Inf Syst Front
DOI 10.1007/s10796-012-9401-x

Capturing data quality requirements for web applications by means of DQ_WebRE

César Guerra-García · Ismael Caballero · Mario Piattini

Abstract The number of Web applications which are part of Business Intelligence (BI) applications has grown exponentially in recent years, as has their complexity. Consequently, the amount of data used by these applications has also increased. The larger the number of data used, the greater the chance to make errors is. That being the case, managing data with an acceptable level of quality is paramount to success in any organizational business process. In order to raise and maintain adequate levels of Data Quality (DQ), it is indispensable for Web applications to be able to satisfy specific DQ requirements. To do so, DQ requirements should be captured and introduced into the development process of the Web Application, together with the other software requirements needed in the applications. In the field of Web application development, however, there appears to us to exist a lack of proposals aimed at managing specific DQ software requirements. This paper considers the MDA (Model Driven Architecture) approach and, principally, the benefits provided by Model Driven Web Engineering (MDWE), putting forward a proposal for two artifacts. These consist of a metamodel and a UML profile for the management of Data Quality Software Requirements for Web Applications (DQ_WebRE).

C. Guerra-García (✉)
Department of Information Technologies, Polytechnic University of San Luis Potosí, UPSLP, Urbano Villalón 500, 78363 San Luis Potosí, México
e-mail: Cesar.guerra@upslp.edu.mx

I. Caballero · M. Piattini
Institute of Information Systems and Technologies, University of Castilla-La Mancha, Camino de Moledores s/n, 13071 Ciudad Real, Spain

I. Caballero
e-mail: Ismael.caballero@uclm.es

M. Piattini
e-mail: Mario.piattini@uclm.es

Keywords Data Quality · Web Engineering · Requirements Engineering · Model Driven Web Engineering · Requirements modeling

1 Introduction

Looking at the present state of affairs, it is clear that more and more companies have migrated their applications to Internet based solutions, managing a large amount of data through Web applications, in the main. The usage of these applications has created new ways of boosting their business, taking advantage of business intelligence applications in a way that the vast potential of customer relationships has never been exploited before (Phan and Vogel 2010). It is important to highlight that the usability of Web portals and applications it is paramount for their success (Fang and Holsapple 2011). However, problems due to inadequate levels of quality in the data which flows through these Web applications arise more commonly than expected with unexpected results (Bertino et al. 2010; Caro et al. 2008; Janjua et al. 2012).

(C Batini et al. 2007) mention some examples of common situations in which Information Systems that use data with inadequate levels of quality have negatively affected the work of employees, the satisfaction of the customers, and, consequently, organizational performance.

It can be proven that these problems provoke different kinds of damage within organizations (Ballou and Pazer 2003; Kahn et al. 2002; Pipino et al. 2002; Scannapieco and Berti-Équille 2006; Shankaranayanan and Cai 2005). This damage represents higher and higher penalty costs, both in economic and in social terms (Eppler and Helfert 2004; Laudon 1986; Wang et al. 1995). But only when organizations are conveniently aware of the consequences, it is only when they actually suffer from these that they want to eradicate this kind of problems.

Published online: 22 December 2012

The first typical initiative is to be reactive to such kinds of problems (Sarsfield 2009): so organizations consider the adoption of specific Data Quality Software (e.g. data cleansing, standardization, matching, merging, enrichment and data profiling), as proposed in (Karel et al. 2009). Although useful, this can only be used as a "*post-mortem*" solution, and does not avoid problems in the long term, since an Information System is in a continuous process of *living* (Guerra-García et al. 2011). In addition, these solutions are not commonly proactive, and do not focus on the data quality requirements of specific users (Lucas 2010). Neither can they comprise organizational resources a fact, which implies that problems will come back and that the organization will lose money again. An alternative that could be really productive, therefore, is to fix systematic errors by means of some kind of customization of the Information System, trying to solve, or at least alleviate, the effects of some DQ potholes (D. Strong et al. 1997).

This customization is based upon the concept of Data Quality Requirement (DQR), which can be defined as: "*the specification of a set of dimensions of Data Quality that a set of data should meet for a specific task performed by a given user*" (Guerra-García et al. 2011). For each one of the tasks to be executed using the Information Systems, some specific Data Quality Requirements are to be collected, managed, and later transformed into the corresponding Data Quality Software Requirements (DQSR). Such DQSR are to be introduced during the earliest phases of the Web Application development as a complement to the remainder of the Software Requirements.

Much research in MDWE (Model Driven Web Engineering) is concerned principally with the analysis and design phases. In this respect, different languages, methods and tools for Web modeling have been proposed and released, almost all of which offer specific processes to support the systematic and semiautomatic development of these applications. This therefore makes MDWE a good starting point for the insertion of new features, such as DQ issues. This is the reason why we have decided to work upon the MDWE paradigm, with the intention of making it easy for our findings and results to be shared with any other target platforms (e.g. desktop applications). As is well-known, MDWE proposes a representation of concepts by means of supporting the development process using a set of models, transformations and relations between models. This leads to agile developments and assures consistency between models (Escalona and Aragón 2008).

On the other hand, as concluded in (Guerra-García et al. 2011), there are no works that even partially cover the various corresponding issues related to the management of DQ software requirements when Web applications are being modeled and developed. The lack of methodologies and proposals for these DQ software requirement specification initiatives leads to the need to consider such requirements throughout the software development process. In a more specific sense, it highlights the need for these to be borne in mind in the phase of specification of initial requirements (Guerra-García et al. 2009).

Hence, the main contribution of this work towards both areas, namely Requirements Engineering and MDWE, is twofold: on one hand we propose an extended metamodel to help developers to identify DQR better, and on the other hand, we put forward a UML profile which will allow developers to translate such DQR into DQSR, so that DQ issues could be introduced throughout the various diagrams (use case and activity). This all works towards the development of *Data Quality-aware Web applications*.

The remainder of the paper is structured as follows: Section 2 provides a brief description of the model's foundations in DQ, in Web Engineering and in the metamodel (*WebRE*). The extended metamodel with DQ and the proposed UML profile for specification and modeling of Data Quality software requirements (*DQ_WebRE*) are introduced in Section 3. Section 4 shows an illustrated example using the *DQ_WebRE* profile, and finally, some conclusions and future work are presented in Section 5.

2 Foundations of our proposal

In this section, a brief description is provided of the two pillar areas related to our proposal: *Data Quality* and *Web Engineering*.

2.1 Data quality

Various definitions of the concept of Data Quality have been proposed over the last few years (C. Batini et al. 2009). Most of the authors agree, however, that a piece of data has an adequate level of quality if it is valid for the purpose to which a user wishes to put it as regards a particular task in a specific context (D. M. Strong et al. 1997). This is based on Juran's "*fitness for use*". One of the most interesting strategies for the study of DQ for a specific context is to divide it into smaller pieces known as *data quality Dimensions* (Lee et al. 2006). It is essential to point out here that the set of several data quality dimensions is typically known as a *DQ Model*.

We ought to highlight here that when a user is specifying his/her data quality requirements (DQR), s/he can choose those data quality dimensions from those proposed in the model provided in (D. M. Strong et al. 1997). The corresponding DQR are then to be translated into DQSR, which is the central point of the customization of the Information System (IS) for supporting Data Quality concerns. The customization consists of adding to the existing features of the IS around the data used those characteristics that allow us to guarantee that the DQR could be satisfied. In this last sense, the special characteristics for customizing the data are identified in a generic manner by the ISO/IEC 25012 (ISO-25012 2008) standard. This provides a Data Quality model for target data managed in information systems, and considers fifteen characteristics which are classified in two groups:

– *Inherent*: this refers to the extent to which quality characteristics of data have the intrinsic potential to satisfy stated and implied needs when data is used under specified conditions.
– *System dependent*: this refers to the extent to which data quality is obtained and preserved within a computer system, when data is used under specified conditions.

Table 1 shows the definitions for each one of the data quality characteristics proposed by the ISO/IEC 25012 standard.

It is worth mentioning that these characteristics should be reinterpreted and redefined each time, in an effort to represent with greater precision how to measure the level of data quality of a piece of data in a specific context.

2.2 Web engineering

Many methodological proposals have arisen for the development of Web applications in a systematic and rigorous way. All of them focused mainly on supporting requirements analysis and design phases: NDT (Escalona and Aragón 2008), UWE (Koch and Kraus 2002), WebML (Ceri et al. 2000), WebRE (Escalona and Koch 2006), SOD-M (De Castro and Marcos 2009) and WebSA (Meliá and Gómez 2005). A comparative study of these methodologies is shown in (Escalona and Koch 2004). This study shows the types of requirements managed by each proposal, principally, along with the techniques used and the extent of detail of each proposal in terms of its development process.

The NDT proposal (*Navigational Development Technique*) (Escalona and Aragón 2008) presents a methodology that covers the requirements and analysis phases in Web development. This methodology is based completely on the MDWE approach; it proposes NDT metamodels and

Table 1 Data Quality characteristics proposed by the ISO/IEC 25012 standard

Characteristic	Description
Inherent	
Accuracy	The degree to which data have attributes that correctly represent the true value of the intended attribute of a concept or event in a specific context of use.
Completeness	The degree to which subject data associated with an entity have values for all expected attributes and related entity instances in a specific context of use.
Consistency	The degree to which data have attributes that are free from contradiction and are coherent with other data in a specific context of use.
Credibility	The degree to which data have attributes that are regarded as true and believable by users in a specific context of use.
Currentness	The degree to which data have attributes that are of the right age in a specific context of use.
Inherent and System dependent	
Accessibility	The degree to which data can be accessed in a specific context of use, particularly by people who need supporting technology or special configuration because of some disability.
Compliance	The degree to which data have attributes that adhere to standards, conventions or regulations in force and similar rules relating to data quality in a specific context of use.
Confidentiality	The degree to which data have attributes that ensure that they are only accessible and interpretable by authorized users in a specific context of use.
Efficiency	The degree to which data have attributes that can be processed and provide the expected levels of performance by using the appropriate amounts and types of resources in a specific context of use.
Precision	The degree to which data have attributes that are exact or that provide discrimination in a specific context of use.
Traceability	The degree to which data have attributes that provide an audit trail of access to the data and of any changes made to the data in a specific context of use.
Understandability	The degree to which data have attributes that enable it to be read and interpreted by users, and are expressed in appropriate languages, symbols and units in a specific context of use.
System dependent	
Availability	The degree to which data have attributes that enable them to be retrieved by authorized users and/or applications in a specific context.
Portability	The degree to which data have attributes that enable them to be installed, replaced or moved from one system to another while preserving the existing quality in a specific context of use.
Recoverability	The degree to which data have attributes that enable them to maintain and preserve a specified level of operations and quality, even in the event of failure, in a specific context of use.

transformations among them. NDT offers a suitable environment for the elicitation, definition, analyzing and validation of Web requirements.

UWE (*UML-based Web Engineering*), Koch and Kraus's proposal, (Koch and Kraus 2002) covers the special aspects of Web application analysis and design: these authors define a set of special views represented graphically by UML diagrams, such as a navigation model and a presentation model, through the use of UML profiles. They also show how to use different kinds of static diagrams to model the static aspects of the Web applications. This proposal is based on a common metamodel and also includes tools to support the design and semi-automatic generation of Web applications.

WebML (*Web Modeling Language*) (Ceri et al. 2000) is a high-level language for the specification and the design of data-intensive Web applications. It emphasizes the definition of primitives for composition and navigation which can be used to design complex requirements. It also encompasses some advanced modeling aspects of Web sites, including presentation, user modeling and personalization.

In the WebRE (*Web Requirements Engineering*) proposal by (Escalona and Koch 2006), the authors focus on Web requirements specification through models. They present a metamodel which contains the key concepts needed for the specification of specific requirements, such as: use cases of navigation, use cases of Web process, specific activities and structural elements (nodes, user interface, etc.).

The SOD-M (*Service-Oriented Development Method*) proposal (De Castro and Marcos 2009) presents a service-oriented approach towards information system development that starts by using business modeling to identify the services required by the customers of a business, thus making it possible to create a Web service composition model. It uses UML as the modeling language and defines a specific UML profile for the service-oriented development. SOD-M focuses on the development of the behavioral aspect and defines guidelines with which to build behavioral models from high-level business modeling.

In (Meliá and Gómez 2005), the authors propose a generic design approach named WebSA (*Web Software Architecture*). This is based on the MDA (Model Driven Architecture) paradigm, and proposes a model-driven development of a set of architectonic models of UML and QVT (Query/View/Transformations) transformations as mechanisms with which to integrate the functional aspects of the actual methodologies with the architectonic aspects.

All of these methodologies focus primarily on how to identify and define the functional aspects of the Web Application: those related to the semantics of models and those oriented towards capturing the relevant properties of this type of Web applications. It should be remarked, however, that none of these proposals includes ways of introducing issues regarding the quality characteristics of the data that is managed and stored by these applications. Only a few of them, such as those in (Ceri et al. 2000; Escalona and Aragón 2008; Escalona and Koch 2006), give a brief mention to certain needs with respect to implementing specific information objectives that should be considered when designing a Web application. They neither explore their study in greater depth, however, nor consider any requirements or specifications of DQ characteristics.

2.3 WebRE metamodel

It is worth highlighting that the key concepts managed in the *WebRE* metamodel were defined by taking as a basis the similarities of all methods and proposals reviewed by the authors and summarized in (Escalona and Koch 2006). *WebRE* uses the power of metamodeling to merge different approaches. It also defines a unified metamodel, in accordance with certain OMG standards such as *MDA* (OMG 2001), UML (OMG 2005b), OCL (OMG 2005a), QVT (OMG 2008). This alignment to the most widely-used standard in Software Engineering is a way of assuring its capability of adaptation to any other software development.

The metamodel proposed by Escalona and Koch in (Escalona and Koch 2006) permits the main elements for Web requirements to be modeled in a UML class diagram. The metaclasses represent concepts without any information about their representation; they are grouped in two packages according to the structure of UML: "*WebRE Structure*" and "*WebRE Behavior*".

The functionality of a Web system, described in the "*WebRE Behavior*" package, is modeled by means of a set of instances of two types of specific use cases: "*Navigation*" and "*WebProcess*", and specific activities such as "*Browse*", "*Search*" and "*UserTransaction*".

The "*WebRE Structure*" package contains the metaclasses used to describe the structural elements of a Web application: *Node*, *Content* and *Web User Interface* (*WebUI*). A brief description of each element is shown in Table 2.

The UML profile for Web requirements engineering specifies how the concepts of the WebRE metamodel relate to, and are represented in, the UML standard, using stereotypes and constraints (Escalona and Koch 2006).

One of the main advantages of this metamodel is its flexibility: it allows the easy inclusion of new elements. This thus enables developers to add some of these to manage some new concepts, as we have done on adding the DQ concerns, in order to specify and model the DQ software requirements. Establishing the corresponding relationships of these new DQ elements to each one of the elements listed in the WebRE profile, e.g. use cases ("*WebProcess*"), or specific activities like "*UserTransaction*".

In the following section, we set out a complete description of our proposed metamodel and the UML profile for managing specific data quality requirements in the development of Web applications.

Table 2 Elements of WebRE metamodel

Element	Description
WebUser	Represents any user who interacts with the Web application.
Navigation	Represents a specific use case which includes a set of "*Browse*" type activities that the *WebUser* will be able to perform to reach a target node.
WebProcess	Models the main functionalities (normally *business process*) of the Web application. It represents another use case which can be refined by different *Browse*, *Search* and *UserTransaction* type activities.
Browse	Represents a normal browse activity in the system; it can be improved by a *Search* activity.
Search	It has a set of parameters, which allow us to define queries on the data storage in "*Content*" metaclass. The results will be shown in the target node.
UserTransaction	Represents complex activities that can be expressed in terms of transactions initiated by users.
Node	Represents a point of navigation at which the user can find information. Each instance of a Browse activity starts in a node (source) and finishes in another node (target). The Nodes are shown to the users as pages.
Content	Represents where the different pieces of information are stored.
WebUI	Represents the concept of Web page.

3 A proposal for a metamodel and a profile for including data quality concerns in Web applications

After carrying out an in-depth analysis of the various Web Engineering proposals, and given their features, we decided to take the one proposed by (Escalona and Koch 2006) as a basis for our work. It satisfies one of the key requirements for our research: its compatibility with "de jure" standards in the market. Escalona and Koch's proposal presents a metamodel with which to represent concepts and relationships of Web Requirements Engineering. This metamodel is used as a basis for defining a UML profile for Web Requirements (*WebRE*) (Escalona and Koch 2006).

Having shown the main characteristics and elements of the *WebRE* metamodel in Section 2.3, in this section we should describe our proposal. One of the most important motivations of this work is to provide analysts and designers of Web applications with the artifacts needed to specify and describe the corresponding DQSR in a clear and intuitive manner.

To develop our proposal, we have extended Escalona and Koch's metamodel, in order to deal with those elements which are considered to be essential for the specification of DQSR and integrate them into the proposal. Having conducted a systematic review on the main proposals for the specification and modeling of DQ requirements, presented in (Guerra-García et al. 2010), we decided to incorporate the following key elements (namely stereotype) the majority of which were inferred mainly from (Becker et al. 2009; Becker et al. 2007; Caballero et al. 2007) (see Fig. 1):

– For the Behavior Package we have included the following new classes: "*InformationCase*", "*DQ_Requirement*", "*DQ_Req_Specification*" and "*Add_DQ_Metadata*";
– For the Structure Package we have added these new classes: "*DQ_Metadata*", "*DQ_Validator*" and "*DQConstraint*".

Bearing in mind the goal of modeling DQ Requirements, we have introduced these new elements with the idea of letting Web-Application users be aware of the level of the quality of the data that they are using for a task at hand.

In order to make our approach usable by any of the available IDEs (*Integrated Development Environment*) on the market, we have also implemented a UML profile for Web application requirements, which has been extended with data quality issues (*DQ_WebRE*).

The new stereotyped elements proposed in this UML profile are classified and defined according to the "Base class" that they belong. Each one of them is in charge of a specific task as described below.

The "*InformationCase*" and "*DQ_Requirement*" elements are specific use cases; the first one represents the use case in charge of managing the data involved with each Web functionality (e.g. *WebProcess*). While the second use case "*DQ_Requirement*", it is in charge to model the DQ requirements related to the "*InformationCase*" elements (see Fig. 2).

The "*Add_DQ_Metadata*" element to represent a particular activity, it is responsible of add the operations and information of the attributes belonging both to the elements "*DQ_Metadata*" as "*DQ_Validator*" (see Fig. 3).

The elements "*DQ_Metadata*", "*DQ_Validator*" y "*DQConstraint*" represent structural elements of the Web application. In the element "*DQ_Metadata*" will be stored the DQ metadata. The "*DQ_Validator*" element is in charge to manage the different operations related to the DQ, in order to validate the "*WebUI*" elements. By other way, in the "*DQConstraint*" elements will be stored the specific data to the specification of constraints (see Fig. 4).

Finally, the "*DQ_Req_Specification*" element will be used to the detailed specification of each one of the DQ requirements, through of requirements diagrams (see Fig. 5).

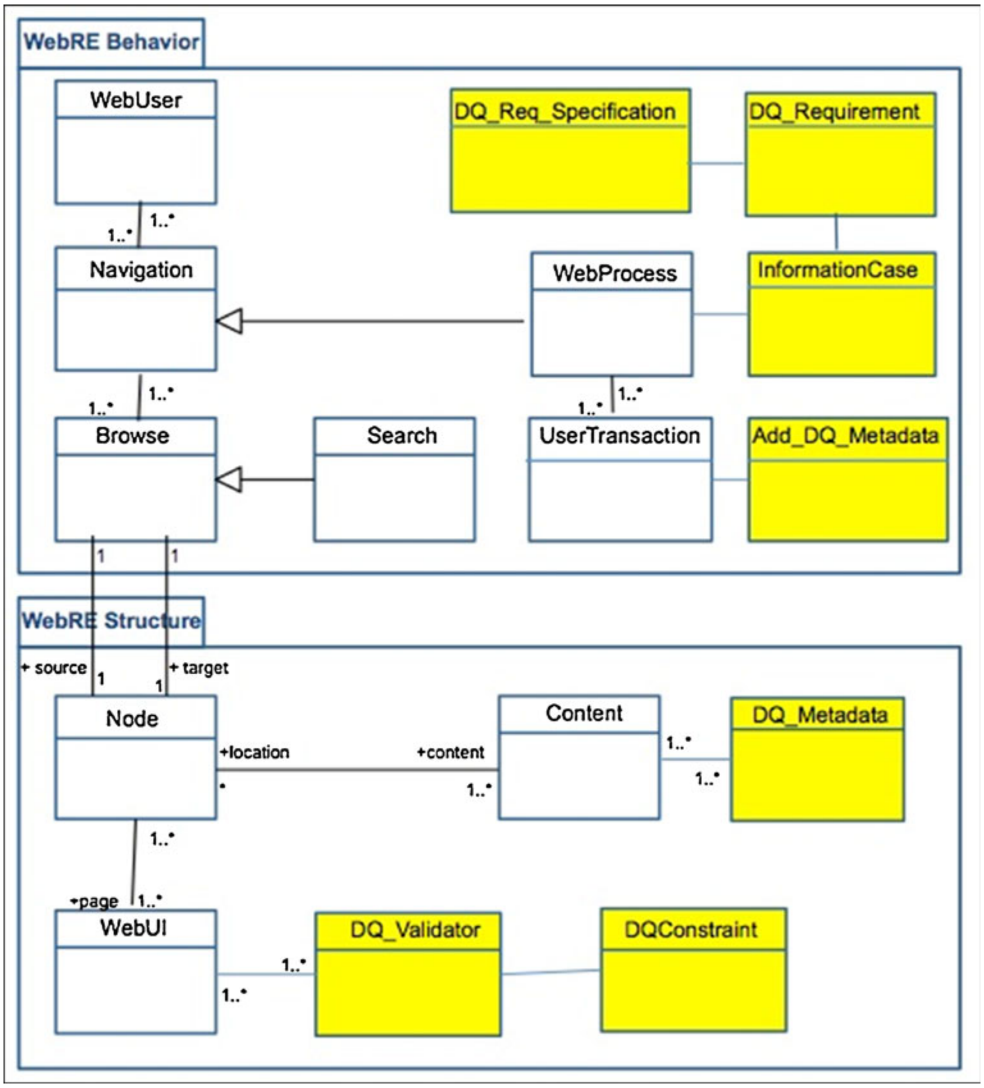


Fig. 1 Extended metamodel with DQ elements

The specification of each new stereotype is described in deep in Table 3. We have, to be precise, used the commercial tool Enterprise Architect to implement this new profile. Consequently, users can take advantage of the joint use of Enterprise Architect and *DQ_WebRE* to go ahead with the analysis by means of the corresponding diagrams. On the left-hand side of the tool (see Fig. 6) one can observe a special "*toolbox*" with its own elements defined in the *DQ_WebRE* profile.

4 Case study

In this section we shall demonstrate how to use our proposal by means of a case study. As developers, we are interested in highlighting how to capture the main functionalities of the system, as well as in how to complement such requirements with the main data quality requirements for the data that a user may need during the execution of the software developed.

Given that our proposal can be used with any of the software development methodologies, we selected the Unified Development Process (UDP) (Jacobson et al. 1999) for this case study because of its widespread use. In UDP, the Analysts will first model the system's principal use cases, and then, by means of activity diagrams, attempt to provide a more detailed description of each one of the use cases identified.

The case study we are going to present is based on the EasyChair Conference System (EasyChair 2012), one of the Web applications used most in supporting conference management. This application allows the following functionalities: paper submission, management and monitoring of Program Committee (PC) members, the assignment of papers to reviewers and some other features that are not included as part of this running example. It is worth mentioning that this application can be used by at least three different roles played by the user (e.g. Author, PC member and Chair). Each one will have their own DQRs for each of the functionalities.

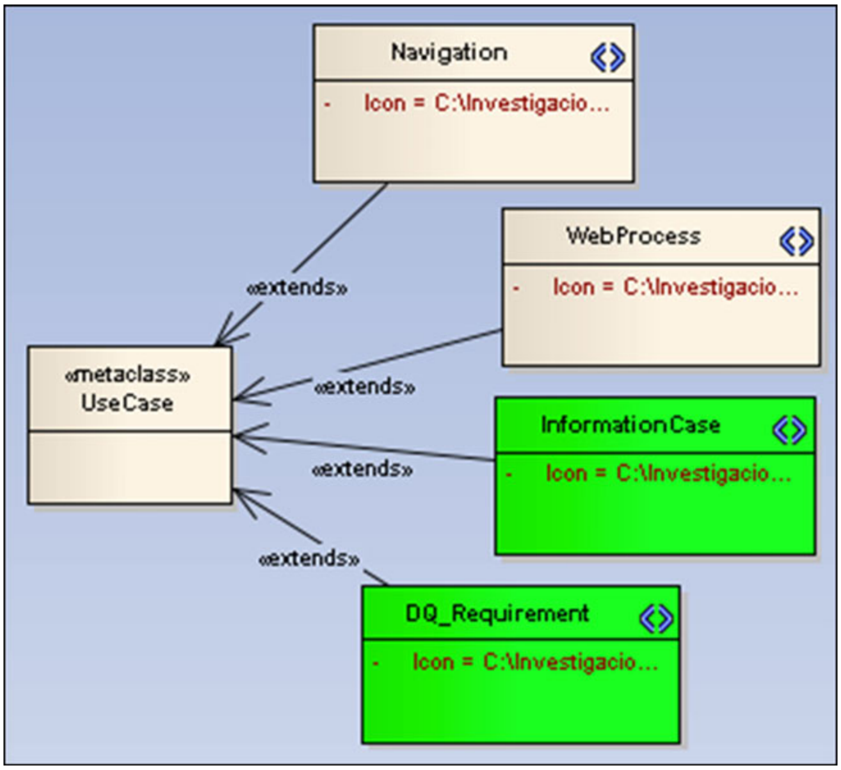

Navigation
- Icon = C:\Investigacio...
WebProcess
- Icon = C:\Investigacio...
InformationCase
- Icon = C:\Investigacio...
DQ_Requirement
- Icon = C:\Investigacio...
«metaclass»
UseCase
«extends»
«extends»
«extends»
«extends»

Fig. 2 New *Use cases* elements defined in “DQ_WebRE” profile

For the sake of simplicity, let us focus only on the use case “*Add new review to submission*” that a *PC member* requires of the EasyChair application (see Fig. 6).

Some data that will have to be used within this use case are: *first_name*, *last_name*, *email_address*, *overall_evaluation*, *reviewer_confidence*, etc. (see comment attached to

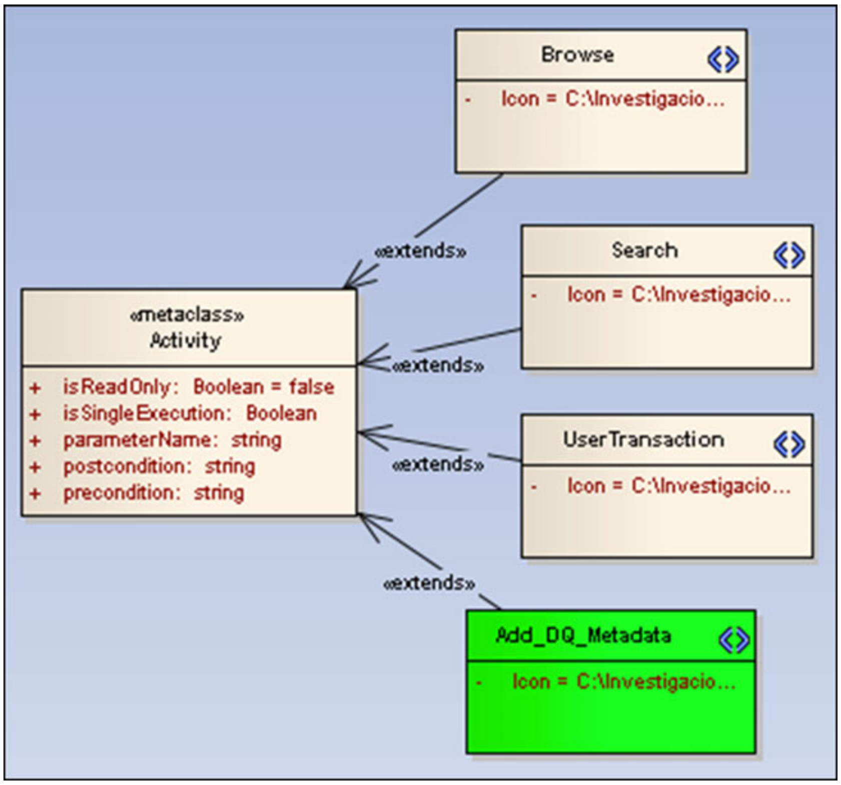

Browse
- Icon = C:\Investigacio...
Search
- Icon = C:\Investigacio...
UserTransaction
- Icon = C:\Investigacio...
Add_DQ_Metadata
- Icon = C:\Investigacio...
«metaclass»
Activity
+ isReadOnly: Boolean = false
+ isSingleExecution: Boolean
+ parameterName: string
+ postcondition: string
+ precondition: string
«extends»
«extends»
«extends»
«extends»

Fig. 3 New *Activity* element defined in “DQ_WebRE” profile

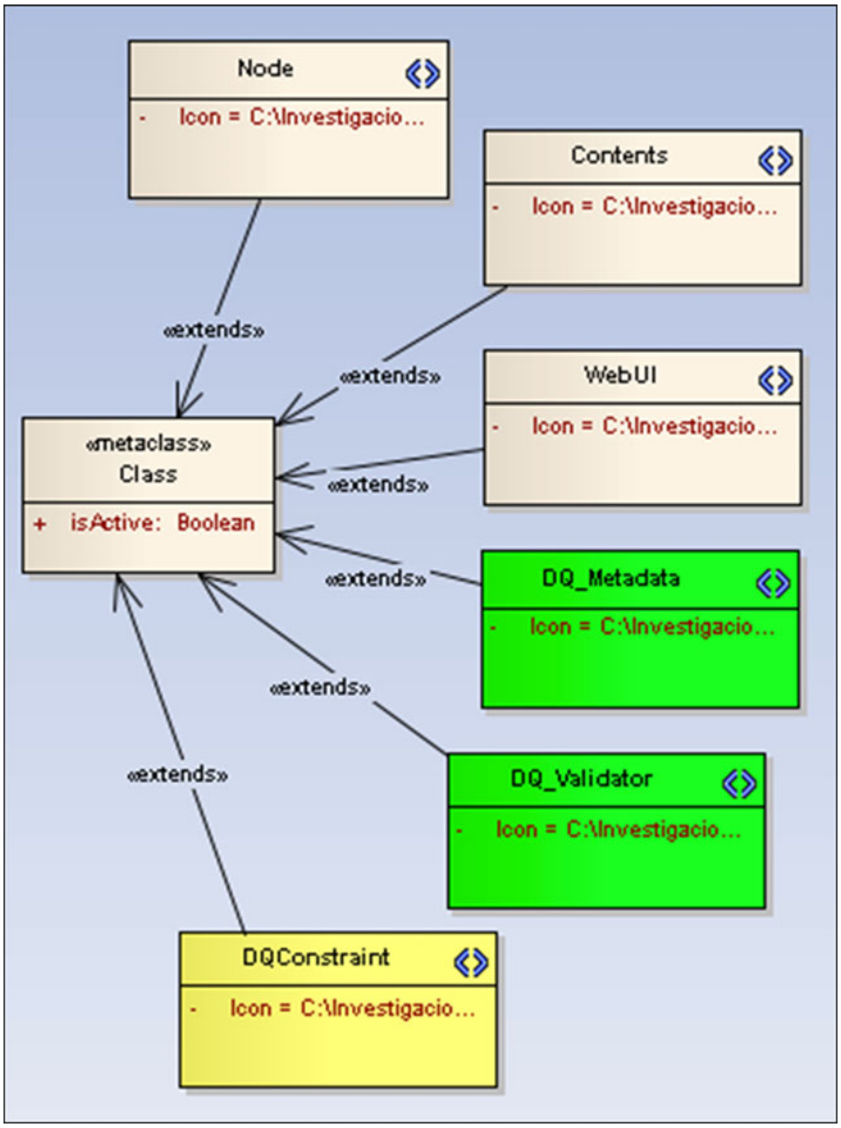

Node
- Icon = C:\Investigacio...
Contents
- Icon = C:\Investigacio...
WebUI
- Icon = C:\Investigacio...
«metaclass»
Class
+ isActive: Boolean
DQ_Metadata
- Icon = C:\Investigacio...
DQ_Validator
- Icon = C:\Investigacio...
DQConstraint
- Icon = C:\Investigacio...
«extends»

Fig. 4 New *Class* elements defined in "DQ_WebRE" profile

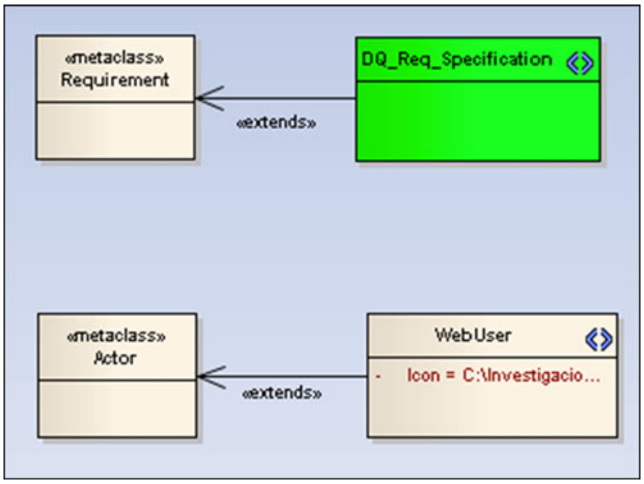


Fig. 5 New *Requirement* and *Actor* element defined in "DQ_WebRE" profile

classes "*information of reviewer"* and "*evaluation scores*" of type "*Contents*" in Fig. 7).

Once the data has been identified, the next step is to capture and introduce the data quality requirements. We can observe the specification of a particular use case "*Add all data as result of review*" (stereotyped as "*InformationCase*"), which will be in charge of managing all the data involved in the use case "*Add new review to submission*".

We would like the readers to be aware that we are interested in enabling the system to be responsible for the characteristics that the data will have if it is to assure the best levels of quality for the specified dimensions. This means that the analyst must meet new requirements here (probably functional ones) in the systems, so new functionalities that address data quality software requirements can be executed. The analysts, who are already provided with the *DQ_WebRE profile* to model these new requirements (see Fig. 6),

Table 3 Stereotype specification for DQ software requirements in DQ_WebRE profile

Name	Base class	Description	Constraints	Tagged values
InformationCase	UseCase	The IC, unlike normal use cases, has the main function of representing use cases that manage and store the data involved with the functionalities of the "*WebProcess*" type. These data will be subject to the specific requirements of data quality (*DQ_Requirement*) that are associated with them; we consider that the best way to link them is through a relationship of the "*include*" type, thus allowing them satisfy such DQ requirements.	Must be related to at least one element of "*WebProcess*" type.	None.
DQ_Requirement	UseCase	This represents a specific use case which is necessary to model the DQ requirements (*DQ dimensions*) that are related to the "*InformationCase*" use cases.	Must be related to ("*include*") at least one element of type "*Information Case*".	None.
DQ_Req_Specification	Element	Abstract class that represents a particular element ("*Requirement*" type). It will be used to specify each of the DQ requirements added through requirements diagrams in detail.		ID: Integer. Text: String.
Add_DQ_Metadata	Activity	This represents a particular activity which is related to the different "*UserTransaction*" activities. This metaclass is responsible for validating and adding the operations and information associated with each of the attributes (*DQ_metadata*) belonging to the "*DQ_Metadata*" or "*DQ_Validator*" metaclasses.	Not mandatory.	None.
DQ_Metadata	Class	This represents a structural element of a Web application, and the DQ metadata will be managed and stored here. These sets of metadata are associated with *Content* elements. It will thus be possible to specify various DQ requirements (*DQ dimensions*) directly linked to data stored in the elements of the "*Content*" type.	Not mandatory.	DQ_metadata: set(String)
DQ_Validator	Class	This represents a structural element. This metaclass will be responsible for managing different DQ operations in order to validate or restrict *WebUI* elements.	Not mandatory.	None.
DQConstraint	Class	This represents a structural element of a Web application. In this element are stored the specific data of the different constraints, which will be related to elements of type *DQ_Validator*. Besides its corresponding bounds (e.g. "*upper_bound*" and "*lower_bound*").	Must be related to at least one element of type "*DQ_Validator*".	DQConstraint: set (String). upper_bound: Integer. lower_bound: Integer

are then encouraged to define the corresponding functionalities to satisfy the perception of the data quality for each one of the application's users.

Having said this, it is possible to specify the following DQ requirements for the specified *InformationCase*:

1. If we wish to guarantee the *Confidentiality* of data, the execution of the next DQ functional requirement should be executed "*check that data will be accessed only by authorized users*". We are going to check if a user is authorized by means of a specific value for an attribute *Authorized*. This implies adding a new "Authorized" meta-attribute, as well as implementing the methods for capturing the value and for carrying out the checking. With this requirement, what is intended is to identify the piece of software that will be responsible for capturing some metadata in charge of ensuring that the information to be stored will only be accessed by users who meet a certain level of security defined previously in the application (e.g. *security level*).
2. With the objective of guaranteeing the *Completeness* of data, the next DQ functional requirement should be executed- "*verify that all data have been completed by reviewer*". In a similar way to that already outlined above, this requirement will be responsible for ensuring that all the data that will be entered by the reviewer are completed in every available field. The way of performing this verification will be through the specification of a particular function responsible for it (e.g. *check completeness*). This function will have to be implemented in a particular class specifically for management of DQ (e.g. "*DQ_Validator*").

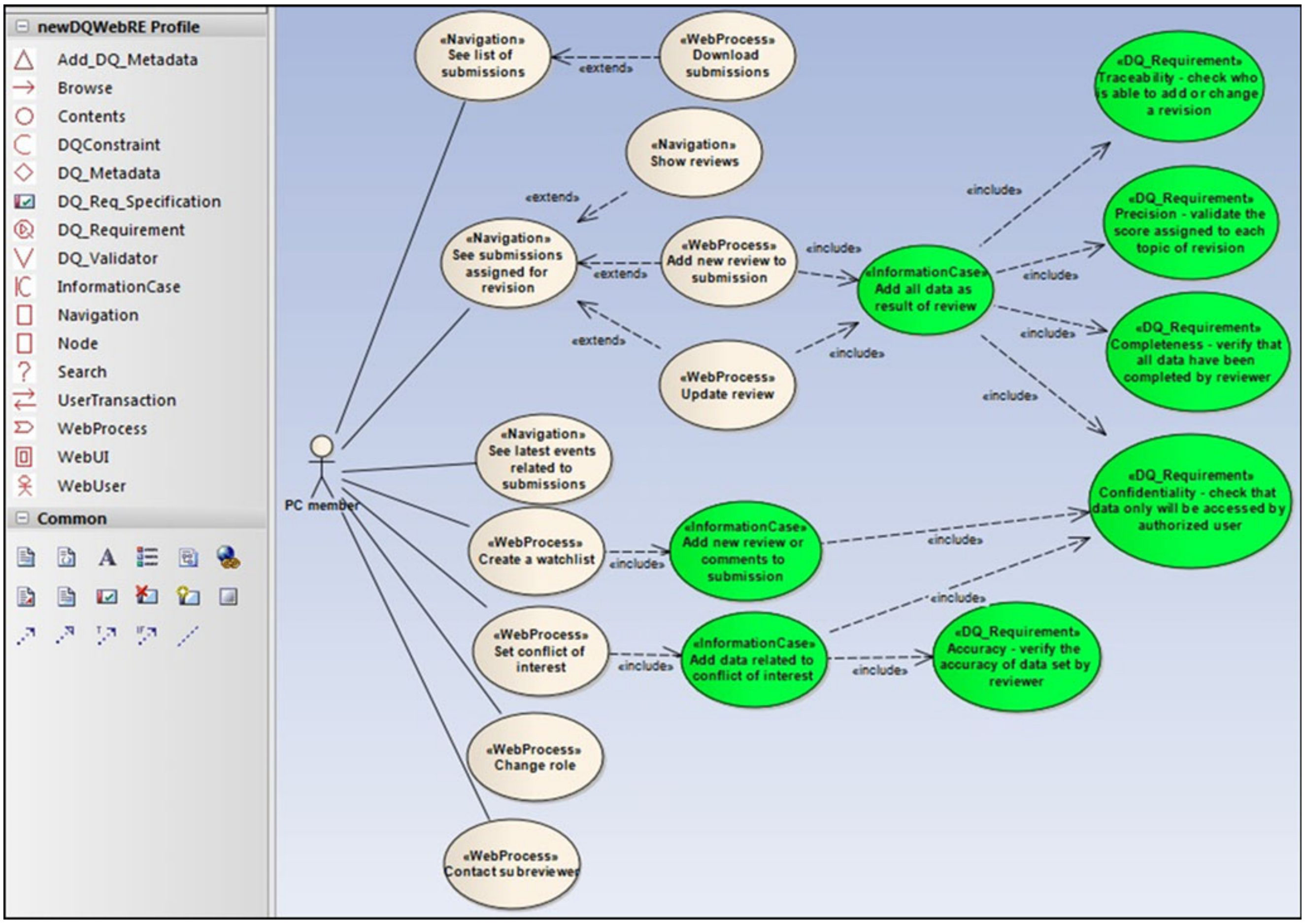

newDQWebRE Profile
Add_DQ_Metadata
Browse
Contents
DQConstraint
DQ_Metadata
DQ_Req_Specification
DQ_Requirement
DQ_Validator
InformationCase
Navigation
Node
Search
UserTransaction
WebProcess
WebUI
WebUser
Common
«Navigation» See list of submissions
«WebProcess» Download submissions
«extend»
«Navigation» Show reviews
«extend»
«Navigation» See submissions assigned for revision
«WebProcess» Add new review to submission
«extend»
«extend»
«WebProcess» Update review
«include»
«include»
«InformationCase» Add all data as result of review
«include»
«include»
«include»
«include»
«DQ_Requirement» Traceability - check who is able to add or change a revision
«DQ_Requirement» Precision - validate the score assigned to each topic of revision
«DQ_Requirement» Completeness - verify that all data have been completed by reviewer
«DQ_Requirement» Confidentiality - check that data only will be accessed by authorized user
PC member
«Navigation» See latest events related to submissions
«WebProcess» Create a watchlist
«include»
«InformationCase» Add new review or comments to submission
«include»
«include»
«WebProcess» Set conflict of interest
«include»
«InformationCase» Add data related to conflict of interest
«include»
«DQ_Requirement» Accuracy - verify the accuracy of data set by reviewer
«WebProcess» Change role
«WebProcess» Contact subreviewer

Fig. 6 Use case diagram specifying DQ requirements

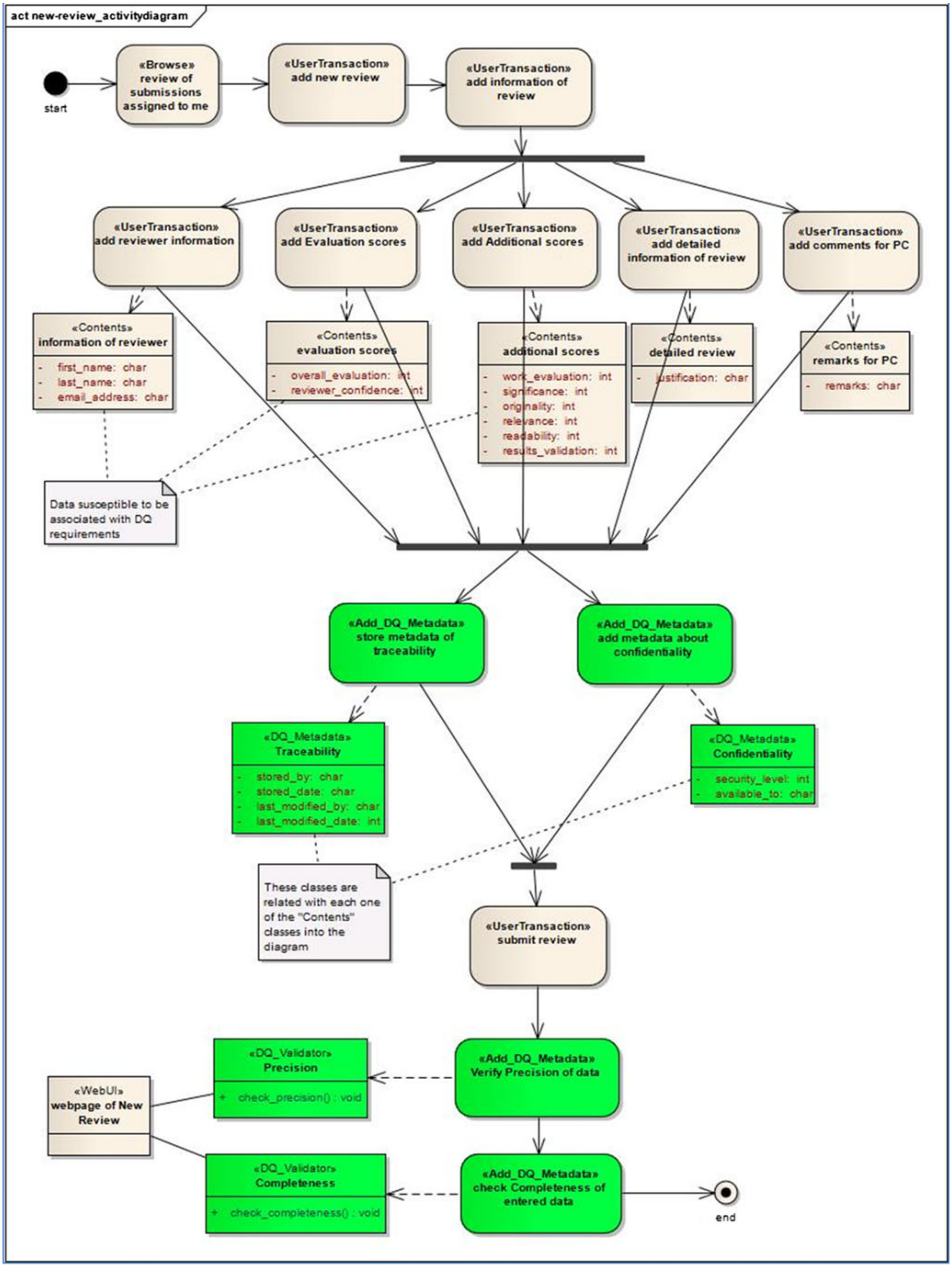
3. Similarly, the analyst will have to introduce the requirement that enables the IS to guarantee the characteristic of *Traceability* of data, through the specification of the following DQ functional requirement: "*check who is able to add or change a revision*". This traceability requirement will make the application responsible for adding the metadata whose purpose will be to keep records about who stored the data (e.g. *stored_by*, *last_modified_by*), as well as when it was stored the first time (*stored_date*) and modified the last time (*last_modified_date*). These metadata will be stored later in a specific class (stereotyped as "*DQ_Metadata*").
4. Likewise, the requirement of Precision: "*validate the score assigned to each topic of revision*"; this will be responsible for validating that all fields related to "*Evaluation scores*" fulfill this requirement, through implementing a specific function later. This function will be in a particular class which is specifically for management of DQ (stereotyped as "*DQ_Validator*").

Having completed the use case diagram, and aiming to give a well-detailed description of the use case "*Add new review to submission*", it is possible to draw an activity diagram, using the corresponding elements (stereotyped) defined in the *DQ_WebRE profile*, as shown in Fig. 7.

For this activity diagram (see Fig. 7), the analysts could model the specific activities to meet the specified DQ requirements; these activities will be related to different elements which are particular features of the development of a Web application.

In this activity diagram, the first of the activities, those named "*store metadata of traceability*" and "*add metadata about confidentiality*" (stereotyped as "*Add_DQ_Metadata*") will be responsible for capturing the metadata related to Traceability ("*stored_by*", "*stored_date*", "*last_modified_by*", "*last_modified_date*") and Confidentiality ("*security_level*", "*available_to*"). These metadata will be stored in an instance of the "*DQ_Metadata*" class and will be used later to satisfy the DQ requirement of *Traceability* and *Confidentiality,* respectively.

Fig. 7 Activity diagram with Data Quality management ▶


act new-review_activitydiagram
start
«Browse» review of submissions assigned to me
«UserTransaction» add new review
«UserTransaction» add information of review
«UserTransaction» add reviewer information
«UserTransaction» add Evaluation scores
«UserTransaction» add Additional scores
«UserTransaction» add detailed information of review
«UserTransaction» add comments for PC
«Contents» information of reviewer
- first_name: char
- last_name: char
- email_address: char
«Contents» evaluation scores
- overall_evaluation: int
- reviewer_confidence: int
«Contents» additional scores
- work_evaluation: int
- significance: int
- originality: int
- relevance: int
- readability: int
- results_validation: int
«Contents» detailed review
- justification: char
«Contents» remarks for PC
- remarks: char
Data susceptible to be associated with DQ requirements
«Add_DQ_Metadata» store metadata of traceability
«Add_DQ_Metadata» add metadata about confidentiality
«DQ_Metadata» Traceability
- stored_by: char
- stored_date: char
- last_modified_by: char
- last_modified_date: int
«DQ_Metadata» Confidentiality
- security_level: int
- available_to: char
These classes are related with each one of the "Contents" classes into the diagram
«UserTransaction» submit review
«DQ_Validator» Precision
+ check_precision() : void
«Add_DQ_Metadata» Verify Precision of data
«WebUI» webpage of New Review
«DQ_Validator» Completeness
+ check_completeness() : void
«Add_DQ_Metadata» check Completeness of entered data
end

Note that all these "*DQ_Metadata*" classes are related to the *data* managed in the previous activities of "*add reviewer information*", "*add evaluation scores*", "*add additional scores*", "*add detailed information of review*" and "*add comments for PC*" (stereotyped as "*UserTransaction*").

Finally, the activities "*Verify Precision of data*" and "*Check Completeness of entered data*" will enable the IS to be responsible for adding the specific operations of "*check_precision*()" and "*check_completeness*()", respectively (as part of the definition of the corresponding instance of a "*DQ_Validator*" class), in order to verify the Precision and Completeness of the data managed in the element "*webpage of New Review*" (stereotyped as "*WebUI*").

5 Conclusions and future work

Over the last decade, the amount and complexity of Web applications aimed at satisfying diverse business processes has grown dramatically. For the success of such software, it is of paramount importance that they be able to provide data with appropriate levels of quality. To do so, we posit that Web applications should be somehow customized, by introducing some features that allow us to take care about the quality of the data. This can be done by first of all capturing some kinds of DQR (Data Quality Requirements) that will be translated later into the corresponding DQSR (Data Quality Software Requirements) of the application.

Unfortunately, our conclusion has been that none of the existing Web development methodologies currently include issues that would address the management of DQ software requirements. An adequate management of DQ requirements would help developers to eliminate or at least to minimize the possible problems due to inadequate levels of quality in the data used.

To address these DQ concerns better, and taking the MDA approach (Bézivin 2004) as a basis, we have presented an extended metamodel and a UML profile (*DQ_WebRE*) with which to permit DQ software requirements to be captured in Web applications. The UML profile proposed will allow developers to introduce and model the key concepts of data quality from the initial stage of the development process, thus allowing developers to be aware of the DQ software requirements that need to be implemented for each one of the functionalities (use cases) that the Web application provides.

As part of our future work, and taking the MDA process as a guideline, we plan the incorporation of mechanisms focused on the design stage, in order to translate the DQ requirements into the corresponding design elements. We consider that an excellent choice would be to use transformation rules and implement them by employing the *QVT* (Query/View/Transformation) language (OMG 2008). We will thus be able to design models and produce code in a semiautomatic manner, with the eventual objective of developing Web applications more quickly and, in turn, ensuring the quality of the data that they manage.

Acknowledgments This work has been funded by the following projects: PEGASO/MAGO project (MICINN and FEDER, TIN2009-13718-C02-01), IQMNet (TIN2010-09809-E) project which are supported by the Spanish Ministerio de Educación y Ciencia. ENGLOBAS (PII2I09-0147-8235) and ARMONIAS (PII2I09-0223-7948) projects, both of which are supported by the Consejería de Educación y Ciencia of Junta de Comunidades de Castilla-La Mancha.

References

Ballou, D. P., & Pazer, H. L. (2003). Modeling completeness versus consistency tradeoffs in information decision contexts. *IEEE Transactions on Knowledge and Data Engineering, 15*(1), 240–243.

Batini, C., Barone, D., Mastrella, M., Maurino, A., & Ruffini, C. (2007). A Framework and a Methodology for Data Quality Assessment and Monitoring. In *12th International Conference on Information Quality, MIT, Cambridge, MA, November, 10–11.*

Batini, C., Cappiello, C., Francalanci, C., & Maurino, A. (2009). Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, *41*(*3*).

Becker, D., McMullen, W., & Hetherington-Young, K. (2007). a flexible and generic data quality metamodel. In *International Conference on Information Quality.*

Becker, D., Jaster, J., & Kuperman, J. (2009). Flexible and generic data quality metadata exchange. In *International Conference on Information Quality, ICIQ '09.*

Bertino, E., Maurino, A., & Scannapieco, M. (2010). *Guest editors' introduction: data quality in the internet Era*. pp. 11–13.

Bézivin, J. (2004). In search of a basic principle for model driven engineering. *UPGRADE, Novática, 2*(2), 21–24.

Caballero, I., Verbo, E. M., Calero, C., & Piattini, M. (2007). A data quality measurement information model based on ISO/IEC 15939. In *12th International Conference on Information Quality, MIT, Cambridge, MA, November, 10–11*

Caro, A., Calero, C., Caballero, I., & Piattini, M. (2008). A proposal for a set of attributes relevant for Web Portal Data Quality. *Software Quality Journal.*

Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks, 33*(1–6), 137–157.

De Castro, V., & Marcos, E. (2009). Towards a service-oriented MDA-based approach to the alignment of business process with IT Systems: from the business model to a web service composition model. *International Journal of Cooperative Information Systems, 18*(2), 225–260.

EasyChair EasyChair Conference System. http://www.easychair.org/. Accessed 14 December 2012.

Eppler, M., & Helfert, M. (2004). A classification and analysis of data quality costs. In *International Conference on Information Quality, MIT, Cambridge, MA, USA.* (pp. 311–325).

Escalona, M. J., & Aragón, G. (2008). NDT. A model-driven approach for web requirements. *IEEE Transactions on Software Engineering, 34*(3), 377–390. doi:10.1109/TSE.2008.27.

Escalona, M. J., & Koch, N. (2004). Requirements engineering for web applications: a comparative study. *Journal on Web Engineering, 2*, 193–212.

Escalona, M. J., & Koch, N. (2006). Metamodeling the Requirements of Web Systems. In S. B. Heidelberg (Ed.), *Web Information*

Systems and Technologies (Vol. Volume 1, pp. 267–280, Lecture Notes in Business Information Processing).

Fang, X., & Holsapple, C. W. (2011). Impacts of navigation structure, task complexity, and users' domain knowledge on Web site usability-an empirical study. *Information Systems Frontiers, 13* (4), 453–469.

Guerra-García, C., Caballero, I., & Piattini, M. (2009). DQ-VORD: A methodology for managing and integrating data quality requirements into software requirement specification. In *IADIS International Conference on WWW/INTERNET 2009, Rome, Italy, 19–22 November, 2009.* (pp. 392–399).

Guerra-García, C., Caballero, I., & Piattini, M. (2010). A systematic literature review of how to introduce data quality requirements into a software product development. In *5th. International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE, Athens, Greece, 22–24 July, 2010.* (pp. 12–19)

Guerra-García, C., Caballero, I., & Piattini, M. (2011). A survey on how to manage specific data quality requirements during information system development. *Lecture Notes in Computer Science* (Evaluation of Novel Approaches to Software Engineering), To be published.

ISO-25012 (2008). *ISO/IEC 25012: Software Engineering-Software product Quality Requirements and Evaluation (SQuaRE)-Data Quality Model*.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The unified software development process*. Reading: Addison-Wesley.

Janjua, N. K., Hussain, F. K., & Hussain, O. K. (2012). Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making. *Information Systems Frontiers*, 1–26.

Kahn, B. K., Strong, D. M., & Wang, R. Y. (2002). Information quality benchmarks: product and service performance. *Communications of the ACM, 45*(4ve), 184–192.

Karel, R., Moore, C., & Coit, C. (2009). Forrester's report for Business Process and Application Professionals on Trends 2009: Master Data Management. *Forrester.*

Koch, N., & Kraus, A. (2002). *The expressive power of UML-based web engineering.* Paper presented at the Second Int. Workshop on Web-oriented Software Technology (IWWOST '02), Málaga, Spain., June 2002.

Laudon, K. C. (1986). Data quality and due process in large interorganizational record system. *Communications of the ACM, 29*(1), 4–11.

Lee, Y. W., Pipino, L. L., Funk, J. D., & Wang, R. Y. (2006). *Journey to data quality.* Cambridge: Massachussets Institute of Technology.

Lucas, A. (2010). Corporate data quality management in context. In *15th International Conference on Information Quality, ICIQ 2010, Little Rock, Arkansas.*

Meliá, S., & Gómez, J. (2005). Applying Transformations to Model Driven Development of Web applications. In S. B. Heidelberg (Ed.), *Perspectives in Conceptual Modeling* (Vol. Volume 3770/ 2005, pp. 63–73, Lecture Notes in Computer Science).

OMG (2001). *Model Driven Architecture (MDA)- document number ormsc/2001-07-01*.

OMG (2005a). *OCL 2.0 Specification. Version 2.0*. (pp. 185): Object Management Group (OMG).

OMG (2005b). Unified Modeling Language: Superstructure. Versión 2.0. <http://www.omg.org/docs/formal/05-07-04.pdf>.

OMG (2008). MOF QVT Final Adopted Specification. http://www.omg.org/spec/QVT/1.0/ [Accessed in January, 2012].

Phan, D. D., & Vogel, D. R. (2010). A model of customer relationship management and business intelligence systems for catalogue and online retailers. *Information Management, 47*(2), 69–77. doi:10.1016/j.im.2009.09.001.

Pipino, L., Lee, Y., & Wang, R. (2002). Data quality assessment. *Communications of the ACM, 45*(4), 211–218.

Sarsfield, S. (2009). *The data governance imperative*: IT Governance Publishing.

Scannapieco, M., & Berti-Équille, L. (2006). Report from the First and Second International Workshops on Information Quality in Information Systems- IQIS 2004 and IQIS 2005 in Conjunction with ACM SIGMOD/PODS Conferences. *SIGMOD RECORD, 35*(2), 50–52.

Shankaranayanan, G., & Cai, Y. (2005). A web services application for the data quality management in the B2B networked environment. In *38th Hawaii International Conference on System Sciences (HICSS-38 2005), Big Island, HI, USA, 3–6 January 2005*: IEEE Computer Society.

Strong, D., Lee, Y., & Wang, R. (1997). Ten potholes in the road to information quality. *IEEE Computer*, 38–46.

Strong, D. M., Lee, Y. W., & Wang, R. Y. (1997b). Data quality in context. *Communications of the ACM, 40*(5), 103–110.

Wang, R., Storey, V., & Firth, C. (1995). A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering, 7*(4).

César Guerra-García is associate professor at the Polytechnic University of San Luis Potosí. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software and data quality, requirements engineering and web engineering.

Ismael Caballero is associate professor at the University of Castilla-La Mancha and belongs to the Alarcos Research Group at the UCLM. He holds the PhD degree in Computer Science from the University of Castilla-La Mancha. His research interests include software and data quality, database design and software development based on quality.

Mario Piattini is full professor at the UCLM. His research interests include software quality, metrics and maintenance. He holds the PhD degree in Computer Science from the Technical University of Madrid, and leads the Alarcos Research Group at the Universidad de Castilla-La Mancha. He is CISA, CISM e CGEIT by ISACA.