, are becoming more demanding and diverse in their requirements. Consequently, WebApp quality and especially the quality in use, namely, the perceived quality by the end user has taken on increased significance as web and now cloud deployment have become mainstream delivery methods. Systematic means for evaluating QinU is important because it enables understanding the quality satisfaction level achieved by the application and provides useful information for recommendation and improvement processes in a consistent manner over time. Coincident with consistent and systematic evaluation of WebApp quality, the main goal is to ultimately improve its QinU.

This leads to our strategy with the objectives of understanding and improving the QinU—as nonfunctional requirements—of WebApps. QinU is currently redefined in the ISO 25010 standard [1], which was reused and enlarged by the 2Q2U (*internal/external Quality, Quality in use, actual Usability*, *and User experience*) quality framework—see [2] for an in-depth discussion. QinU from the actual usability standpoint (that embraces performance or "do" goals in contrast to hedonic or "be" goals [3]) is defined as the degree to which specified users can achieve specified goals with effectiveness in use, efficiency in use, learnability in use, and accessibility in use in a specified context of use [2].

Utilizing 2Q2U quality models, we developed SIQinU as an integrated means to evaluate and find possible problems in QinU which are then related to external quality (EQ) characteristics and attributes (by doing a mapping between

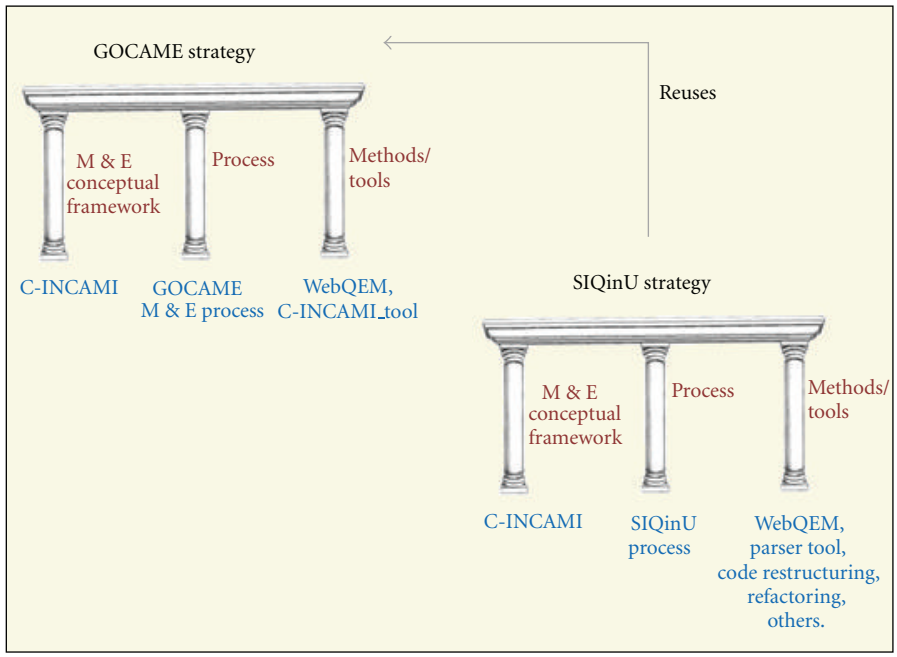


Figure 1: Allegory of the three GOCAME pillars, which are reused to a great extent by SIQinU.

QinU problems and EQ). This is followed by evaluating the application from the EQ standpoint and then making recommendations for improvements if necessary. The new version, based on recommended improvements, is reevaluated to gauge the improvement gain from both the EQ and QinU point of views. One aspect of SIQinU's uniqueness is that it collects user usage data from WebApps in a real context of use whereby code snippets are inserted (or using similar techniques) to gather data related to the task being executed by users at the subtask level enabling nonintrusive evaluations.

It is worth mentioning that SIQinU aligns with the GOCAME strategy [4]. GOCAME, a multipurpose goal-oriented strategy, was previously developed for supporting measurement and evaluation (M&E) programs and projects. Its rationale is based on three main pillars or principles, namely, (i) a conceptual framework utilizing an ontological base; (ii) a well-defined measurement and evaluation process; (iii) quality evaluation methods and tools instantiated from both the framework and process. This is allegorically depicted in Figure 1.

GOCAME's first principle is that designing and implementing a robust M&E program require a sound conceptual framework. Often times, organizations conduct start and stop measurement programs because they do not pay enough attention to the way nonfunctional requirements, contextual properties, metrics, and indicators should be designed, implemented, and analyzed. Any M&E effort requires an M&E framework built on a sound conceptual base, that is, on an ontological base, which explicitly and formally specifies the main agreed concepts, properties, relationships, and constraints for a given domain. To accomplish this, we utilize the C-INCAMI (*Contextual-Information Need, Concept model, Attribute, Metric, and Indicator*) framework and its components [4, 5] based on our metrics and indicators ontology.

GOCAME's second principle requires a well-established M&E process in order to guarantee repeatability in performing activities and consistency of results. A process prescribes a set of phases, activities, inputs and outputs, interdependencies, sequences and parallelisms, check points, and so forth. Frequently, process specifications state what to do but do not mention the particular methods and tools to perform specific activity descriptions. Thus, to provide repeatability and replicability in performing activities, a process model for GOCAME was proposed in [6], which is also compliant with both the C-INCAMI conceptual base and components. Finally, methods and tools—the third pillar in the GOCAME strategy—can be instantiated from both the conceptual framework and process, for example, the WebQEM (*Web Quality Evaluation*) methodology [7] and its tool called C-INCAMI_tool [4].

SIQinU utilizes the above three GOCAME principles while also reusing the C-INCAMI conceptual base and process. However, since SIQinU is a specific-purpose goal-oriented strategy, it has specific processes, methods, and procedures that are not specified in GOCAME. Since the process aspect is critical in specifying SIQinU, given of its numerous interrelated phases and activities, this work defines its process model in detail through illustration with excerpts of a real case study. This case study was thoroughly illustrated in [8], and also aspects of its internal and external validity were considered in [9] as well.

Note that processes can be modeled taking into account different views [10] such as (i) functional that includes the activities' structure, inputs, and outputs; (ii) informational

that includes the structure and interrelationships among artifacts produced or consumed by the activities; (iii) behavioral that models the dynamic view of processes; (iv) organizational that deals with agents, roles, and responsibilities. Additionally, a methodological view is described in [11], which is used to represent the process constructors (e.g., specific methods) that can be applied to different descriptions of activities. In order to specify all these views, different modeling languages can be used. However, no modeling language fits all needs and preferences. Each has its own strengths and weaknesses, which can make it more suitable for modeling certain views than others [12].

This paper using UML 2.0 activity diagrams [13] and the SPEM profile [14] stresses the functional, informational, organizational, and behavioral views for the SIQinU process. Modeling its process helps to (i) ease the repeatability and understandability among practitioners, (ii) integrate and formalize different activities that are interrelated in different phases, and (iii) promote the learnability and interoperability by reusing the same ontological base coming from the C-INCAMI framework. This paper is an extension of the work presented in [15] elaborating on new aspects and views (e.g., informational and organizational) for both GOCAME and SIQinU process, as we remark later on. Summarizing, the main contributions of this paper are

(i) a six-phased strategy (SIQinU) useful for understanding and improving the QinU for WebApps, which is specified and illustrated from the process viewpoint regarding activities (i.e., the functional view), interdependencies (behavioral view), artifacts (informational view), and roles (organizational view).

(ii) foundations for reusing a multipurpose goal-oriented strategy (i.e., GOCAME) to derive and integrate more specific-purpose strategies (e.g., SIQinU) regarding its conceptual M&E framework, methods, and process views.

The remainder of this paper is organized as follows. Section 2 gives an overview of the six-phased SIQinU strategy. Section 3 provides the GOCAME rationale considering its three pillars, which are to a great extent reused by SIQinU; particularly, in Section 3.4, we discuss why SIQinU is in alignment with GOCAME regarding its conceptual M&E framework (Section 3.1), its process views (Section 3.2), and its methods (Section 3.3). Section 4 models and illustrates the six-phased SIQinU process from the above-mentioned process views. Section 5 analyzes related work considering the two quoted contributions, and, finally, in Section 6, concluding remarks as well as future work are discussed.

2. Overview of the SIQinU Strategy

SIQinU is an evaluation-driven strategy to iteratively and incrementally improve a WebApp's QinU by means of mapping actual system-in-use problems—that happened while real users were performing common WebApp tasks—to measurable EQ product attributes and by then improving the current WebApp and assessing the gain both at EQ and QinU levels. SIQinU can be implemented in an economic and systematic manner that alleviates most of the problems identified with typical usability testing studies which can be expensive, subjective, nonrepeatable, time consuming, and unreliable due to users being observed in an intrusive way. This is accomplished through utilizing server-side capabilities to collect user usage data from log files adding, for example, snippets of code in the application to specifically record data used to calculate measures and indicator values for QinU in a nonintrusive way.

Note that SIQinU can apply to systems in use other than WebApps if data can be collected for analysis regarding user activities. This may be possible in client-server network environments where the developer has control over the server code and the activities of users at their client workstations can be collected. Therefore, the major constraint is in collecting easily large amounts of data in a nonintrusive manner from which to measure and evaluate the QinU serving as the basis for improvement.

The SIQinU strategy uses quality models such as those specified in the ISO 25010 standard [1] and its enhancement, that is, the 2Q2U quality framework [2]. Once the QinU model has been established, the data collected, and metrics and indicators calculated, a preliminary analysis is made. If the agreed QinU level is not met, then EQ requirements are derived considering the application's QinU problems and its tasks, subtasks, and associated screens. In turn, taking into account the derived EQ requirements, an evaluation of the WebApp attributes is performed by inspection. Thus a product analysis regarding the EQ evaluation is performed, and changes for improvement are recommended. If the improvement actions have been implemented, then the new version is reevaluated to gauge the improvement gain both from the EQ and the QinU standpoint. Ultimately, SIQinU is a useful strategy not only for understanding but also—and most importantly—for improvement purposes.

SIQinU uses the concepts for nonfunctional requirements specification, measurement, and evaluation design, and so forth, established in the C-INCAMI framework as we will see in Section 3.1. Also, SIQinU has an integrated, well-defined, and repeatable M&E process, which follows to great extent the GOCAME process as we will discuss in Sections 3.2 and 3.3. Specifically, the SIQinU process embraces six phases as shown in Figure 2, which stresses the main phases and interdependencies.

Additionally, Table 1 provides, with Phase (Ph.) reference numbers as per Figure 2, a brief description of each phase, the involved activities, and main artifacts. Section 4 thoroughly illustrates phases, activities, interdependencies, artifacts, as well as roles taking into account aspects of the functional, behavioral, informational, and organizational views.

Lastly, in the Introduction, we stated as contribution that GOCAME—a previously developed strategy—can be reused to derive and integrate more specific-purpose strategies (as is the case of SIQinU) regarding its conceptual M&E framework, process, and methods. Therefore, in the following section, the GOCAME strategy regarding these principles is outlined.

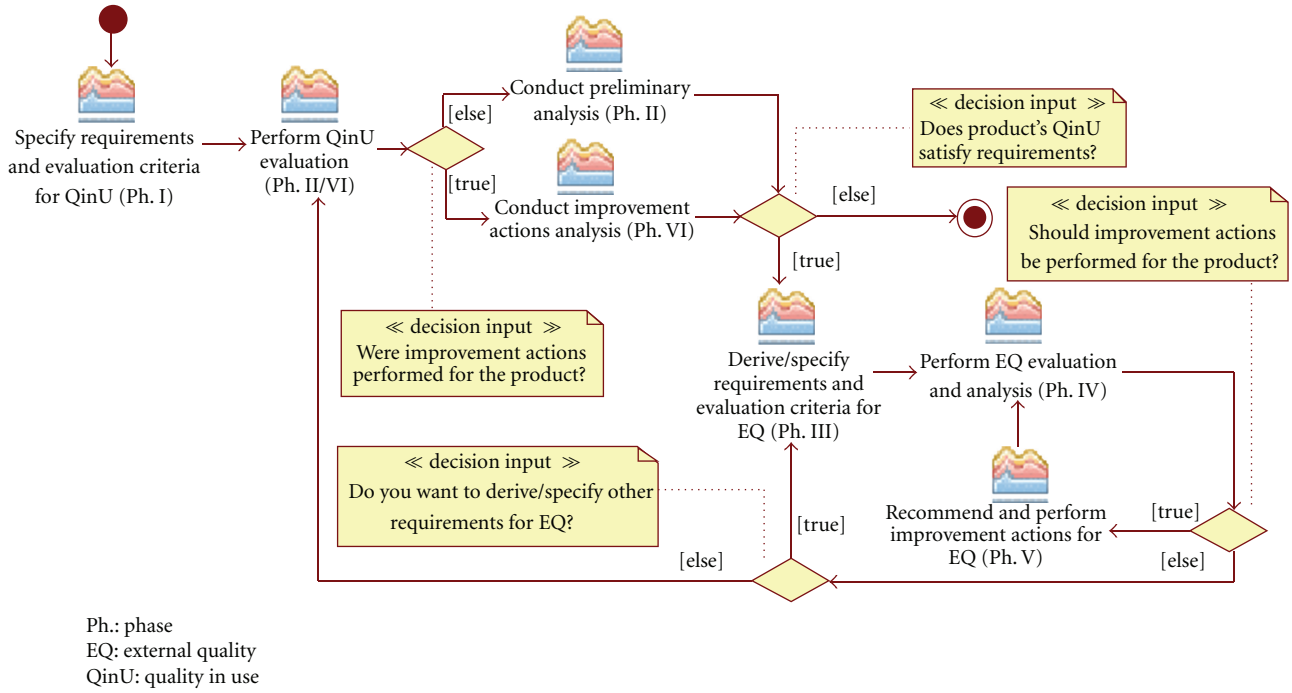


Figure 2: Process overview of SIQinU stressing phases and interdependencies.

3. GOCAME Strategy

GOCAME is a multipurpose M&E strategy that follows a goal-oriented and context-sensitive approach in defining projects. It allows the definition of M&E projects including well-specified context descriptions, providing therefore more robust evaluation interpretations among different project results at intra- and interorganization levels.

GOCAME is based on the three above-mentioned pillars, namely, a conceptual framework (described in Section 3.1); a M&E process (Section 3.2); methods and tools (Section 3.3). Finally, in Section 3.4, we discuss why SIQinU is in alignment with GOCAME regarding these capabilities.

3.1. C-INCAMI Conceptual Framework. The C-INCAMI framework provides a domain (ontological) model defining all the concepts and relationships needed to design and implement M&E processes. It is an approach in which the requirements specification, M&E design, and analysis of results are designed to satisfy a specific information need in a given context. In C-INCAMI, concepts and relationships are meant to be used along all the M&E activities. This way, a common understanding of data and metadata is shared among the organization's projects lending to more consistent analysis and results across projects.

Following the main activities of the process (shown in Section 3.2), the framework—that is, the related concepts and relationships—is structured in six components or modules, namely,

(i) *measurement and evaluation project definition*;

(ii) *nonfunctional requirements specification*;

(iii) *context specification*;

(iv) *measurement design and implementation*;

(v) *evaluation design and implementation*;

(vi) *analysis and recommendation specification*.

For illustration purposes, Figure 3 shows the main concepts and relationships for four components (i.e., from (ii) to (v)), and Table 2 defines the used terms, stressed in *italic* in the following text). The entire modeling of components can be found in [4, 5].

Briefly outlined, the GOCAME strategy follows a goal-oriented approach in which all the activities are guided by agreed *Information Needs*; these are intended to satisfy particular nonfunctional requirements of some *Entity* for a particular purpose and stakeholder's viewpoint. The nonfunctional requirements are represented by *Concept Models* including high-level *Calculable Concepts*, as in ISO 25010's quality models [1], which, in turn, measurable *Attributes* of the entity under analysis are combined. The instantiated quality models are the backbone for measurement and evaluation. *Measurement* is specified and implemented by using *Metrics*, which define how to represent and collect attributes' values; *Evaluation* is specified and implemented by using *Indicators*, which define how to interpret attributes' values and calculate higher-level calculable concepts of the quality model.

Since each *MEProject* does not occur in isolation, we therefore say that measurement and evaluation should be supported by *Context*; thus, context specifications may be

Table 1: SIQinU phases, activities, and artifacts.

Phases (Ph.)	Phase description and activities involved	Artifacts (work products)
Ph. I Specify requirements and evaluation criteria for QinU	Taking into account the recorded data of the WebApp's usage, we reengineer QinU requirements. This embraces designing tasks, defining user type, specifying usage context and characteristics. Activities include (see Figure 8) (i) establish information need; (ii) specify project context; (iii) design tasks; (iv) select QinU concept model; (v) design QinU measurement and evaluation; (vi) design Preliminary Analysis	(1) Information Need specification (2) Context specification (3) Task/subtasks specification (4) QinU NFR tree (5) QinU metrics and indicators specification (6) Analysis design
Ph. II Perform QinU evaluation and conduct preliminary analysis	As per Ph. I, data is collected purposely targeting QinU attributes for improvement. Depending on the WebApp's data collection capabilities, we collect data such as the date/time, the data is gathered, errors, task, and subtask completion and accuracy, and so forth. It includes (see Figure 12) (i) collect and parse data pertaining to tasks with their subtasks; (ii) quantify QinU attributes; (iii) calculate QinU indicators; (iv) conduct preliminary analysis	(1) Parsed data file (2) Measure and indicator values for QinU (3) QinU preliminary analysis report
Ph. III Derive/Specify Requirements and Evaluation Criteria for EQ	Based on Ph. I and II, we derive EQ requirements, that is, characteristics and attributes, with their metrics and indicators in order to understand the current WebApp's quality. Activities include (see Figure 13) (i) select EQ concept model; (ii) design EQ measurement; (iii) design EQ evaluation	(1) EQ NFR tree (2) EQ metrics and indicators specification
Ph. IV Perform EQ evaluation and analysis	Activities include (see Figure 15) (i) quantify EQ attributes; (ii) calculate EQ indicators; (iii) conduct an EQ analysis and identify parts of the WebApp that need improvement	(1) Measure and indicator values for EQ (2) EQ analysis report (and new report after reevaluation)
Ph. V Recommend, perform improvement actions, and reevaluate EQ	Using the EQ attributes that require improvement, we make improvement recommendations for modifying the WebApp, that is, version 1 to 1.1. Activities include (see Table 9) (i) recommend improvement actions; (ii) design improvement actions; (iii) perform improvement actions; (iv) evaluate improvement gain to note improvement from benchmark in Ph. IV. Note that once changes were made on the WebApp (Phase V), evaluators could detect that other EQ attributes (from problems identified in QinU) should be derived—under the premise that if further EQ improvement in these new attributes will result in greater impact on the improvement gain in QinU. This concern is taken into account in the process as shown in Figure 2	(1) EQ recommendations report (2) Improvement plan (3) New application version
Ph. VI Reevaluate QinU and analyze improvement actions	Once the new version has been used by real users, we evaluate QinU again to determine the influence of what was improved for the WebApp's EQ on QinU. This provides insight to further develop the *depends-on* and *influences* relationships [8]. Activities include (i) evaluate QinU again to determine level of improvement from Ph. II; (ii) conduct improvement action analysis, which includes developing *depends-on* and *influences* relationships between EQ improvements and QinU	(1) New measure and indicator values for QinU (2) QinU improvement analysis report (3) EQ/QinU attribute relationship table (see Table 11)

provided in order to support sounder analysis, interpretations, and recommendations. A summarized description for each component is provided below.

3.1.1. M&E Project Definition Component. This component defines and relates a set of *Project* concepts needed to articulate M&E activities, roles, and artifacts.

A clear separation of concerns among *Nonfunctional Requirements Project*, *Measurement Project,* and *Evaluation Project* concepts is made for reuse purposes as well as for easing management's role. The main concept in this component is a measurement and evaluation project (*MEProject*), which allows defining a concrete requirement project with the information need and the rest of the nonfunctional requirements information. From this requirement project, one or more measurement projects can be defined and associated; in turn, for each measurement project, one or more evaluation projects could be defined. Hence, for each measurement and evaluation project we can manage associated subprojects accordingly. Each project also has information such as responsible person's name and contact

Table 2: Some M&E terms—see [4] for more details.

Concept	Definition
Project terms	
Evaluation project	A project that allows, starting from a measurement project and a concept model of a nonfunctional requirement project, assigning indicators, and performing the calculation in an evaluation process.
Measurement project	A project that allows, starting from a nonfunctional requirements project, assigning metrics to attributes, and recording the values in a measurement process.
MEProject (i.e., measurement and evaluation project)	A project that integrates related nonfunctional requirements, measurement and evaluation projects, and then allows managing and keeping track of all related metadata and data.
Project	Planned temporal effort, which embraces the specification of activities and resources constraints performed to reach a particular goal.
Nonfunctional requirements project	A project that allows specifying nonfunctional requirements for measurement and evaluation activities.
Nonfunctional requirements terms	
Attribute (synonyms: property, feature)	A measurable physical or abstract property of an entity category.
Calculable concept (synonym: characteristic, dimension)	Abstract relationship between attributes of entities and information needs.
Concept model (synonyms: factor, feature model)	The set of subconcepts and the relationships between them, which provide the basis for specifying the concept requirement and its further evaluation or estimation.
Entity	A concrete object that belongs to an entity category.
Entity category (synonym: object)	Object category that is to be characterized by measuring its attributes.
Information need	Insight necessary to manage objectives, goals, risks, and problems.
Requirement tree	A requirement tree is a constraint to the kind of relationships among the elements of the concept model, regarding the graph theory.
Context terms	
Context	A special kind of entity representing the state of the situation of an entity, which is relevant for a particular information need. The situation of an entity involves the task, the purpose of that task, and the interaction of the entity with other entities as for that task and purpose.
Context property (synonyms: context attribute, feature)	An attribute that describes the context of a given entity; it is associated to one of the entities that participates in the described context.
Measurement terms	
Calculation method	A particular logical sequence of operations specified for allowing the realization of a formula or indicator description by a calculation.
Direct metric (synonyms: base, single metric)	A metric of an attribute that does not depend upon a metric of any other attribute.
Indirect metric (synonyms: derived, hybrid metric)	A metric of an attribute that is derived from metrics of one or more other attributes.
Measure	The number or category assigned to an attribute of an entity by making a measurement.
Measurement	An activity that uses a metric definition in order to produce a measure's value.
Measurement method (synonyms: counting rule, protocol)	The particular logical sequence of operations and possible heuristics specified for allowing the realization of a direct metric description by a measurement.
Metric	The defined measurement or calculation method and the measurement scale.
Scale	A set of values with defined properties. *Note*. The scale type depends on the nature of the relationship between values of the scale. The scale types mostly used in software engineering are classified into nominal, ordinal, interval, ratio, and absolute.
Unit	A particular quantity defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity.

Table 2: Continued.

Concept	Definition
Evaluation terms	
Decision criterion (synonym: acceptability level)	Thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result.
Elementary indicator (synonyms: elementary preference, criterion)	An indicator that does not depend upon other indicators to evaluate or estimate a calculable concept.
Elementary model (synonym: elementary criterion function)	Algorithm or function with associated decision criteria that model an elementary indicator.
Evaluation (synonym: calculation)	Activity that uses an indicator definition in order to produce an indicator's value.
Global indicator (synonyms: global preference, criterion)	An indicator that is derived from other indicators to evaluate or estimate a calculable concept.
Global model (synonyms: scoring, aggregation model, or function)	Algorithm or function with associated decision criteria that model a global indicator.
Indicator (synonym: criterion)	The defined calculation method and scale in addition to the model and decision criteria in order to provide an estimate or evaluation of a calculable concept with respect to defined information needs.
Indicator value (synonym: preference value)	The number or category assigned to a calculable concept by making an evaluation.

information, starting and ending date, amongst other relevant information. Ultimately, this separation of concerns for each MEProject facilitates the traceability and consistency for intra- and interproject analysis.

3.1.2. Nonfunctional Requirements Specification Component. This component includes concepts and relationships needed to define the nonfunctional requirements for measurement and evaluation. One key concept is the *Information Need*, which specifies (see Figure 3)

(i) the purpose for performing the evaluation (which can be for instance "understand," "predict," "improve," "control," etc.);

(ii) the focus concept (*CalculableConcept*) to be assessed (e.g., "operability," "quality in use," "actual usability," etc.);

(iii) the category of the entity (*EntityCategory*) that will be assessed, for example, a "Web application" (which its superCategory is a "product" or "information system") and the concrete *Entities* (such as "JIRA," "Mantis" WebApps, etc.). Other super categories for entities can be "resource," "process," "information system-in-use" (e.g., as a Web application-in-use), and "project"

(iv) the userViewpoint (i.e., the intended stakeholder as "developer," "final user," etc.) from which the focus concept (and model) will be evaluated;

(v) the *Context* that characterizes the situation defined by the previous items to a particular MEProject.

The focus concept constitutes the higher-level concept of the nonfunctional requirements; in turn, a calculable concept and its subconcepts are related by means of a *Concept Model*. This model may be a tree-structured representation in terms of related mid-level calculable concepts and lower-level measurable *Attributes*, which are associated to the target entity. Predefined instances of metadata for information needs, entities, and entity categories, calculable concepts, attributes, and so forth, and its corresponding data can be obtained from an organizational repository to support reusability and consistency in the requirements specification along the organizational projects.

3.1.3. Context Specification Component. This component includes concepts and relationships dealing with the context information specification. The main concept is *Context*, which represents the relevant state of the situation of the entity to be assessed with regard to the stated information need. We consider Context as a special kind of *Entity* in which related relevant entities are involved. Consequently, the context can be quantified through its related entities. By relevant entities, we mean those that could affect how the focus concept of the assessed entity is interpreted (examples of relevant entities of the context may include resources as a network infrastructure, a working team, lifecycle types, the organization, or the project itself, among others).

In order to describe the situation, attributes of the relevant entities (involved in the context) are used. These are also Attributes called *Context Properties* and can be quantified to describe the relevant context of the entity under analysis. A context property inherits the metadata from the Attribute class such as name, definition, and objective, and also adds other information (see Figure 3). All these context properties' metadata are meant to be stored in the organizational repository, and, for each MEProject, the particular metadata and its values are stored as well. A detailed illustration of context and the relationship with other C-INCAMI components can be found in [5].

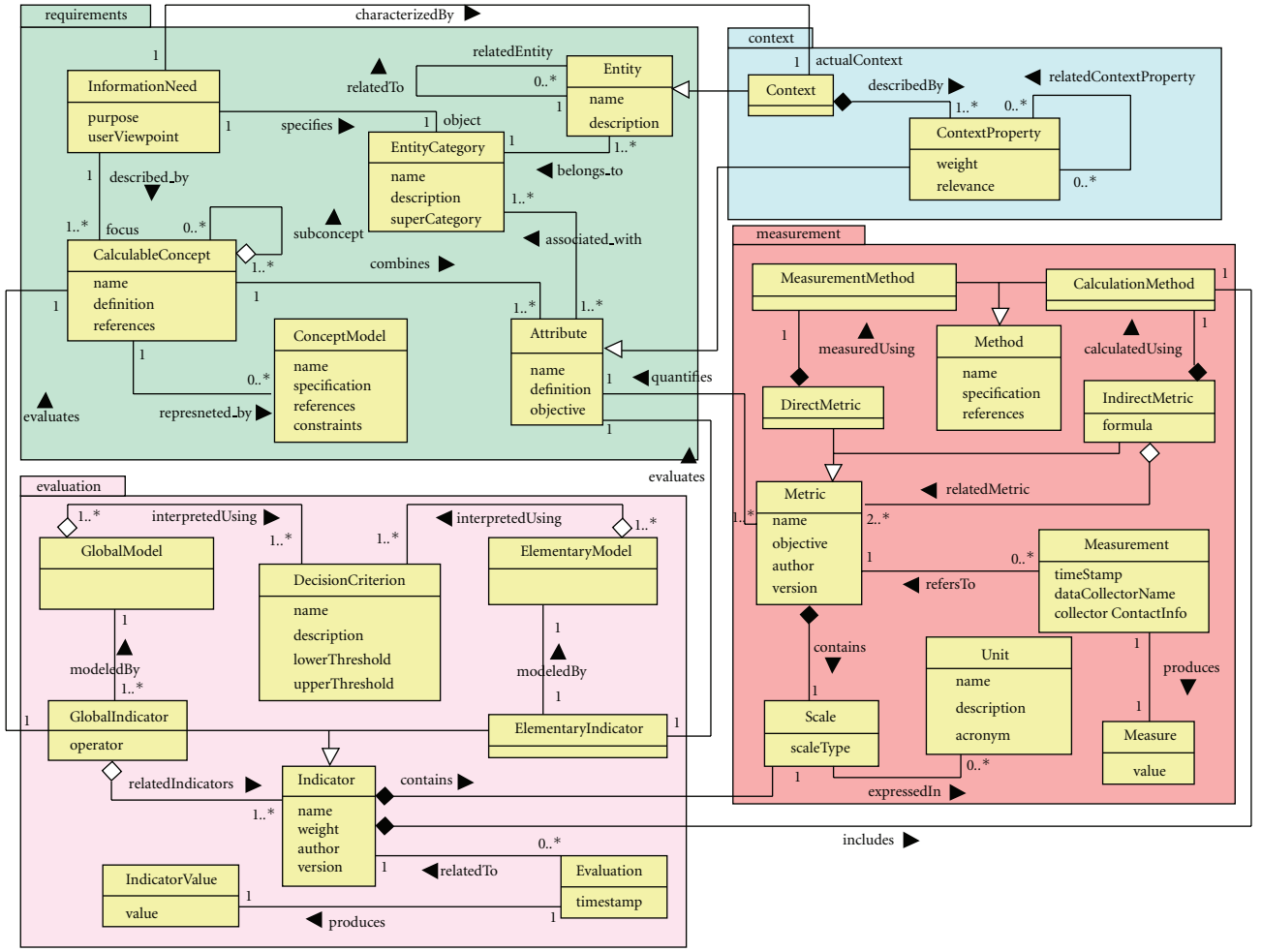


Figure 3: Main concepts and relationships of the C-INCAMI framework. Four out of six C-INCAMI components are depicted as packages, namely, nonfunctional requirements specification, context specification, measurement design and implementation, and evaluation design and implementation.

3.1.4. Measurement Design and Implementation Component. This module includes the concepts and relationships intended to specify the measurement design and implementation, for instance, the concrete Entities that will be measured, the selected *Metric* for each attribute, and so on.

Regarding measurement design, a metric provides a *Measurement* specification of how to quantify a particular attribute of an entity, using a particular *Method*, and how to represent its values, using a particular *Scale*. The properties of the measured values in the scale with regard to the allowed mathematical and statistical operations and analysis are given by the *scaleType* [16]. Two types of metrics are distinguished. *Direct Metrics* are those for which values are obtained directly from measuring the corresponding entity's attribute, by using a *Measurement Method*. On the other hand, *Indirect Metrics*' values are calculated from others direct metrics' values following a function specification and a particular *Calculation Method*.

For measurement implementation, a Measurement specifies the activity by using a particular metric description in order to produce a *Measure* value. Other associated metadata is the data collector name and the timestamp in which the measurement was performed.

3.1.5. Evaluation Design and Implementation Component. This component includes the concepts and relationships intended to specify the evaluation design and implementation. Indicator is the main term, which allows specifying how to calculate and interpret the attributes and calculable concepts of nonfunctional requirement models.

Two types of indicators are distinguished. First, *Elementary Indicators* that evaluate lower-level requirements, namely, attributes combined in a concept model. Each elementary indicator has an *Elementary Model* that provides a mapping function from the metric's measures (the domain) to the indicator's scale (the range). The new scale is interpreted using agreed *Decision Criteria*, which help analyze the level of satisfaction reached by each elementary nonfunctional requirement, that is, by each attribute. Second, *Partial/Global Indicators*, which evaluate mid-level

and higher-level requirements, that is, subcharacteristics and characteristics in a concept model. Different aggregation models (*GlobalModel*), like logic scoring of preference models, neuronal networks models, and fuzzy logic models, can be used to perform evaluations. The global indicator's value ultimately represents the global degree of satisfaction in meeting the stated information need for a given purpose and user viewpoint.

As for the implementation, an *Evaluation* represents the activity involving a single calculation, following a particular indicator specification—either elementary or global—producing an *Indicator Value*.

It is worthy to mention that the selected metrics are useful for a measurement process as long as the selected indicators are useful for an evaluation process in order to interpret the stated information need.

3.1.6. Analysis and Recommendation Specification Component. This component includes concepts and relationships dealing with analysis design and implementation as well as conclusion and recommendation. Analysis and recommendation component uses information coming from each MEProject (which includes requirements, context, measurement, and evaluation data and metadata). By storing all this information and by using different kinds of statistical techniques and visualization tools, stakeholders can analyze the assessed entities' strengths and weaknesses with regard to established information needs, and justify recommendations in a consistent way. Note this component is not shown in Figure 3. However, it is shown in Table 5 from the process specification standpoint.

3.2. GOCAME Measurement and Evaluation Process. When modeling a process, often engineers think more about what a process must do rather than how activities should be performed. In order to foster repeatability and reproducibility, a process specifies (i.e., prescribes or informs) a set of phases and activities, inputs and outputs, interdependencies, among other concerns. Also, to deal with the inherent complexity of processes, process views—also quoted in process modeling literature as perspectives—are used. A view is a particular model or approach to represent, specify, and communicate regarding the process. For instance, according to [10], a process can be modeled taking into account four views, namely, functional, behavioral, informational, and organizational.

Considering these process views, the functional perspective for GOCAME represents what activities and tasks (instead of the often-used term "task" in process modeling, which represents a fine grained or atomic activity, we will use the term "sub-activity" in the rest of the text, since, in Section 4, for QinU modeling, the term task has a very specific meaning) should be specified, what hierarchical activities structure (also known as task breakdown structure) there exists, what conditions (pre- and postconditions) should be accomplished, and what inputs and outputs (artifacts) will be required. Taking into account the terminology and components used in the C-INCAMI framework (Section 3.1), the integrated process of GOCAME embraces the following core activities: (i) *Define Non-Functional Requirements*; (ii) *Design the Measurement*; (iii) *Design the Evaluation*; (iv) *Implement the Measurement*; (v) *Implement the Evaluation*; (vi) *Analyze and Recommend* as shown in Figure 4. In addition, in Table 3, we enumerate these six activities, their involved subactivities, and the main output artifacts.

The behavioral view represents the dynamics of the process, that is, the sequencing and synchronization of activities, parallelisms, iterations, feedback loops, beginning and ending conditions, among other issues. The core GOCAME activities as well as sequences, parallelisms, main inputs, and outputs are depicted in Figure 4. The ≪*datastore*≫ stereotype shown in the figure represents repositories; for instance, the *Metrics* repository stores the metadata for the previously designed metrics. More details for the GOCAME functional and behavioral process views can be found in [6].

On the other hand, the informational view is concerned with those artifacts produced or required (consumed) by activities, the artifact breakdown structure, strategies of configuration management, and traceability models. For example, for illustration purpose, in Figure 5, the structure for the *Non-Functional Requirements Specification,* and *Metrics Specification* documents, which are outputs of *A.1* and *A.2* activities (see Table 3) is modeled. As the reader can observe in Figure 5(a), the *Non-Functional Requirements Specification* artifact is composed of the *Information Need Specification*, the *Context Specification* and the *Non-Functional Requirements Tree* documents. Besides, the *Metrics Specification* artifact (Figure 5(b)) is composed of a set of one or more *Metric Specification*, which in turn is composed of a *Scale* and a *Calculation* or *Measurement Method* descriptions. Note that, aimed at easing the communication among stakeholders these models can complement the textual specification made in the third column of Table 3.

Finally, the organizational view deals with what agents and their associated resources participate-plan-execute-control what activities; which roles (in terms of responsibilities and skills) are assigned to agents; what groups' dynamic and communication strategies are used, among other aspects. To illustrate this, Figure 6 depicts the different roles and their associated GOCAME activities. In Table 4, each role definition and its involved activities are also listed. Note that we have used italics in the definition column (in Table 4) to show the terminological correspondence between the process role definition and the C-INCAMI conceptual framework. It is important to remark that a role can be assumed by a human or an automated agent. And a human agent can be embodied by one or more persons, that is, a team.

In order to combine the above views, Table 5 presents a template which specifies just the *Analyze and Recommend* activity. The template specifies the activity name, objective and description, the subactivities and involved roles, input and output artifacts, pre- and postconditions. Also a diagram representing the *Analyze and Recommend* activity is attached as well to enable understanding and communicability.

Summarizing, the GOCAME M&E process can be described as follows. Once the *nonfunctional requirements*

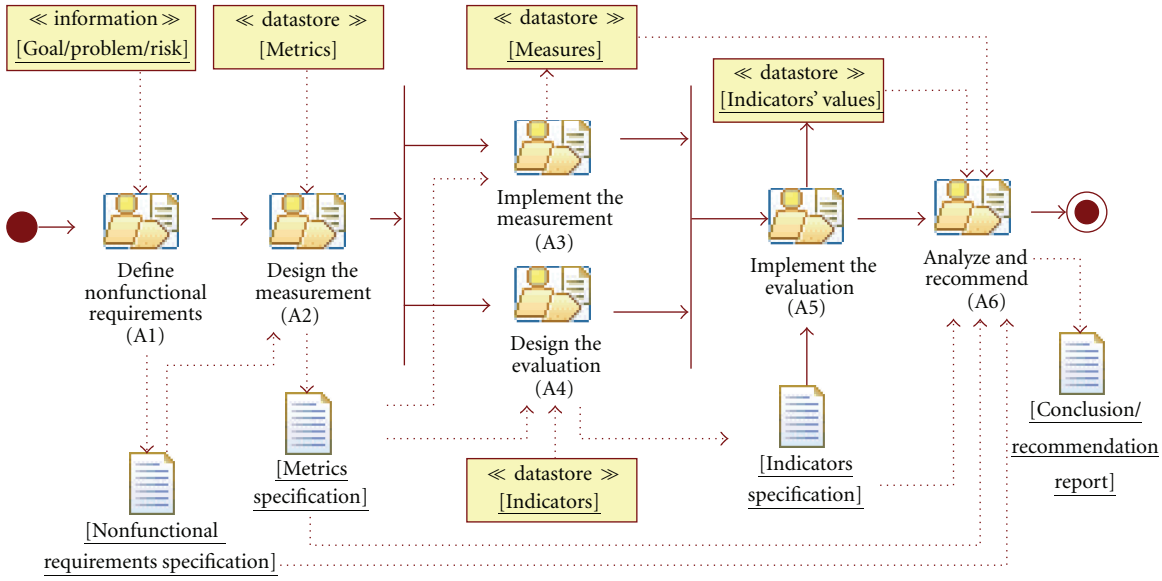


Figure 4: Overview of the GOCAME measurement and evaluation process.

Table 3: GOCAME core activities and main output artifacts.

Activities (A.)	Subactivities	Artifacts (Work Products)
A.1 Define Nonfunctional requirements	Subactivities include (i) establish information need; (ii) specify project context; (iii) select a concept model. Note that these (and below) subactivities can in turn be broken down in new ones—see [6] for more details.	Nonfunctional requirements specification (this artifact is composed of (i) information need specification; (ii) context specification; (iii) nonfunctional requirements tree)
A.2 Design the measurement	Subactivities include (i) establish entity (optional); (ii) assign one metric to each attribute.	Metrics specification
A.3 Implement the measurement	Subactivities include (i) establish entity; (ii) measure attributes	Measure values
A.4 Design the evaluation	Subactivities include (i) identify elementary indicators; (ii) identify partial and global indicators.	Indicators specification (this artifact is composed of (i) elementary indicators specification; (ii) partial/global indicators specification)
A.5 Implement the evaluation	Subactivities include (i) calculate elementary indicators; (ii) calculate partial and global indicators	Indicator values
A.6 Analyze and recommend	Subactivities include (i) design the analysis; (ii) implement the analysis; (iii) elaborate the conclusion report; (iv) perform recommendations.	Conclusion/recommendation report (this artifact is composed of (i) analysis specification; (ii) analysis report; (iii) conclusion report; (iv) recommendations report)

project has been created by the *nonfunctional requirements manager*, then, the *define non-functional requirements* activity has a specific *goal* or *problem* (agreed with the *evaluation requester*) as input and a *nonfunctional specification document* as output. Then, in the *design the measurement* activity, the *metrics expert* identifies the metrics to quantify attributes. The metrics are selected from a *metrics repository*, and the output is the *metric specification document*. Once the measurement was designed—taking into account raised issues for the evaluator requester, for example, the precision of metrics, and so forth—the *evaluation design* and the *measurement implementation* activities can be performed in any order or in parallel as shown in Figure 4. Therefore, the *design the evaluation* activity is performed by the *indicators expert* who allows identifying elementary and global indicators and their acceptability levels (agreed also with the *evaluation requester*). Both the *measurement design* and the *evaluation design* are led by the *measurement and evaluation managers* accordingly. To the *implement the measurement* activity, the *data collector* uses the specified metrics to obtain the measures, which are stored in the *measures repository*. Next, the *implement the evaluation* activity can be

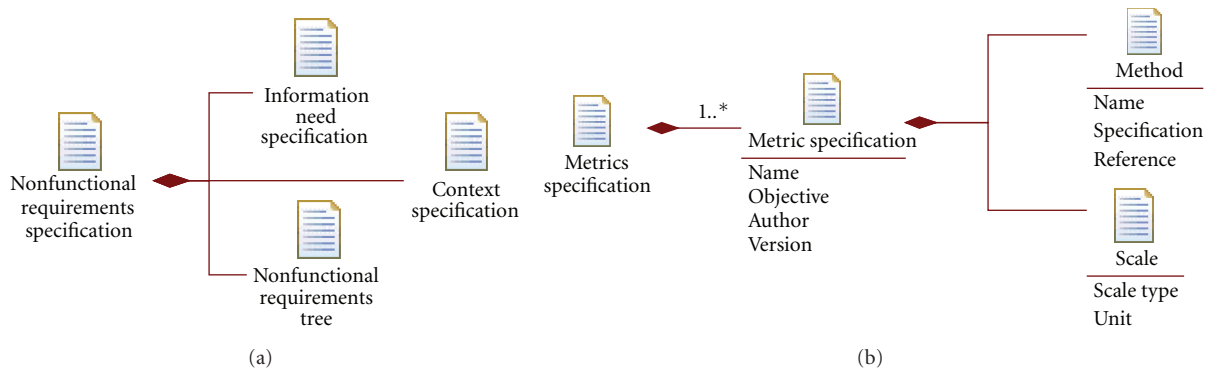


Figure 5: Excerpt of the informational view for A.1 and A.2 in Table 3, regarding artifact composition. (a) Nonfunctional requirements specification documents; (b) metrics specification document.

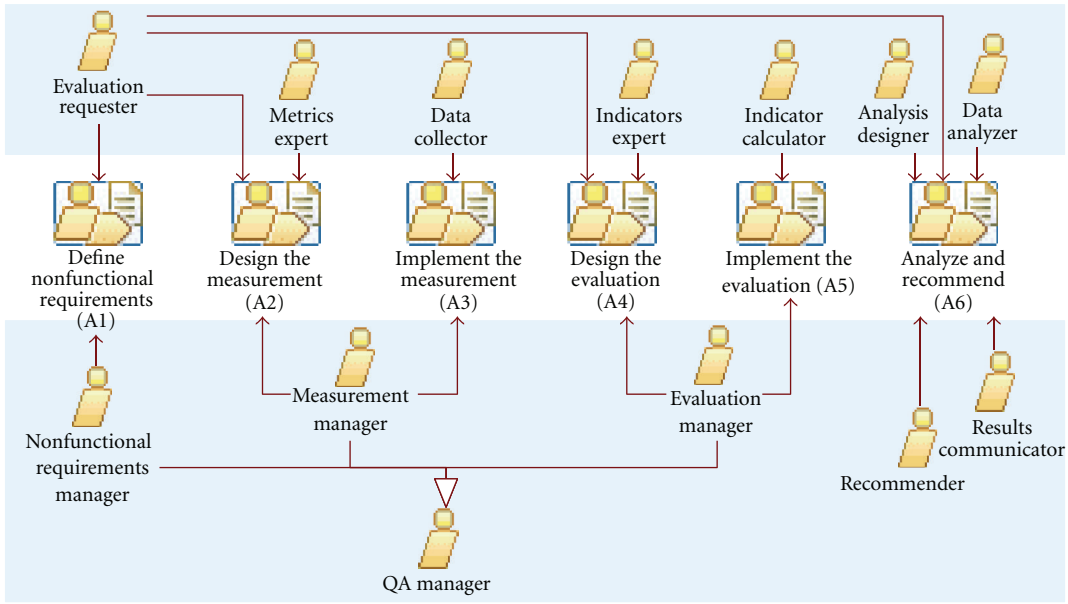


Figure 6: The organizational view: roles assigned to the GOCAME activities.

carried out by the *indicator calculator*—this role usually is enacted by a tool. Finally, *analyze and recommend* activity is performed by *analysis designer*, *data analyzer*, *recommender*, and *results communicator* roles. This activity has as inputs measures and indicators values (i.e., data), the requirements specification document, and the associated metrics and indicators specifications (i.e., metadata) in order to produce a *conclusion/recommendation report*.

3.3. GOCAME Methods and Tools: WebQEM and C-INCAMI_Tool. While activities state "what" to do, methods describe "how" to perform these activities accomplished by agents and roles, which in turn can be automated by tools. In addition, a methodology is a set of related methods. Since the above M&E process includes activities such as specify the requirements tree and identify metrics, we have envisioned a methodology that integrates all these aspects and tools that automate them; that is, a set of well-defined and cooperative methods, models, techniques, and tools that, applied consistently to the process activities, produces the different outcomes.

Particularly, the WebQEM and its associated tool the so-called C-INCAMI_Tool (see screenshots in Figure 7) were instantiated from the conceptual framework and process. The methodology supports an evaluation-driven approach, relying on experts and/or end users to evaluate and analyze different views of quality such as EQ and QinU for software and system-in-use applications. Note that GOCAME strategy and its methodology can be used to evaluate not only software/WebApps but also other entity categories, such as resources and processes.

In addition to the above-mentioned views, a methodological view is presented in [11]. This represents the process constructors to be applied to the different descriptions of activities in a given process. Note that, for a specific activity description, we can have one or more methods that give

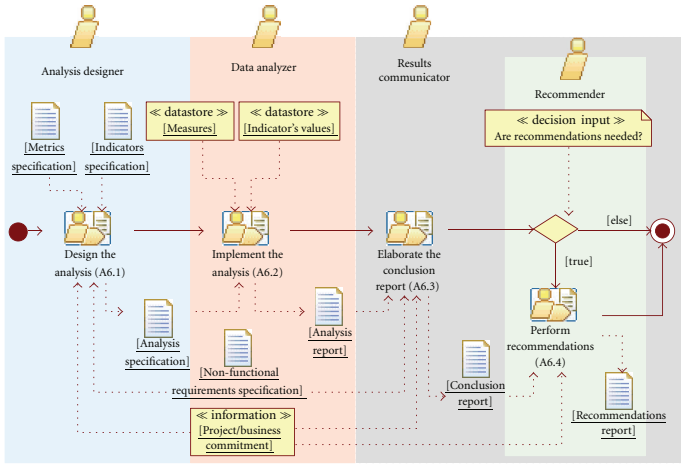
Table 4: GOCAME role definitions and involved activities.

Role name	Definition/comment	Activities (as per Figure 4)
Quality Assurance (QA) Manager	Responsible for leading a *measurement and evaluation project* (MEProject in Table 2) regarding the requester needs. Note this role is specified by three subroles as per Figure 5.	Note this role is responsible of the activities involved in three specific subroles as per Figure 5.
Nonfunctional requirements manager	Responsible for the *nonfunctional requirements project*. This role should be played by a nonfunctional requirement engineer.	(i) Define non-functional requirements
Measurement manager	Responsible for leading a *measurement project*.	(i) Design the measurement (ii) Implement the measurement
Evaluation manager	Responsible for leading an *evaluation project*.	(i) Design the evaluation (ii) Implement the evaluation
Evaluation requester	Responsible for requesting an evaluation. Note that this role can be accomplished by a human or an organization.	(i) Define nonfunctional requirements (ii) Design the measurement (iii) Design the evaluation
Metrics expert	Responsible for identifying the appropriate *metrics* from a catalogue for each *attribute* of the *requirements tree*, based on the established *information need*.	(i) Design the measurement
Data collector	Responsible for gathering *measures* of the *attributes* using the *metrics* specification. Note that the data collector role can be accomplished by either a human agent or an automatic agent.	(i) Implement the measurement
Indicators expert	Responsible for identifying the most appropriate *indicators* from a catalogue and to define *decision criteria* for each *attribute* and *calculable concept* of the *requirements tree* based on the established *information need*.	(i) Design the evaluation
Indicator calculator	Responsible for calculating the *indicators values* using the *indicators* specification. Note this role usually is accomplished by an automatic agent.	(i) Implement the evaluation
Analysis Designer	Responsible for identifying the appropriate data analysis methods and techniques to be used regarding scales, scale types, and the project/business commitment in addition to visualization and documentation techniques.	(i) Analyze and Recommend
Data analyzer	Responsible for conducting the data analysis based on the design of the analysis. Note this role can be accomplished by either a human agent or an automatic agent or both.	(i) Analyze and recommend
Recommender	Responsible for conducting the recommendations based on the conclusion report and taking into account the business commitment.	(i) Analyze and recommend
Results communicator	Responsible for communicating the evaluation results and recommendations to the evaluation requester.	(i) Analyze and recommend

support to the same activity, and, for a given method, we can have one or more tools that enact it. For instance, in Table 3, for the *A.5* activity, and particularly for the *calculate the partial/global indicators* subactivity, many methods can accomplish this such as "linear additive scoring method," "neural network method," among others.

3.4. Why Is SIQinU in Alignment with GOCAME? As we have indicated in the last paragraph of Section 2, SIQinU also relies on the three GOCAME principles above outlined and depicted in Figure 1. In fact, SIQinU utilizes the C-INCAMI conceptual base, underlying process and methods as we discuss in Section 4. However, since SIQinU is a specific-purpose goal-oriented strategy, it has specific activities, some particular methods, and procedures that are not taken into account in GOCAME. Moreover, while GOCAME is a multipurpose strategy regarding the strategy aim, SIQinU is a specific-purpose strategy. This is so, because in GOCAME the information need purpose can be "understand," "predict," "improve," "control"—as indicated in Section 3.1.2—while, in SiQinU, the purposes are just "understand" and ultimately "improve." In addition, GOCAME was designed to allow assessing different calculable concepts and entities such as the EQ or QinU of any product (including WebApps), the cost of a product, the capability quality of a resource, among others. Meanwhile, SIQinU was designed to evaluate specifically QinU of systems in-use (in a non-intrusive way) and EQ of systems, as for example, WebApps. Even more in SiQinU, from the nonfunctional requirements standpoint, QinU is evaluated from the "do goals" or pragmatic view, rather than from the "be goals" (subjective view), as thoroughly discussed in [2].

Table 5: Process template in which information and views are documented for the *Analyze and Recommend* activity.

Activity: analyze and recommend	*Code* (in Figure 4): A6
Objective: elaborate and communicate a conclusion report and (if necessary) a recommendation report for a decision-making process.	
Description: identify and select procedures, techniques and tools to be used in order to analyze data, metadata, and information, coming from metrics and indicators, for a given information need. Based on the analysis results, a conclusion report is produced, and, a recommendations report, if necessary, is yielded as well. All these reports are communicated to the evaluation requester.	
	*Subactivities*: (i) Design the analysis (A6.1) (ii) Implement the analysis (A6.2) (iii) Elaborate the conclusion report (A6.3) (iv) Perform recommendations (A6.4) *Involved roles*: (i) Analysis designer (ii) Data analyzer (iii) Recommender (iv) Results communicator
Input artifacts: (i) Nonfunctional requirements specification (ii) Metrics specification (iii) Indicators specification (iv) Measures (v) Indicators values (vi) Project/business commitment	*Output Artifacts*: Conclusion/Recommendation report. Note that this artifact is composed of (i) Analysis specification; (ii) Analysis report; (iii) Conclusion Report; and (iv) Recommendations report.
Preconditions: a MEProject must be implemented.	*Postconditions*: the MEProject finishes when the conclusion and/or recommendation report is communicated and agreed on between the QA manager and the requester of the evaluation.

Considering the process, SIQinU also reuses the GOCAME activities. For example, the GOCAME A1, A2, A4 activities, and, to some extent, the A6 activity (recall Figure 4) are included in SIQinU Ph. I and Ph. III (recall Figure 2). Likewise, the A3 and A5 activities are included in Ph. II and Ph. IV phases. However, there are particular activities in SIQinU that are not included into GOCAME. For example, in Phase V, we have activities devoted to produce WebApp improvements, as well as in Phase II, there exist activities for data filtering and collection, since SIQinU proposes utilizing server-side capabilities to gather, in a nonintrusive way, user usage data.

Considering the conceptual framework, SIQinU reuses totally C-INCAMI, that is, the ontological M&E conceptual base and its six components commented in Section 3.1. As above-mentioned SIQinU extends GOCAME activities, there are new activities (e.g., in Ph V for improvement techniques, and those related to nonintrusive data filtering in Ph. II), which lack the ontological root in C-INCAMI. Note that C-INCAMI concepts and components deal primarily with nonfunctional requirements, measurement, and evaluation issues, rather than functional aspects for design refactoring, code programming or restructuring, and so forth, which implies other domain scope and model. Note that C-INCAMI is a flexible framework that can be extended and linked with other domain models and frameworks to deal, for example, with functional aspects.

Also measurement and evaluation methods as commented in Section 3.3 are reused. However, other methods and techniques that are not included in GOCAME such as those for changing the current WebApp version (in Phase V) are needed. Finally, all the roles defined in Table 4 are totally reused as well, adding new ones for the activities of Ph V as, for example, the "Maintenance Project Manager" role (i.e., the responsible for leading a maintenance project and identifying the appropriate methods, techniques, and tools to be used for change—improve—the application) and the "Developer" role (i.e., the responsible for conducting the software/web application changes).

Despite the mentioned similarities with GOCAME, the modeling of the functional and behavioral views in SIQinU is necessary given the amount of involved phases, activities,

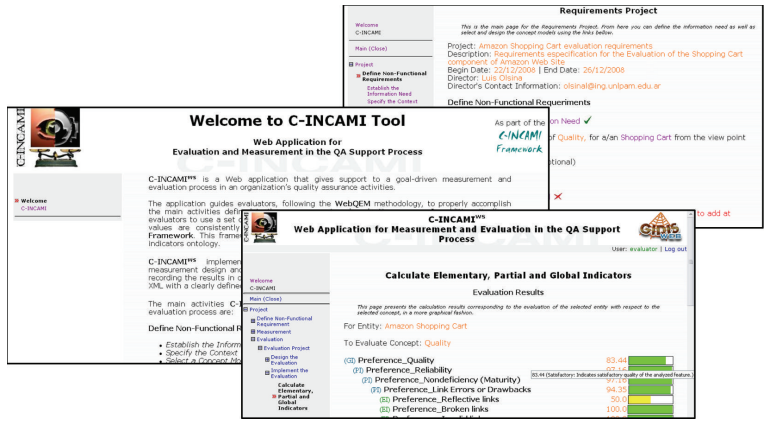


Figure 7: Snapshots of the C-INCAMI_tool.

subactivities, and their workflows. These issues will be highlighted in the next section.

4. A Process Model View for SIQinU

Process modeling is rather a complex endeavor. Given the inherent complexity of the process domain, a process can be modeled taking into account different views as analyzed in Section 3.2 for GOCAME. With the aim to model the SIQinU phases and activities, their inputs and outputs, sequences, parallelism, and iterations, we specify below using UML activity diagrams and the SPEM profile [14], the functional view taking into account behavioral concerns as well. Aspects of the organizational and informational views are to a lesser extent specified in the following diagrams, since as indicated in Section 3.4 many of the roles and artifacts are reused from the GOCAME strategy. Note that, in order to facilitate the communication, automation, and collaboration, different modeling languages and tools can be used to specify all the views. Each has its own strengths and weaknesses, which can make it more suitable for modeling certain views than others [12]. However, nowadays, SPEM is widely used and according to [17] can be adopted by researchers and practitioners for different disciplines, not just software engineering.

Also in Section 2 (Figure 2 and Table 1), we depicted the SIQinU phases, so below we concentrate on the specifications of activities and their descriptions. In order to illustrate the SIQinU process, excerpts of a case study conducted in mid-2010 are used (see details of the case study in [8]). This case study examined JIRA (http://www.atlassian.com/), a defect reporting WebApp in commercial use in over 24,000 organizations in 138 countries around the globe. JIRA's most common task, *Entering a new defect*, was evaluated in order to provide the most benefit, since entering a new defect represents a large percentage of the total usage of the application. We studied 50 beginner users in a real work environment in their daily routine of testing software and reporting defects in a software testing department of ABC, a company (with fictitious name but real one) specializing in software quality and testing. The beginner users were testers which were the majority of users. Although there are other user categories such as test managers, QA managers, and administrators, testers are the predominant user type, so we chose beginner testers as our user viewpoint.

4.1. Phase I: Specify Requirements and Evaluation Criteria for QinU. Once the requirements project has been created, using the data of the WebApp's usage recorded in log files, we reengineer and establish QinU requirements, that is, characteristics with measurable attributes, with the objective of not only understanding but also improving the system-in-use with real users. From observations of the actual WebApp, this phase embraces defining user type, designing tasks, specifying usage context and dimensions for QinU (e.g., actual usability) and their attributes. Based on these specifications, metrics (for measurement) and indicators (for evaluation) are selected. Below we describe the seven core activities (see the flow in Figure 8) involved in Ph. I.

4.1.1. Establish Information Need. This activity, according to the C-INCAMI framework (recall requirements package in Figure 3), involves *Define the purpose* and *the user viewpoint*, *establish the object* and *the entity* under study, and *identify the focus* of the evaluation (see Figure 9). These activities are accomplished by the NFR manager role considering the evaluation requester's needs. In the case study, the purpose for performing the evaluation is to "understand" and "improve" the WebApp being used from the userViewpoint of a "beginner tester." The category of the entity (i.e., the object) assessed was a "defect tracking WebApp" while the entity being studied was "JIRA" (called in our study JIRA v.1). The focus (CalculableConcept) assessed is "actual usability" and its subcharacteristics, "effectiveness in use," "efficiency in use," and "learnability in use" [2].

4.1.2. Specify Project Context. Once the information need specification document is yielded, we optionally can *Specify Project Context* as shown in Figure 8. It involves the *Select relevant Context Properties* subactivity—from the organizational repository of context properties [5], and, for each

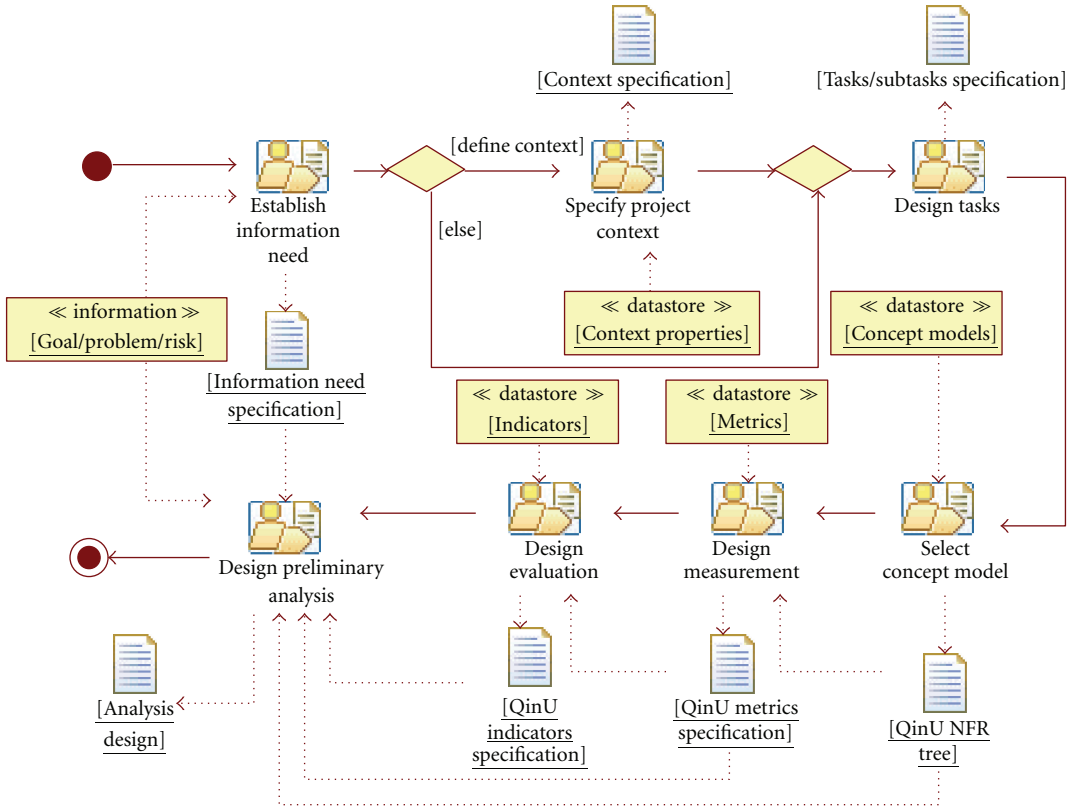


Figure 8: Overview for the specify requirements and evaluation criteria for QinU process (Ph. I).

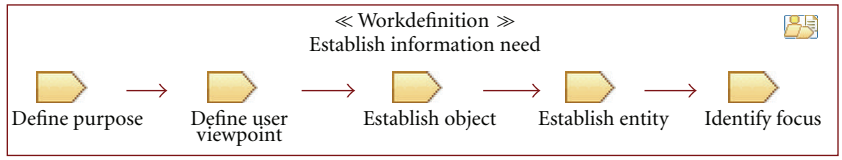


Figure 9: Activities for establish information need.

selected property, the *Quantify Context Property* activity must be performed—based on the associated metric. In the end, we get as output a context specification document for the specific project.

4.1.3. Design Tasks. In this activity, the most common and representative task or tasks should be designed. It is also important to choose a task that is performed for which sufficient data can be collected. In our case study, the selected task by the evaluation requester was "entering a new defect," as indicated above. This JIRA task included 5 subtasks specified by the task designer, namely: (i) Summary, steps, and results; (ii) Add Detail Info; (iii) Add Environment Info; (iv) Add Version Info; (v) Add Attachment (see details of tasks and screens in [8]).

4.1.4. Select QinU Concept Model. It involves both *Select a Model* and *Edit the Model* subactivities. Concept models are chosen from an organizational repository regarding the quality focus. For example, in our case study, the NFR manager based on the previously stated information need and taking into account the concept focus to evaluate actual usability, he instantiated a concept model for the "do goals" of the user [2]. Then, if the selected model is not totally suitable, for example, some subcharacteristics or attributes are missing, it is necessary to *Edit the Model*, adding or removing subconcepts, and/or attributes accordingly.

Finally, a requirements tree where attributes are the leaves and the concept focus is the root is yielded. For the selected concept model and regarding the information need and task at hand, the NFR manager instantiated the model as shown in Table 6 (attributes are in italic). Basically, the NFR manager, in the end, needs to satisfy the objectives of the sponsoring organization, that is, the evaluation requester.

4.1.5. Design QinU Measurement. For each attribute of the requirements tree—highlighted in italic in Table 6—we *Identify a Metric* to quantify them. The appropriate metrics

Table 6: Instantiated QinU NFR tree for JIRA case study.

1. Actual usability
1.1. Effectiveness in use
1.1.1. *Subtask correctness*
1.1.2. *Subtask completeness*
1.1.3. *Task successfulness*
1.2. Efficiency in use
1.2.1. *Subtask correctness efficiency*
1.2.2. *Subtask completeness efficiency*
1.2.3. *Task successfulness efficiency*
1.3. Learnability in use
1.3.1. *Subtask correctness learnability*
1.3.2. *Subtask completeness learnability*
1.3.3. *Task successfulness learnability*

are selected from a repository. In the C-INCAMI framework, two types of metrics are specified, a direct metric which applies a measurement method, that is, our data collection procedures from log files, and an indirect metric which uses a formula (based on other direct and/or indirect metrics) and calculation method (recall measurement package in Figure 3). If the metric is indirect, it is necessary *identify related metrics* and *identify attributes quantified by related Metrics* (see Figure 10). These two subactivities allow identifying the extra attributes and metrics for the indirect metric so that data collector may later gather the data accordingly.

In the JIRA case study, the metrics used to measure attributes were selected by the metrics expert from a metric catalogue which contains over 30 indirect metrics and their associated direct metrics. Below we illustrate the selected indirect metric for the subtask completeness efficiency (coded 1.2.2 in Table 6) attribute:

> Metric: average ratio of subtasks that are completed incorrectly or correctly per unit of time to do it (AvgRCput).
>
> Interpretation: 0 <= AvgRCput, more is better.
>
> Objective: calculate the overall average proportion of the subtasks that are completed, whether correct or incorrect, per time unit (usually seconds or minutes).
>
> Calculation Method (Formula): AvgRCput = AvgRC/AvgTC
>
> > AvgRC = Average ratio of subtasks that are completed incorrectly or correctly
> > AvgTC = Average time for a complete subtask, correct or incorrect
>
> Scale: numeric
>
> Type of Scale: ratio
>
> Unit (type, description): subtasks effectiveness/time, subtask completeness effectiveness per time unit (usually seconds or minutes).

As final output of these activities, we get the QinU metrics specification document.

4.1.6. Design QinU Evaluation. Once the metric specifications have been completed, we can design an indicator for each attribute and calculable concept of the requirements tree. Taking into account the C-INCAMI framework (recall evaluation package in Figure 3), there are two indicator types: elementary and global indicators. The elementary Indicators evaluate attributes and map to a new scale based on the metric's measures. The new scale is interpreted to analyze the level of satisfaction reached by each attribute. On the other hand, the global indicators (also called partial indicator if it evaluates a subcharacteristic) evaluate characteristics in a concept model and serve to analyze the level of global (or partial) satisfaction achieved.

Following the activities flow depicted in Figure 11, for each attribute of the requirements tree, the indicators expert should specify an elementary indicator by means of the next iterative activities: *establish the elementary model*, *establish the calculation method* (optional), and *identify the scale*.

The first activity (*establish the elementary model*) involves establishing a function to map between measure and indicator values and define the associated decision criteria or acceptability levels (see Section 3.1.5). In our case, the indicators expert and the evaluation requester defined three acceptability ranges in the indicator percentage scale, namely, a value within 70–90 (a marginal—bold—range) indicates a need for improvement actions; a value within 0–70 (an unsatisfactory—italic—range) means changes must take place with high priority; a score within 90–100 indicates a satisfactory level—bold italic—for the analyzed attribute. The acceptance levels in this case study were the same for all indicators, both elementary and partial/global, but could be different depending on the needs of the evaluation requester.

Note that the *establish the calculation method* activity is not mandatory because usually the model used is an easily interpreted function. In other cases, the calculation method should be specified.

Regarding to the partial/global indicators, these are specified in a similar way to the elementary indicators, as we can see in Figure 11. For example, in the JIRA case study, a global (linear additive) aggregation model to calculate the requirements tree was selected, with equal weights for their elements. This approach was used given that it was an exploratory study. Different weights would be assigned based on the requester's objectives to reflect the different levels of importance relative to one another. For example, for effectiveness in use, some organizations may weigh mistakes or correctness more heavily than completeness depending on the domain. A pharmacy or accounting application, for example, may have a higher weighting for accuracy.

The final output for the QinU evaluation design is an indicators specification document for quality in use. An artifact hierarchy (i.e., the informational view) of the indicators specification document is shown in Figure 12.

4.1.7. Design Preliminary Analysis. Taking into account the underlying SIQinU improvement objective, the specific QinU requirements for the project, the task, the metrics and indicators specifications, as well as the data properties with

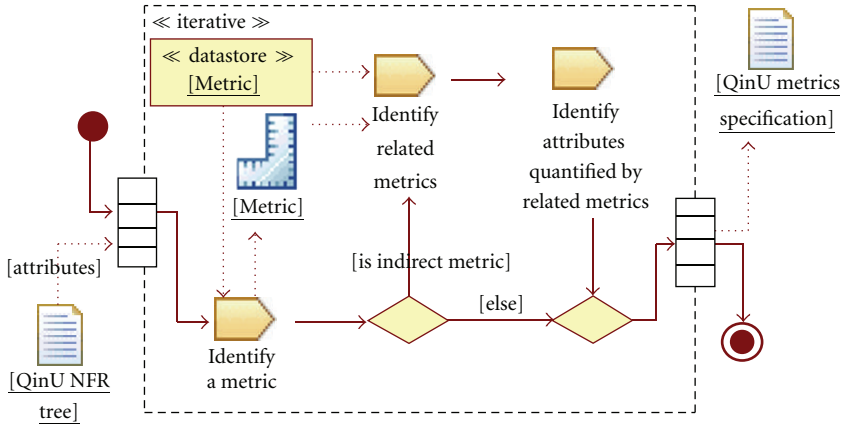


Figure 10: Design QinU Measurement activity.

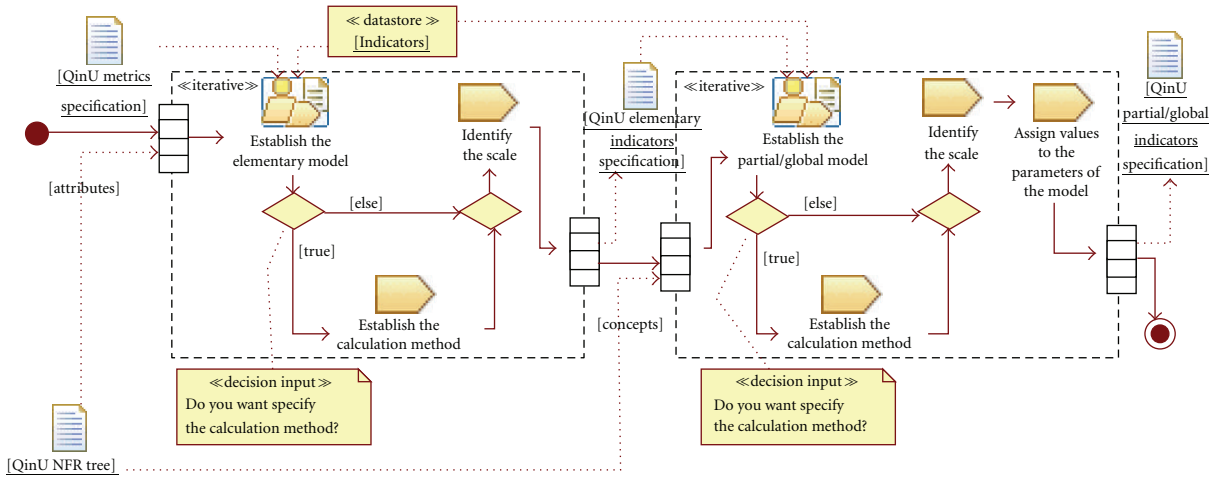


Figure 11: Design QinU evaluation activity.

regard to the scale, a preliminary analysis design should be drawn by the analysis designer role. This activity involves deciding on the allowable mathematical and statistical methods and techniques for analysis regarding the scale type, dataset properties, and so forth, the suitable tools for the kinds of analysis at hand, the presentation, and visualization mechanisms, and so forth.

4.2. Phase II: Perform QinU Evaluation and Analysis. This phase involves the basic activities to accomplish the first purpose of the SIQinU strategy, namely, understand the current QinU satisfaction level of the actual WebApp in use. To achieve this, the next four activities (see Figure 13) should be performed.

4.2.1. Collect Data. Taking into account the tasks specification, the log files with the user usage data are analyzed and the relevant data is filtered and organized to facilitate the measurement for each attribute in the next activity. Note that a tool can be used to process the log file for extracting the relevant data from user records.

4.2.2. Quantify Attributes. After collecting the data, we derive measurement values for each attribute in the QinU requirements tree. The values are obtained based on the measurement or calculation methods specified in QinU metrics specification according to the *design QinU measurement* activity (Figure 10).

4.2.3. Calculate Indicators. Taking into account the measures (values) and the indicators specification, the indicators values are calculated by the indicators calculator. The global indicator value ultimately represents the degree of satisfaction in meeting the stated information need for a concrete entity, for a given purpose, and user viewpoint. Within the *calculate indicators* activity, first, the *calculate elementary indicator* activity should be performed for each attribute of the requirements tree, and then, using these indicators' values and the specified partial or global model, the partial and global indicators are calculated by performing the *Calculate Partial/Global Indicator* activity for each calculable concept. Table 7, columns 2 and 3, shows each element of the QinU nonfunctional requirements tree evaluated at task level

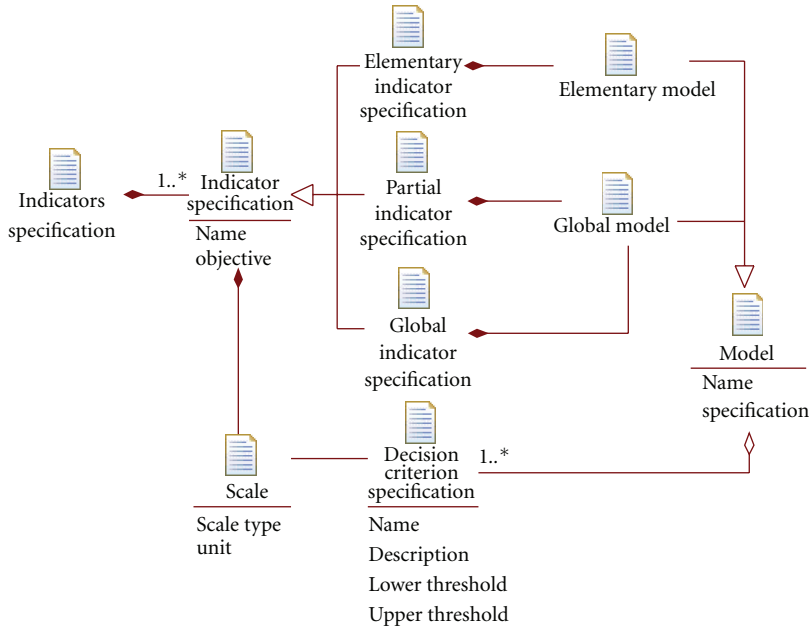


Figure 12: An informational view of the QinU indicators specification document.

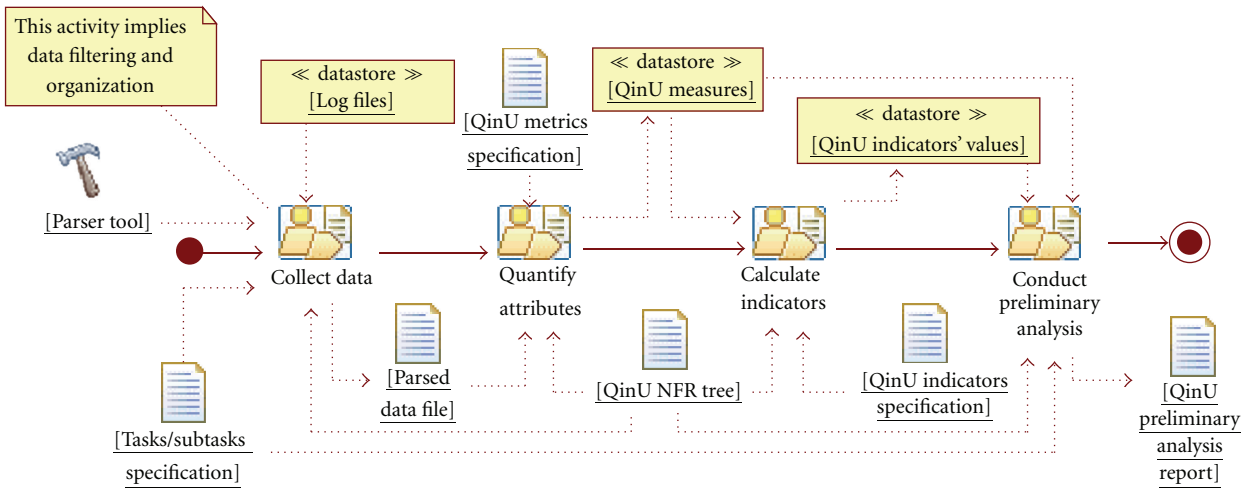


Figure 13: Overview for the perform QinU evaluation and analysis phase (Ph. II).

with elementary, partial and global indicators for the current version of JIRA (i.e., v.1).

4.2.4. Conduct Preliminary Analysis. After calculating indicators at all levels, that is, elementary, partial, and global, a preliminary analysis on the current JIRA WebApp task is conducted by the data analyzer role. Basically, it follows the analysis design (produced in the *design preliminary analysis* activity, in Ph. I), that is, implementing the designed procedures and planned tools, and storing results according to established formats in order to produce the preliminary analysis report. The analysis allows us to understand how the application performs overall (globally) and also with respect to each particular attribute for each part of the task (i.e., at subtask levels) being executed by a given user group type.

In the case study, the preliminary analysis was conducted for the above-mentioned task, its five subtasks, and their associated screens in JIRA, for the beginner user type. This allows the recommender to gauge more specifically where users had difficulty, for example, low task successfulness, low completion rate in using the application, among others.

4.3. Phase III: Derive/Specify Requirements and Evaluation Criteria for EQ. Taking into account the preliminary analysis report yielded in Phase II for QinU and the requirements tree defined in Phase I, in Phase III, a requirements tree for

Table 7: QinU evaluation of JIRA, both before (v.1) and after implementing improvements (v.1.1). EI stands for elementary indicator; P/GI stands for partial/global indicator.

Characteristics and attributes	JIRA v.1		JIRA v.1.1	
	EI	P/G I	EI	P/G I
1. Actual Usability		*53.3%*		*67.0%*
1.1. Effectiveness in use		**73.2%**		**86.7%**
1.1.1. *Subtask correctness*	**86.4%**		***91.9%***	
1.1.2. *Subtask completeness*	**87.9%**		***95.5%***	
1.1.3. *Task successfulness*	*45.5%*		**72.7%**	
1.2. Efficiency in use		*29.3%*		*42.8%*
1.2.1. *Subtask correctness efficiency*	*37.4%*		*44.3%*	
1.2.2. *Subtask completeness efficiency*	*37.5%*		*47.3%*	
1.2.3. *Task successfulness efficiency*	*13.1%*		*36.8%*	
1.3. Learnability in use		*57.3%*		**71.6%**
1.3.1. *Subtask correctness learnability*	**78.8%**		**75.1%**	
1.3.2. *Subtask completeness learnability*	*26.4%*		**77.3%**	
1.3.3. *Task successfulness learnability*	*66.7%*		*62.5%*	

EQ is derived. This requirements tree is tailored considering those product features that would need improvement with potential positive impact in QinU, mainly for those problems found in Phase II. In this phase, metrics and indicators are specified in order to evaluate the WebApp through its inspection involving three main activities (see Figure 14), namely, *select EQ concept model*, *design EQ measurement*, and *design EQ evaluation*. Note that these activities are similar to Phase I activities (recall Figure 8), but now from the EQ viewpoint.

4.3.1. Select EQ Concept Model. Given the preliminary analysis report performed in Phase II which may have reported potential problems of the actual WebApp, EQ characteristics and attributes possibly related to those QinU dimensions are identified, resulting then in a new requirements tree. The activities to be performed by the NFR manager are *select a model* and *edit the model*.

In the case study, in this activity, the requirements tree for the EQ viewpoint was established using 2Q2U (as in Phase I), instantiating the characteristics operability and information quality to determine possible effects on effectiveness in use, efficiency in use, and learnability in use. Those EQ characteristics and attributes that are possibly related to those QinU dimensions with potential problems have been instantiated resulting in the requirements tree shown on the left side of Table 8.

4.3.2. Design EQ Measurement. Following Figure 14, once the EQ model was instantiated in a requirements tree, the measurement should be designed to produce the metric specifications to perform Phase IV. As can be seen in Figure 15, this activity is similar to designing the QinU measurement (recall Figure 10 in Phase I) and is performed by the metrics expert, but now the process is executed for EQ attributes. In addition, next to *identify a metric* for an attribute from the repository and its related metrics and attributes (if the selected metric is an indirect metric), the *select a tool* activity can be performed to choose a tool that automates the metric method.

In the case study, for each attribute from the EQ requirements tree shown in Table 8, a metric was identified by the metrics expert. For instance, for the attribute navigability feedback completeness (coded 1.1.1.1), the metric is as follows.

Indirect Metric: task navigability feedback completeness (TNFC).

Objective: calculate the average of completeness considering the navigational feedback completeness level for all subtask screens for the given task.

Calculation method (formula):

$$\text{TNFC} = \sum_{j=1}^{j=n} \left(\sum_{i=1}^{i=m} \text{NFC}_{ij}/m \right)/n, \tag{1}$$

for $j = 1$ to n, where n is the number of subtasks of the given task,

for $i = 1$ to m, where m is the number of screens for subtask j.

Interpretation: 0 <= TNFC <=3, more is better.

Scale: numeric.

Scale type: ratio.

Unit: completeness level (*Note*: this metric can be converted to percentage unit, i.e., TNFC/0.03).

As this is an indirect metric, related metric and attribute were identified:

Attribute: screen navigation feedback.

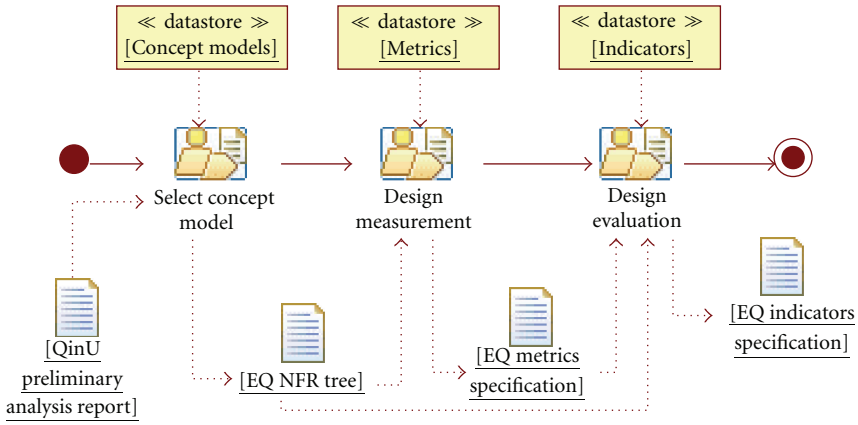


Figure 14: Overview for the derive/specify requirements and evaluation criteria for EQ process (Ph. III).

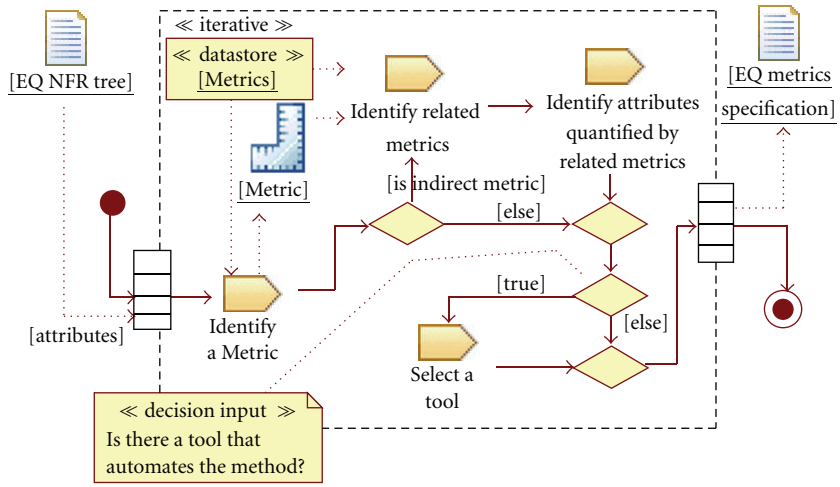


Figure 15: Activity flow involved in design external quality measurement activity.

Direct metric: navigation feedback completeness level (NFC).

Objective: determine the screen meets the criteria for navigation feedback completeness. *Note*: This metric is similar to the breadcrumb or path capability available in many WebApps.

Measurement method (type: objective): the screen is inspected to determine the rating (0–3), where evaluators through inspection observe the availability of the previous (backward), current, and next location (forward) mechanism. Screens should support the completeness of this navigation feedback.

Scale: numerical.

Scale type: ratio.

Allowed values: (0) has none; (1) has one of them; (2) has two of them; (3) has all of them.

Unit: completeness level.

4.3.3. Design EQ Evaluation. Similar to Phase I, one indicator per each attribute and concept of the EQ requirements tree should be identified by the indicators expert. In our case, when elementary, partial, and global indicators were designed, new acceptability ranges (DecisionCriterion in Figure 3) were agreed between the evaluation requester and the indicators expert. The three acceptability ranges in the indicator percentage scale were as follows: a value within 60–80 (a marginal—bold—range) indicates a need for improvement actions; a value within 0–60 (an unsatisfactory—italic—range) means changes must take place with high priority; a score within 80–100 indicates a satisfactory level—bold italic—for the analyzed attribute. Note that this indicator mapping does not necessarily have to be the same as the QinU mapping (e.g., may have different range thresholds) but rather should meet the information need and goals of the evaluation requester.

4.4. Phase IV: Perform EQ Evaluation and Analysis. Based on metric and indicator specifications obtained in Phase III, the measurement and evaluation of the EQ requirements and the analysis of the current product situation are performed. This phase is similar to Phase II, but now for EQ. The involved

activities are shown in Figure 16. Note the similarity with Phase II (recall Figure 13), but, for Phase IV, the *collect data* activity is not performed, that is, it is just carried out in Phase II to obtain user data usage from log files in a nonintrusive way. In this phase, the measurement and evaluation activities are done by inspection.

Once each attribute is measured by the data collector in *quantify attributes* activity, and all indicators are calculated in *calculate indicators* activity, the data analyzer role should *conduct EQ analysis*. The latter activity generates an EQ analysis report with information that allows us to identify, for instance, parts of the application that needs improvement from the EQ viewpoint. In Table 8 (columns 2 and 3), we can see the EQ evaluation results from JIRA (v.1) case study. Note, for example, that some attributes such as error prevention (coded 1.2.2.1.) and context-sensitive help availability (coded 1.1.2.1) need improvement with high priority. Also, we can observe that, for some elementary indicators (attributes), no improvement is needed, for example, stability of main control (coded 1.2.1.2).

4.5. Phase V: Recommend, Perform Improvement Actions, and Reevaluate EQ. Considering the previous EQ analysis report generated in *Conduct EQ Analysis* activity (Phase IV), we make recommendations to improve the application for those EQ attributes that needed improvement. After the recommended changes were completed in the current WebApp and a new version generated, we reevaluate the EQ to determine the improvement gain between both product versions. The activities for this phase are shown in Table 9 and described below.

4.5.1. Recommend Improvement Actions. Based on the EQ analysis report generated in Phase IV, the *recommend improvement actions* activity is carried out by the recommender in order to produce a recommendations report. This document has a set of recommendations for which attributes of the WebApp can be improved. For instance, a ranking of elementary indicators scored from weaker—that is, that fell in the unsatisfactory acceptability level—to stronger, but which did not fall in the satisfactory or bold italic levels can be listed.

Then, the evaluation requester can prioritize recommendations made for improvement action. Considering the case study, in this activity, some of the recommendations listed in the recommendations report were the following:

(i) for increasing the satisfaction level of defaults attribute (1.2.3.1) change fields to have default and make mandatory because they are critical defect description correctness and completeness;

(ii) for increasing the satisfaction level of Error Prevention attribute (1.2.2.1) add context sensitive help and eliminate nonvalid platform combinations.

4.5.2. Design Improvement Actions. Based on the previous recommendations report, the maintenance project manager produces an improvement plan indicating how to actually change the application. This "how" implies planning methods and techniques to be used to actually accomplish the improvement actions in the next activity (*perform improvement actions*). Methods and techniques for changing the WebApp can range from parameterized reconfigurations, code restructuring, refactoring (as made in [18]) to architectural redesign. The eventual method employed depends on the scope of the improvement recommendation as well as the resources of the evaluation requester and the desired effect. The expected effect may include an application easier to operate and learn, faster to run, more secure, among many other aspects.

For example, taking into account the two improvement recommendations listed in the above activity, the improvement plan included the following.

(i) *Recommendation*: add context sensitive help to improve the error prevention (1.2.2.1) attribute. *Action taken*: defect steps moved to next screen on add detail info, with help, examples shown to aid user.

(ii) *Recommendation*: eliminate nonvalid platform combinations to improve error prevention (1.2.2.1). *Action taken*: help provided and invalid combinations not allowed.

(iii) *Recommendation*: change fields to have default and make mandatory because they are critical defect description correctness and completeness to improve the defaults (1.2.3.1) attribute. *Action taken*: done where possible.

4.5.3. Perform Improvement Actions. With the improvement plan, the developer of the WebApp performs changes accordingly, resulting in a new application version (see the activity flow in Table 9). The ABC developer of our JIRA case study made some of the recommended changes, including those shown above, resulting in a new product version termed JIRA v.1.1. This new JIRA version had many other improvements not shown, one of which was the reduction of workload through eliminating one subtask and moving more related items together to make the overall task design more efficient. Thus, JIRA v.1.1 only has 4 subtasks, instead of 5. Because JIRA does not give access to its source code, the developer could not enact all the changes that were recommended; so only some improvements were made. Rather, through changing its configuration, they were able to perform most of the changes. Note that some recommended changes that could not be made were due to the application under study, and not due to SIQinU.

4.5.4. Evaluate Improvement Gain. Once changes were made, the WebApp can be reevaluated by inspection to determine which attributes have been improved, which have not, and get a score which can be compared to the outcomes of Phase IV. The activities involved are *quantify attributes*, *calculate indicators,* and *conduct EQ analysis*. The output is a new EQ analysis report in which the changes made

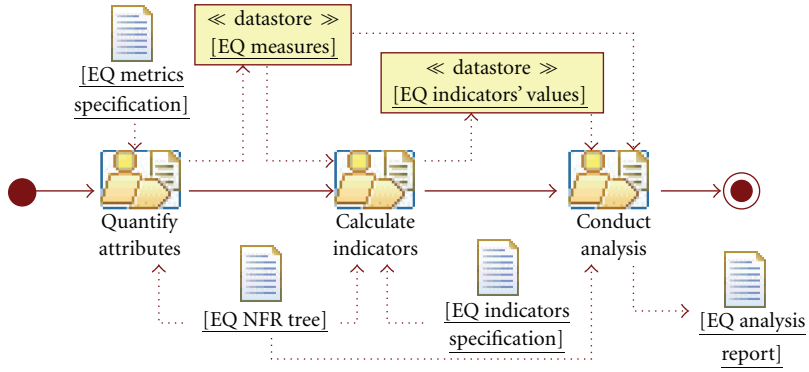


Figure 16: Overview for perform EQ evaluation and analysis phase (Ph. IV).

Table 8: EQ evaluation of JIRA, both before (v.1) and after (v.1.1) implementing improvements.

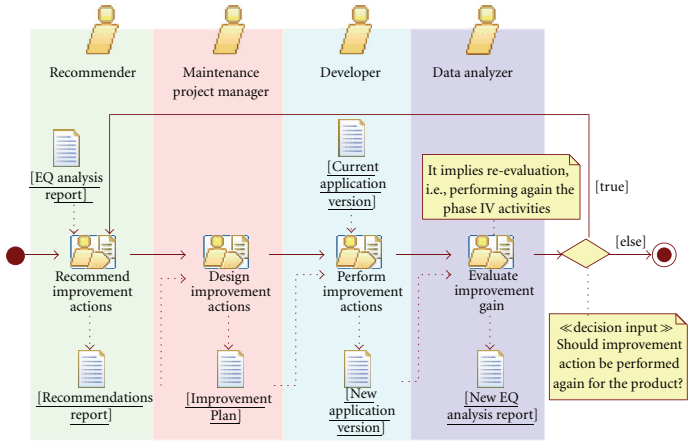
	JIRA v.1		JIRA v.1.1	
Characteristics and attributes	EI	P/G I	EI	P/G I
External Quality		*38%*		**74%**
1. Operability		*30%*		**60%**
1.1. Learnability		*26%*		*59%*
1.1.1. Feedback suitability		*38%*		*38%*
1.1.1.1. Navigability feedback completeness	*33%*		*33%*	
1.1.1.2. Task progress feedback appropriateness	*30%*		*30%*	
1.1.1.3. Entry form feedback awareness	*50%*		*50%*	
1.1.2. Helpfulness		*15%*		***80%***
1.1.2.1. Context-sensitive help availability	*20%*		***80%***	
1.1.2.2. Help completeness	*10%*		***80%***	
1.2. Ease of use		*34%*		**61%**
1.2.1. Controllability		***80%***		***80%***
1.2.1.1. Permanence of main controls	**60%**		**60%**	
1.2.1.2. Stability of main controls	***100%***		***100%***	
1.2.2. Error management		*0%*		*30%*
1.2.2.1. Error prevention	*0%*		*30%*	
1.2.3. Data entry ease		*23%*		**73%**
1.2.3.1. Defaults	*10%*		*50%*	
1.2.3.2. Mandatory entry	*10%*		***80%***	
1.2.3.3. Control appropriateness	*50%*		***90%***	
2. Information quality		*45%*		***88%***
2.1. Information suitability		*45%*		***88%***
2.1.1. Consistency		*40%*		***90%***
2.1.2. Information coverage		*50%*		***85%***
2.1.2.1. Appropriateness	*50%*		***90%***	
2.1.2.2. Completeness	*50%*		***80%***	

between the WebApp versions are compared to determine the improvement gain.

In Table 8 (columns 4 and 5), we can see the results obtained when JIRA v.1.1 was evaluated from EQ viewpoint. As can be seen from comparing the 2 evaluations (columns 2 and 3 with 4 and 5), the overall partial indicator for ease of use improved significantly from 34% to 61% with improvements in many of the individual attributes and an overall improvement in the global indicator from 38% to 74%. The next and final phase examines how these improvements in EQ affect QinU in a real context of use.

4.6. Phase VI: Reevaluate Quality in Use and Analyze Improvement Actions. Once the new application version (generated in Phase V, particularly in *perform improvement actions* activity) has been used by the same user group type in its

Table 9: Process template in which information and views are documented for the *recommend, perform improvement actions, and reevaluate EQ activities*.

Activity: recommend, perform improvement actions, and reevaluate EQ	*code* (in Figure 2): Ph. V
Objective: improve the current application version and determine the improvement gain from the EQ standpoint.	
Description: Considering the EQ analysis report generated in conduct EQ analysis activity (Phase IV), the recommender makes recommendations to improve the current application, and the maintenance project manager produces an improvement plan to enhance the current WebApp. After the recommended changes were implemented by the developer and a new version generated, a reevaluation of the EQ is performed to determine the improvement gain between both application versions.	
	*Subactivities*: (i) Recommend improvement actions (ii) Design improvement actions (iii) Perform improvement actions (iv) Evaluate improvement gain (i.e., in Phase IV) *Involved roles*: (i) Recommender (ii) Maintenance project manager (iii) Developer (iv) Data analyzer
Input artifacts: (i) EQ analysis report (ii) Current application version	*Output artifacts*: (i) EQ recommendations report (ii) Improvement plan (iii) New application version (iv) New EQ analysis report (from Phase IV)
Preconditions: there are EQ attributes with low level of satisfaction met, so improvement actions are needed to enhance the current software/web application version.	*Postconditions*: the Phase V finishes when the EQ attributes met the agreed satisfaction level.

same real context of use that the previous version, then, we are able to perform the QinU reevaluation (in a similar way to Phase II, recall Figure 13) to determine if what was improved from the EQ viewpoint had a positive quality-in-use effect using the identical tasks.

The activities involved are the same that in Phase II (see Figure 2), namely, *collect data*, *quantify attributes* and *calculate indicators*. Finally, *conduct improvement actions analysis* activity is performed to determine the improvement gain and also to hypothesize EQ/QinU relationships. These activities are described below according to the JIRA case study.

4.6.1. Collect Data, Quantify Attributes, and Calculate Indicators. When JIRA v.1.1 was evaluated from the EQ viewpoint, and its satisfaction level was achieved, then this new release was used by real users in the same real context of use as JIRA v.1. After 12 weeks (the same time period), we performed the QinU reevaluation (using the same nonfunctional requirements, metrics, and indicators designed in Phase I) to determine if what was improved from the EQ viewpoint had a positive quality-in-use effect with the same task (Entering a new defect). Following the SIQinU activities involved in Phase VI, we collect the data (i.e., the data collector using the same parser tool), quantify the attributes, and calculate all the indicators in a similar way as in Phase II. In Table 7, columns 4 and 5, we show the evaluation results for JIRA v.1.1.

4.6.2. Conduct Improvement Actions Analysis. Once all indicators were calculated, data analyzer looks at each particular attribute's change for each part of the task being executed by the user group noting the difference to calculate quantified improvement between both WebApp versions from the QinU viewpoint. Table 10 shows the attributes and indicator values regarding the QinU requirements tree depicted in Table 6 for JIRA v.1 and JIRA v.1.1 with the right most columns showing the change.

As we can see from Table 10, all attributes noted improvement with the exception of task successfulness learnability and subtask correctness learnability. A possible explanation for this is that due to metric design with data collected over a 12-week period that the learning process did not improve as much as expected. Remembering our earlier comments that temporal analysis over time is important as user behavior can change over time, these beginning users possibly were not able to ramp up their learning during this time period. And, on the other hand, if the case study had been longer, we

may have seen different behavior and hence measurements. However, their negative change was small compared to the positive changes in the other attributes resulting in an overall average change of attributes evaluation of 13.7%. While the indicators show that most of the attributes in JIRA v.1.1 still need some or significant improvement, there has been notable improvement from JIRA v.1. Again, as previously mentioned, due to the limitations in changing code in JIRA, configurations changes enabled many changes and improvements, but not all, it is possible that JIRA v.1.1 could be even better if more changes could be made. In other instances, depending on the software under evaluation, the tools/methods available to the maintenance project manager, and the time/resources available, more improvements could be made.

4.6.3. Develop "Depends-on" and "Influences" Relationships. A final action of Ph. VI (recall Table 1) is to develop *depends-on* and *influences* relationships between EQ improvements and QinU, which outputs the EQ/QinU attribute relationship table. These come also from the "influences" and "depends-on" relationships stated in the ISO 25010 standard. Table 11 summarizes the relationships found in this first case study. We discovered through our EQ evaluation, that some attributes could be improved and lead to specific improvements in QinU given the context of testers executing the specific task of entering a new defect in JIRA. In carrying out SIQinU, we were able to map EQ attributes to QinU attributes with the goal of ultimately achieving real improvement not only for JIRA but for WebApps and software design in general.

The weakness of our analysis is that we are only able to hypothesize the precise relationships between EQ and QinU attributes, but we cannot quantify the exact contribution from each because we made more than one change at a time. If we made only one change and then measured QinU for JIRA v.1.1, then we could make a more precise hypothesis for a one-to-one or one-to-many relationship. Most likely, those uncovered would be one-to-many, as one-to-one relationships probably are rare in this situation.

Regarding the hypothetical EQ/QinU relationships found in the JIRA case study, they can be further validated through additional case studies; case studies that can be carried out using our proposed SIQinU strategy grounded on a conceptual M&E framework (C-INCAMI), processes, and methods, which ensure repeatability and consistency of results. By doing this, we can generalize the conclusions of found relationships, that is, whether real improvements can be achieved not only for JIRA but for WebApps and software systems in general.

On the other hand, we are aware that this paper does not answer, from the experimental software engineering standpoint, the question of the effectiveness of the proposed strategy to accomplish similar or better objectives and results than using, for example, other strategies for improvement. The JIRA case study was made in a real context of a company, where beginner testers were performing their daily task and experts using SIQinU, as a first study. An experimental study to compare the effectiveness of SIQinU with another strategy should be performed as future work. Nevertheless, we can state that SIQinU is quite scalable in terms of amount of tasks and users performing tasks, and amount of metrics and indicators as well. Once the nonfunctional requirements are developed and data collection procedure is set, that is, the parser tool that filter and organize data from log files which serve as input to get the metrics values, then any number of metrics (and indicators) can be determined across an unlimited number of users. That is the scalability power of this nonintrusive strategy versus using usability observation and heuristic techniques.

5. Related Work

In recent years, the ISO/IEC has worked on a new project, called SQuaRE (*Software product Quality Requirements and Evaluation*), that proposes integrating many ISO standards related to quality models, M&E processes, and so forth. Although ISO 25000 [19] has guidelines for the use of the new series of standards, the documents aimed at specifying M&E processes are not issued yet. So the standards for software measurement process (ISO 15939 [20]) and the process for evaluators (ISO 14598-5 [21]) are still in effect and considered the most recent. Taking into account these standards, the process for measurement has two core activities, namely, *plan the measurement process* and *perform the measurement process* [20].The evaluation process involves five activities: *establishment of evaluation requirements*, *specification of the evaluation*, *design of the evaluation*, *execution of the evaluation plan*, and *conclusion of the evaluation* [21]. We have observed that there is so far no single ISO standard that specifies in an integrated way the M&E process and approach as a whole.

Other work, worthy to mention, is the CMMI (capability maturity model integration) [22] de facto standard, which provides support for process areas such as *measurement and analysis*, among others. It aligns information needs and measurement objectives with a focus on goal-oriented measurement—following to some extent the GQM (*goal question metric*) [23] approach and the [20] measurement process. Although CMMI specifies (specific/generic) practices to accomplish the given process area goals, a process model itself is not defined. To a certain extent, it represents practices (i.e., actions/activities) without explicitly establishing sequences, parallelisms, control points, and so forth. Some specific practices for measurement and analysis are for example, *establish measurement objectives*, *specify measures*, *obtain measurement data*, and *analyze measurement data*. However, a clear distinction between M&E processes is missing in addition to lacking a robust conceptual base for its terms.

Regarding improvement strategies for evaluation of WebApps, we can consider the work in [18], where authors present an approach for incremental EQ improvement whereby the results from EQ evaluation were used to make changes and improvements in a WebApp through WMR (*web model refactoring*) in a systematic way. But although a set of activities is considered, the underlying process is

Table 10: QinU attributes satisfaction level for JIRA v.1 and JIRA v.1.1 with improvements.

Attributes	v.1	v.1.1	Change	
1.1.1. *Subtask correctness*	**86.4**%	***91.9***%	5.5%	↑
1.1.2. *Subtask completeness*	**87.9**%	***95.5***%	7.6%	↑
1.1.3. *Task successfulness*	*45.5*%	**72.7**%	27.2%	↑
1.2.1. *Subtask correctness efficiency*	*37.4*%	*44.3*%	6.9%	↑
1.2.2. *Subtask completeness efficiency*	*37.5*%	*47.3*%	9.8%	↑
1.2.3. *Task successfulness efficiency*	*13.1*%	*36.8*%	23.7%	↑
1.3.1. *Subtask correctness learnability*	**78.8**%	**75.1**%	−3.7%	↓
1.3.2. *Subtask completeness learnability*	*26.4*%	**77.3**%	50.9%	↑
1.3.3. *Task successfulness learnability*	*66.7*%	*62.5*%	−4.2%	↓
		Average change	13.7%	

Table 11: Summary of the influence relationships between EQ and QinU attributes.

EQ attribute	QinU attribute
1.1.2.2 Operability.Learnability.Helpfulness.HelpCompleteness	Learnability in Use. Subtask completeness learnability
2.1.1 Information Quality.InfoSuitability.Consistency	
1.2.1.2 Operability.EaseOfUse.Controllability.StabilityMainControls	Effectiveness in Use. Task Successfulness
1.1.1.2 Operability.Learnability.Feedback Suitability.TaskProgressFeedbackAppropriateness	
1.1.1.3 Operability.Learnability.Feedback Suitability.EntryFormFeedbackAwareness	Efficiency in Use. Subtask completeness efficiency
1.1.2.1 Operability.Learnability.Helpfulness.Context-sensitiveHelpAvailability	
1.2.3.1 Operability.EaseOfUse.DataEntryEase.Defaults	
1.2.3.2 Operability.EaseOfUse.DataEntryEase.MandatoryEntry	
1.2.3.3 Operability.EaseOfUse.DataEntryEase.ControlAppropriateness	
2.1.1 Information Quality.InfoSuitability.Consistency	Effectiveness in Use.Subtask completeness
1.1.2.2 Operability.Learnability.Helpfulness.HelpCompleteness	Effectiveness in Use.Sub-task correctness
1.2.2.1 Operability.EaseOfUse.Error Mgmt.Error Prevention	
1.2.3.1 Operability.EaseOfUse.DataEntryEase.Defaults	
1.2.3.3 Operability.EaseOfUse.DataEntryEase.ControlAppropriateness	
2.1.2.2 Information quality.InfoSuitability.InfoCoverage.Completeness	
2.1.2.2 Information quality.InfoSuitability.InfoCoverage.Appropriateness	

neither well defined nor modeled regarding process views. On the other hand, in [24], a systematic approach to specify, measure, and evaluate QinU was discussed. However, the process used is not explicitly shown, and the outcomes were used just for understanding the current situation of the QinU for an e-learning application, without proposing any improvement strategy.

With the aim of developing quality software, there are several works that focus on improving and controlling the development process and the intermediate products because the quality of the final product is strongly dependent on the qualities of intermediate products and their respective creation processes. For example [25], deal with the use of a software project control center (SPCC) as a means for on-line collecting, interpreting, and visualizing measurement data in order to provide purpose- and role-oriented information to all involved parties (e.g., project manager, quality assurance manager) during the execution of a development project. On the other hand, in [26], the authors considered introducing usability practices into the defined software development process. With this goal in mind, authors offer to software developers a selection of human-computer interface (HCI) techniques which are appropriate to be incorporated into a defined software process. The HCI techniques are integrated into a framework organized according to the kind of software engineering activities in an iterative development where their application yields a higher usability improvement. Also, in the usability field, in [27], authors explore how some open source projects address issues of usability and describe the mechanisms, techniques, and technology used by open source communities to design and refine the interfaces to their programs. In particular, they consider how these developers cope with their distributed community, lack of domain expertise, limited resources, and separation from their users. However, in SIQinU, we start identifying problems in systems in-use, that is, with running applications

used by real users in a real context. Therefore, SIQinU is not focused on early stages of the development process but on how we can understand the current application-in-use's QinU and how we can change the attributes of a software system to improve its QinU.

Taking into account the complexity of processes, in [28], authors propose the use of an electronic process guide to provide a one-off improvement opportunity through the benefits of declaring a defined, systematic, and repeatable approach to software development. An electronic process guide offers several advantages over a printed process handbook, including easy access over the web for the most up-to-date version of the guide, electronic search facilities and hypernavigation to ease browsing information. In this work, authors combine the electronic process guide with the experience management, which refers to approaches to structure and store reusable experiences. It aims at reducing the overhead of information searching that can support software development activities. Experience management also appears to be more effective when it is process centric. Thus, the two concepts have a symbiotic relationship in that the process guide is more useful when supported by experiences and the experience base is more useful when it is process focused. The electronic process guide/experience repository (EPG/ER) is an implementation of this relationship and supports users through provision of guidance that is supplemented by task-specific experiences. However, our process specification is devoted specifically to represent the different views (as proposed in [10]) for measurement and evaluation activities rather than software development activities.

Regarding integrated strategies for M&E, it is worthy to mention the GQM$^+$ Strategies [29]—which are based on the GQM approach—as an approach that allows defining and assessing measurement goals at different organization levels, but it does not specify formal process views to conduct the evaluation and improvement lifecycle as we have shown as an integral component of SiQinU. Also, since issued, the GQM model was at different moments enlarged with proposals of processes and methods. However, [30] pointed out GQM is not intended to define metrics at a level of detail suitable to ensure that they are trustworthy, in particular, whether or not they are repeatable. Moreover, an interesting GQM enhancement, which considers indicators, has recently been issued as a technical report [31]. This approach uses both the balanced scorecard and the *goal-question-indicator-measurement* methods, in order to purposely derive the required enterprise goal-oriented indicators and metrics. It is a more robust approach for specifying enterprise-wide information needs and deriving goals and subgoals and then operationalizing questions with associated indicators and metrics. However, this approach is not based on a sound ontological conceptualization of metrics and indicators as in our research. Furthermore, the terms "measure" and "indicator" are sometimes used ambiguously, which inadvertently can result in datasets and metadata recorded inconsistently, and so it cannot assure that measurement values (and the associated metadata like metric version, scale, scale type, unit, measurement method, etc.) are trustworthy, consistent, and repeatable for further analysis among projects.

Finally, in [32], authors propose the CQA approach, consisting of a methodology (CQA-Meth) and a tool that implements it (CQA-Tool). They have applied this approach in the evaluation of the quality of UML models such as use cases, class, and statechart diagrams. Also authors have connected CQA-Tool to the different tools needed to assess the quality of models. CQA-Tool, apart from implementing the methodology, provides the capacity for building a catalogue of evaluation techniques that integrates the evaluation techniques (e.g., metrics, checklists, modeling conventions, guidelines, etc.), which are available for each software artifact. Compared with our strategies, the CQA approach lacks for instance an explicit conceptual framework from a terminological base. On the other hand, other related work in which authors try to integrate strategic management, process improvement, and quantitative measurement for managing the competitiveness of software engineering organizations is documented in [33]. In this work, a process template to specify activities from different views is considered. However, the integration of the three capabilities as made in GOCAME and SIQinU is not explicit and formalized.

6. Concluding Remarks

Ultimately, the main contribution of this paper is SIQinU, an integrated specific-purpose strategy—that is, for understanding and improving QinU—whose rationale is based on well-defined M&E processes, founded on a M&E conceptual framework backed up by an ontological base, and supported by methods and tools.

In this paper, we have specified the process of the SIQinU strategy modeled stressing the functional, informational, organizational, and behavioral views. Moreover, to illustrate the SIQinU process, excerpts from a JIRA case study were used where real users were employed to collect data and ultimately prove the usefulness of the strategy in improving the application in a process-oriented systematic means. We have also shown SIQinU, to be a derivation of GOCAME, based on three foundations, namely, the process, the conceptual framework, and methods/tools. Relying on the GOCAME foundation, SIQinU has been defined as a systematic approach with the appropriate recorded metadata of concrete projects' information needs, context properties, attributes, metrics, and indicators. This ensures that collected data are repeatable and comparable among the organization's projects. Otherwise, analysis, comparisons, and recommendations can be made in a less robust, nonconsistent, or even incorrect way.

SIQinU, although reusing the general principles of GOCAME, is a specific-purpose goal-oriented strategy with specific activities and methods that are not taken into account in GOCAME. Where GOCAME is a multipurpose strategy with general purposes such as "understand," "predict," "improve," and "control," SiQinU objectives are targeted to "understand" and ultimately "improve." In addition, as discussed in Section 3.4, SIQinU was specifically designed to evaluate QinU and EQ for WebApps, from the "do goals" perspective rather than from the "be goals."

As future work, we are planning to extend SIQinU to include processes and methods not only to gather data in a nonintrusive way (as currently it does) but also to gather data using more traditional intrusive methods such as video recording, observations, and questionnaires. This could not only help to add robustness to Phase III particularly in the derivation process from QinU problems to EQ attributes but also supplement the analysis in Phase II and VI.

Acknowledgment

This work and line of research is supported by the PAE 2007 PICT 2188 project at UNLPam, from the Science and Technology Agency, Argentina.

References

[1] ISO/IEC CD 25010.3. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models, 2009.

[2] P. Lew, L. Olsina, and L. Zhang, "Quality, quality in use, actual usability and user experience as key drivers for web application evaluation," in *Proceedings of the 10th International Conference on Web Engineering (ICWE '10)*, vol. 6189 of *Lecture Notes in Computer Science*, pp. 218–232, Springer, Vienne, Austria, 2010.

[3] N. Bevan, "Extending quality in use to provide a framework for usability measurement," in *Proceedings of the 1st International Conference on Human Centered Design (HCD '09)*, vol. 5619 of *Lecture Notes in Computer Science*, pp. 13–22, Springer, San Diego, Calif, USA, 2009.

[4] L. Olsina, F. Papa, and H. Molina, "How to measure and evaluate web applications in a consistent way," in *Web Engineering: Modeling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, Eds., HCIS, chapter 13, pp. 385–420, Springer, London, UK, 2008.

[5] H. Molina and L. Olsina, "Assessing web applications consistently: a context information approach," in *Proceedings of the 8th International Conference on Web Engineering (ICWE '08)*, pp. 224–230, Yorktown Heights, NJ, USA, July 2008.

[6] P. Becker, H. Molina, and L. Olsina, "Measurement and evaluation as quality driver," *Journal ISI (Ingénierie des Systèmes d'Information)*, vol. 15, no. 6, pp. 33–62, 2010.

[7] L. Olsina and G. Rossi, "Measuring Web application quality with WebQEM," *IEEE Multimedia*, vol. 9, no. 4, pp. 20–29, 2002.

[8] P. Lew, L. Olsina, P. Becker, and L. Zhang, "An integrated strategy to understand and manage quality in use for web applications," *Requirements Engineering Journal*, vol. 16, no. 3, 2011.

[9] E. Mendes, "The need for empirical web engineering: an Introduction," in *Web Engineering: Modelling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe, and L. Olsina, Eds., HCIS, chapter 14, pp. 421–447, Springer, London, UK, 2008.

[10] B. Curtis, M. Kellner, and J. Over, "Process modelling," *Communications of the ACM*, vol. 35, no. 9, pp. 75–90, 1992.

[11] L. Olsina, "Applying the flexible process model to build hypermedia products," in *Proceedings of the Hypertext and Hypermedia: Tools, Products, Methods (HHTPM '97)*, pp. 211–221, Hermes Ed., Paris, France, 1997.

[12] S. Acuña, N. Juristo, A. Merona, and A. Mon, *A Software Process Model Handbook for Incorporating People's Capabilities*, Springer, 1st edition, 2005.

[13] UML.Unified Modeling Language Specification, Version 2.0. Document/05-07-04, 2004.

[14] SPEM. Software Process Engineering Metamodel Specification. Doc./02-11-14., Ver.1.0, 2002.

[15] P. Becker, P. Lew, and L. Olsina, "Strategy to improve quality for software applications: a process view," in *Proceedings of the International Conference of Software and System Process (ICSSP '11)*, pp. 129–138, ACM, Honolulu, Hawaii, USA, 2011.

[16] N. E. Fenton and S. L. Pfleeger, *Software Metrics: a Rigorous and Practical Approach*, PWS Publishing Company, 2nd edition, 1997.

[17] F. García, A. Vizcaino, and C. Ebert, "Process management tools," *IEEE Software*, vol. 28, no. 2, pp. 15–18, 2011.

[18] L. Olsina, G. Rossi, A. Garrido, D. Distante, and G. Canfora, "Web applications refactoring and evaluation: a quality-oriented improvement approach," *Journal of Web Engineering, Rinton Press, US*, vol. 7, no. 4, pp. 258–280, 2008.

[19] ISO/IEC 25000. Software Engineering—Software product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE, 2005.

[20] ISO/IEC 15939. Software Engineering—Software Measurement Process, 2002.

[21] ISO/IEC 14598-5. International Standard, Information technology—Software product evaluation—Part 5: process for evaluators, 1999.

[22] CMMI Product Team. CMMI for Development Version 1.3 (CMMI-DEV, V.1.3) CMU/SEI-2010-TR-033, SEI Carnegie-Mellon University, 2010.

[23] R. Basili, G. Caldiera, and H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering*, J. J. Marciniak, Ed., vol. 1, pp. 528–532, John Wiley & Sons, 1994.

[24] G. Covella and L. Olsina, "Assessing quality in use in a consistent way," in *Proceedings of the International Conference on Web Engineering (ICWE '06)*, pp. 1–8, ACM, San Francisco, Calif, USA, July 2006.

[25] J. Munch and J. Heidrich, "Software project control centers: concepts and approaches," *Journal of Systems and Software*, vol. 70, no. 1-2, pp. 3–19, 2004.

[26] X. Ferre, N. Juriste, and A. M. Moreno, "Framework for integrating usability practices into the software process," in *Proceedings of the 6th International Conference on Product Focused Software Process Improvement (PROFES '05)*, vol. 3547 of *Lecture Notes in Computer Science*, pp. 202–215, Springer, 2005.

[27] D. M. Nichols and M. B. Twidale, "Usability processes in open source projects," *Software Process Improvement and Practice*, vol. 11, no. 2, pp. 149–162, 2006.

[28] F. Kurniawati and R. Jeffery, "The use and effects of an electronic process guide and experience repository: a longitudinal study," *Information and Software Technology*, vol. 48, no. 7, pp. 566–577, 2006.

[29] V. R. Basili, M. Lindvall, M. Regardie et al., "Linking software development and business strategy through measurement," *Computer*, vol. 43, no. 4, pp. 57–65, 2010.

[30] B. A. Kitchenham, R. T. Hughes, and S. G. Linkman, "Modeling software measurement data," *IEEE Transactions on Software Engineering*, vol. 27, no. 9, pp. 788–804, 2001.

[31] W. Goethert and M. Fisher, Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques. Software Engineering Measurement and Analysis Initiative, CMU/SEI-2003-TN-024, 2003.

[32] M. Rodrìguez, M. Genero, D. Torre, B. Blasco, and M. Piattini, "A methodology for continuos quality assessment of software artefacts," in *Proceedings of the 10th International Conference on Quality Software (QSIC '10)*, pp. 254–261, Zhangjiajie, China, July 2010.

[33] J. G. Guzmán, H. Mitre, A. Seco, and M. Velasco, "Integration of strategic management, process improvement and quantitative measurement for managing the competitiveness of software engineering organizations," *Software Quality Journal*, vol. 18, no. 3, pp. 341–359, 2010.