Capturing Data Quality Requirements for Web Applications by means of DQ_WebRE

César Guerra-García
Department of Information Technologies, UPSLP.
Urbano Villalón 500
78363, San Luis Potosí, México
Cesar.Guerra@upslp.edu.mx

Ismael Caballero
Department of Information Technologies and Systems, UCLM.
Paseo de la Universidad 4
13071, Ciudad Real, Spain
Ismael.Caballero@uclm.es

Mario Piattini
Department of Information Technologies and Systems, UCLM.
Paseo de la Universidad 4
13071, Ciudad Real, Spain
Mario.Piattini@uclm.es

ABSTRACT

The number and complexity of Web applications which are part of Business Intelligence (BI) applications had grown exponentially in recent years. The amount of data used in these applications has consequently also grown. Managing data with an acceptable level of quality is paramount to success in any organizational business process. In order to raise and maintain the adequate levels of Data Quality (DQ) it is indispensable for Web applications to be able to satisfy specific DQ requirements. In order to achieve this goal, DQ requirements should be captured and introduced into the development process together with the other software requirements needed in the applications. However, in the field of Web application development, and to the best of our knowledge, no proposals exist with regard to the way in which to manage specific DQ software requirements. This paper considers the MDA (*Model Driven Architecture*) approach and, principally, the benefits provided by Model Driven Web Engineering (*MDWE*) in order to put forward a proposal for two artifacts. These two artifacts are a metamodel and a UML profile for the management of Data Quality Software Requirements for Web Applications (*DQ_WebRE*).

Keywords

Data Quality, Web Engineering, Requirements Engineering, Model Driven Web Engineering, Requirements Modeling.

1. INTRODUCTION

Many companies currently manage a large amount of their business intelligence data through Web applications. The use of these applications has created new ways for enterprises to benefit from the vast potential of client relationships, which has never previously been exploited [1]. However, problems caused by inadequate levels of quality in the data which flows through these Web applications arise more commonly than expected [2, 3]. Batini et al. in [4] 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, and consequently the organization’s performance.

It can be proven that these problems provoke different kinds of damage within organizations [5-9]. This damage is transformed into higher and higher costs in both economical and social terms [10-12], but it is only when organizations become aware of the situation that they are willing to eradicate this kind of problems.

As a first possible solution, organization consider the adoption of specific Data Quality Software (e.g. data cleansing, standardization, matching, merging, enrichment and data profiling), as proposed in [13]. 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 continuously *living* [14]. In addition, this solution is not focused on specific users´ data quality requirements. This implies that some kind of customization of the Information System aimed at preventing DQ problems is necessary.

We shall commence by briefly describing the concept of data quality. The most widely accepted definition of the term Data Quality is based on Deming’s “*fitness for use*” [15]. This signifies that a user can only assess the level of quality of a set of data for a particular task to be executed in a specific context, according to a set of criteria, thus determining whether or not these data can be used for that purpose [16]. It is essential to point out here that this set of criteria is typically denominated as a DQ Model. A DQ Model is composed of several DQ dimensions or characteristics. A user would therefore desire that a set of data would comply with the requirements specified by a DQ Model, namely with a DQ Requirement.

In order to obtain a better understanding of the concept of “*DQ requirement*”, we decided to use the definition coined in [14]: “*the specification of a set of dimensions or characteristics of DQ that a set of data should meet for a specific task performed by a determined user*”. Our objective is to help to identify those DQ Requirements which will be translated into specific DQ software requirements. The latter will be introduced in the earliest stages of Web application development, thus complementing the Software Requirements Specification. The researching question was how to introduce the specific DQ requirements into the development of Web applications. In order to seek an answer to this question, we have deliberated the possibility of considering the paradigm of Model Driven Web Engineering (*MDWE*) as a suitable layout to support the solution. MDWE proposes representing concepts by supporting the development process by means of a set of models, transformations and relations between models, which leads to agile developments and assures consistency between models [17].

Much research in MDWE is principally concerned 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. Moreover, as previously stated, there are no works that partially cover the various corresponding issues related to the management of DQ software requirements at the moment of modeling and developing Web applications, as is concluded in [14]. 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, and in a more specific sense, in the initial requirements specification phase [18]. More precisely, the main contribution of this work towards both the area of Requirements Engineering and to MDWE is the proposal of an extended metamodel and a UML profile which will allow developers to incorporate aspects of DQ software requirements. These artifacts will permit DQ issues to be introduced into the various diagrams (use case and activity), thus collaborating in the design of *Data Quality-aware Web applications*.

The remainder of the paper is structured as follows: Section Two provides a brief description of the model's foundations on DQ, on Web Engineering and on 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 Three. Section Four shows an illustrated example using the *DQ_WebRE* profile, and finally, some conclusions and future work are presented in Section Five.

2. RELATED WORK

2.1 Data Quality

Various definitions of the concept of Data Quality exist [19]. However, most authors agree 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 an specific context [16]. 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* [20].

Amongst the various data quality models considered as standards, we have considered that of ISO/IEC 25012 [21] for our research, since it is the "de jure" model. It provides a Data Quality model for data managed in information systems, and considers fifteen dimensions or characteristics which are grouped 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 of the data quality dimensions proposed by the ISO/IEC 25012 standard [21]. It is worth mentioning that these dimensions or characteristics should be reinterpreted and redefined to better represent how to measure the level of data quality of a piece of data in a context. These dimensions or characteristics should be considered when specifying a data quality requirement. More importantly, a reinterpretation of the definition provided in Table 1 must be carried out in order to customize the idea of data quality to the user's perception of it.

Table 1. Data Quality dimensions proposed by ISO/IEC 25012 standard.

Dimension	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.

2.2 Web Engineering and MDWE

We analyzed different methodologies supporting requirements analysis and design phases and found the following proposals: NDT [17], UWE [22], WebML [23], WebRE [24], and WebSA [25]. A comparative study of these methodologies is shown in [26]. This last study principally shows the types of requirements managed by each proposal, along with the techniques used and the extent of detail of each proposal in terms of their development process.

All of these methodologies are principally focused on how to identify and define the functional aspects, related to the semantics of models, oriented towards capturing the relevant properties of this type of Web applications. However, none of these proposals includes the quality characteristics of the data that is managed and stored by these applications. Only a few of the proposals, such as those in [17, 23, 24], mention certain specific information objectives that should be considered when designing a Web application. However, they neither explore their study in greater depth, nor do they consider any requirements or specifications of DQ characteristics.

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 [24]. *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* [27], *UML* [28], *OCL* [29], *QVT* [30].

The metamodel proposed by Escalona and Koch in [24] permits the principal 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 standard UML using stereotypes and constraints [24].

One of the main advantages of this metamodel is that it is very flexible: it allows the easy inclusion of new elements. It will thus enable us to add new elements with which to manage the DQ, in order to specify and model the DQ software requirements in a particular way, and to relate these new DQ elements to each element listed in the profile, e.g. use cases ("*WebProcess*"), or specific activities like "*UserTransaction*".

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.
UserTransac-tion	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 METAMODEL AND PROFILE OF DQ SOFTWARE REQUIREMENTS FOR WEB APPLICATIONS

After carrying out an in-depth analysis of the different Web Engineering proposals, and given their features, we decided to take that proposed by Escalona and Koch [24] as a basis for our work, since it satisfies one of the key requirements for our research: its compatibility with "de jure" standards. 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*) [24].

Having shown the main characteristics and elements of the *WebRE* metamodel in Section 2.2, in this section we describe our proposal. One of the most important motivations of this work is to provide the analysts and designers of Web applications with the artifacts needed to specify and describe certain DQ software requirements in a clear and intuitive manner.

We therefore intend to extend Escalona and Koch´s metamodel for the integration of those elements which are considered to be essential for the specification of DQ software requirements. Having conducted a systematic review on the main proposals for the specification and modeling of DQ requirements, presented in [31], we decided to incorporate the following key elements (namely *stereotype)* which were principally inferred from [32-34] (see Figure 1):

- For the Behavior Package: "*InformationCase*", "*DQ_Requirement*", "*DQ_Req_Specification*" and "*Add_DQ_Metadata*";
- For the Structure Package: "*DQ_Metadata*" and "*DQ_Validator*".

Bearing the objective of modeling DQ Requirements in mind, we have introduced these new elements, which allow what a user may require to control the level of quality of the data used in a Web Application to be modeled. In order to make our approach operative, we have also implemented a UML profile for Web application requirements which has been extended with data quality issues (*DQ_WebRE*) (see Figure 2). We have used the commercial tool Enterprise Architect to implement the new profile and to later manage the corresponding diagrams. On the left-hand side of the tool (see Figure 3) we can observe a special "*toolbox*" with its own elements defined in the *DQ_WebRE* profile. The specification of each new stereotype is described in Table 3.

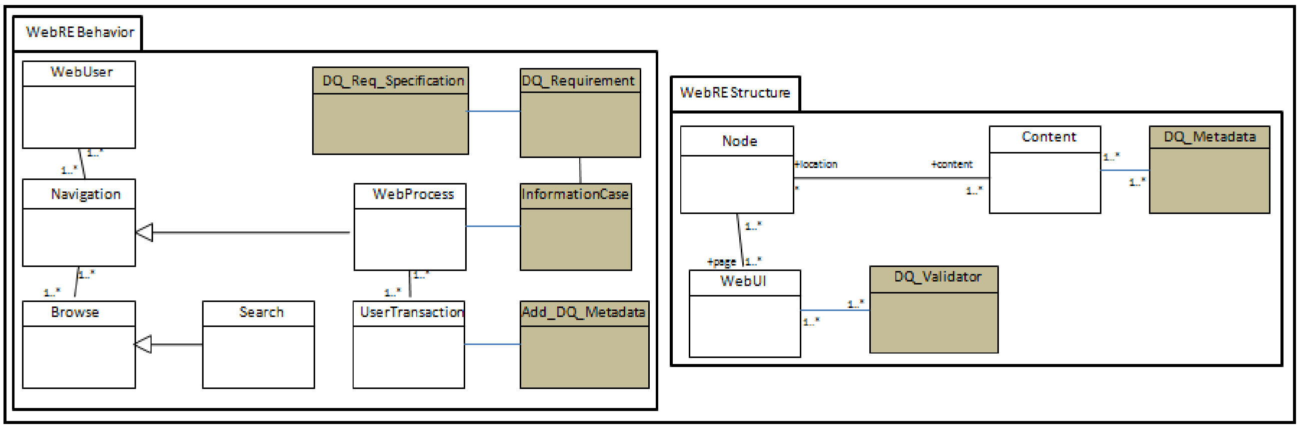

Figure 1. Extended metamodel with DQ elements.

Table 3. Stereotypes 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.

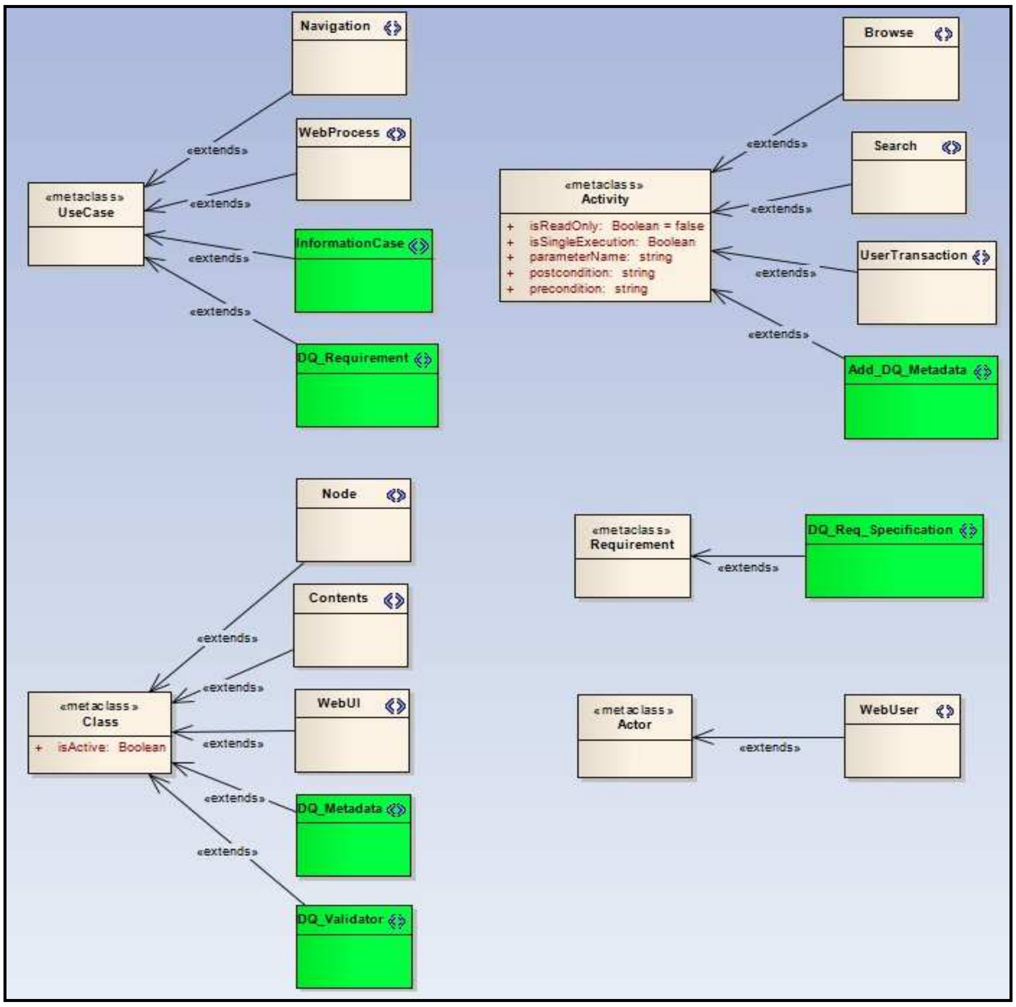

Figure 2. DQ_WebRE profile.

4. EXAMPLE OF APPLICATION

In this section we shall demonstrate how to use our proposal by means of an example. As developers, we are interested in highlighting how to capture the main functionalities of the system, in addition to the principal data quality requirements for the data used in the execution of the software that will implement the desired functionalities.

Bearing in mind the focus of the Unified Development Process (UDP) [35], 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 use case identified.

The example describes a typical business process for the reservation and payment of tickets for particular concerts and events, using a Web application. The flow of events is described as follows: The client will first be able to browse the available events; however, in order to make a reservation, s/he must be registered and log in into the system. Once logged in, s/he can select the event and verify its availability and cost. If the client agrees with these data, s/he can proceed by entering specific data to make the reservation. S/he must then provide payment for the ticket, and the system will send him/her the electronic ticket by email. To satisfy this functional requirement, the analysts introduce the use case "*Make ticket reservation*". Some data that will be used in this use case are: *Invoice_number*, *ID_client*, *Client*, *Address*, *Cost*, etc. (see comment attached to class *Invoice* and *Reservations* in Figure 4).

Once the data has been identified, the next step is to capture and introduce the data quality requirements. It is known that if the specific functionality defined by this use case is to succeed, the data used must be reliable, accurate, complete and confidential.

In our context, a piece of data can be said to be credible when it has been provided by an authorized user. At this point, we do not wish to argue whether a user is or is not authorized. Let us simply suppose that we have a database containing solely authorized users. We must model a query to this database, but must also bear in mind that the intention is to warranty a specific database.

We would like to make readers aware that we are interested in enabling the system to be responsible for warranting that data which will have the best levels of quality for the specified dimensions. This signifies that the analyst must introduce the new requirements (probably functional requirements) into the systems with the objective of executing new functionalities in order to obtain this warranty. The analysts are therefore provided with the *DQ_WebRE* profile to model these new requirements (see Figure 3). They are then encouraged to define the corresponding functionalities to satisfy the perception of the data quality for each of the application' users.

Returning to our example, if we wish to verify the credibility of the data, the execution of a <<*DQ_Requirement*>> "[Credibility] *Check if data have been provided by an authorized user*" must be executed.

In addition, in order to ensure the level of accuracy of a piece of data, the use case named "<<DQ_Requirement>> [Accuracy] *Check if the incoming data satisfy a specific standard format*" should be executed to ensure that the data managed fulfills this DQ requirement.

The Analysts will similarly be able to guarantee the level of confidentiality of data, through the use case named "<<DQ_Requirement>> [Confidentiality] *Check data is only shown to an allowed user*". Finally, the analyst will be able to guarantee this DQ requirement by means of a use case named "<<DQ_Requirement>> [Completeness] *Check that all data introduced is complete*".

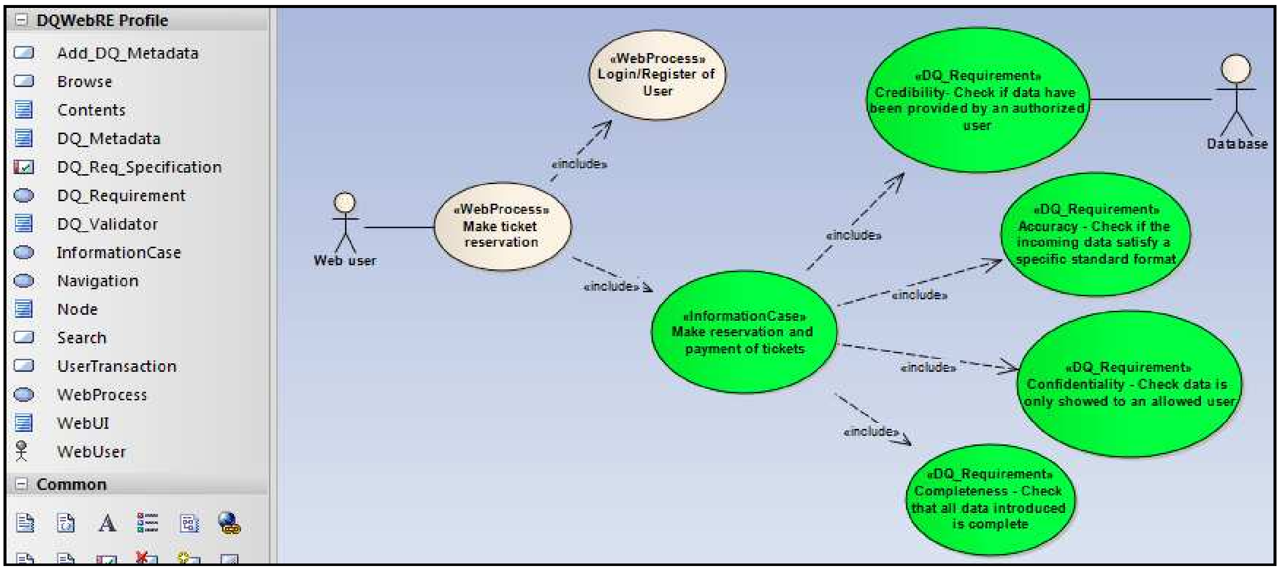
DQWebRE Profile
Add_DQ_Metadata
Browse
Contents
DQ_Metadata
DQ_Req_Specification
DQ_Requirement
DQ_Validator
InformationCase
Navigation
Node
Search
UserTransaction
WebProcess
WebUI
WebUser
Common
«WebProcess» Login/Register of User
«include»
Web user
«WebProcess» Make ticket reservation
«include»
«DQ_Requirement» Credibility- Check if data have been provided by an authorized user
Database
«include»
«DQ_Requirement» Accuracy - Check if the incoming data satisfy a specific standard format
«include»
«InformationCase» Make reservation and payment of tickets
«include»
«DQ_Requirement» Confidentiality - Check data is only showed to an allowed user
«include»
«DQ_Requirement» Completeness - Check that all data introduced is complete

Figure 3. Use cases diagram specifying DQ requirements.

It is also possible to model the corresponding activity diagram by making use of the stereotyped elements defined in the new profile (see Figure 4). In this figure, the diagram shows the main activities carried out in order to describe the "*Make ticket reservation*" use case.

In this activity diagram (Figure 4), the Analysts will be able to model the specific activities to meet the DQ software requirements, and these activities will be related to different elements which are specific to the development of a Web application. These specific DQ activities are derived from the DQ software requirements that each user defines for the data that will be managed in each *InformationCase*.

In this example, the "*Verify and add Confidentiality metadata*" activity will verify and add Confidentiality metadata ("*Available_to*" and "*Security_level*"). These metadata will be stored in an instance of the "*DQ_Metadata*" class, and thus fulfill the DQ requirement of *Confidentiality*.

The "*Verify Accuracy of data*" activity will be responsible for adding the specific operations (as part of the definition of the corresponding instance of a "*DQ_Validator*" class specifically aimed at satisfying this requirement) in order to verify the Accuracy of the data managed in the "*Webpage of reservations*" element (of *WebUI* type). The "*Verify Completeness of data*" activity will similarly be in charge of adding the specific functions in order to verify the Completeness of the data managed in each element that appears in the "*Webpage of payment*" of *WebUI* type.

Finally, the "*Verify Credibility of data*" activity will be responsible for managing and adding the DQ metadata ("*Client_valid*" and "*Card_valid*") stored in an instance of the "*DQ_Metadata*" class, in order to guarantee the DQ requirement of *Credibility*, and will be related to the Invoice data (of "*Content*" type).

5. CONCLUSIONS AND FUTURE WORK

In the last decade, the amount and complexity of Web applications whose aim is to satisfy diverse business processes has grown dramatically. It is paramount that these web applications will be able to provide data with appropriate quality levels. In order to achieve this goal, we consider that Web applications should implement certain kinds of artifacts to provide them with data quality awareness. This could be achieved by capturing some kinds of data quality requirements that will be later translated into the software requirements of the application. Unfortunately, none of the existing Web development methodologies include the management of DQ software requirements. A correct management of DQ requirements would help developers to anticipate the needs of users who require data for their tasks, in order to eliminate or at least minimize the possible problems caused by inadequate levels of quality in the data used. When executing software, users would therefore benefit from a higher level of trust in their tasks and processes, both internal (within the same organization) and external (business processes) with other companies and clients.

In order to solve these DQ problems, and taking the MDA approach [36] as a basis, we present 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 us 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 functionality (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 option would be to use transformation rules and implement them by employing the *QVT* (Query/View/Transformation) language [30]. 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.

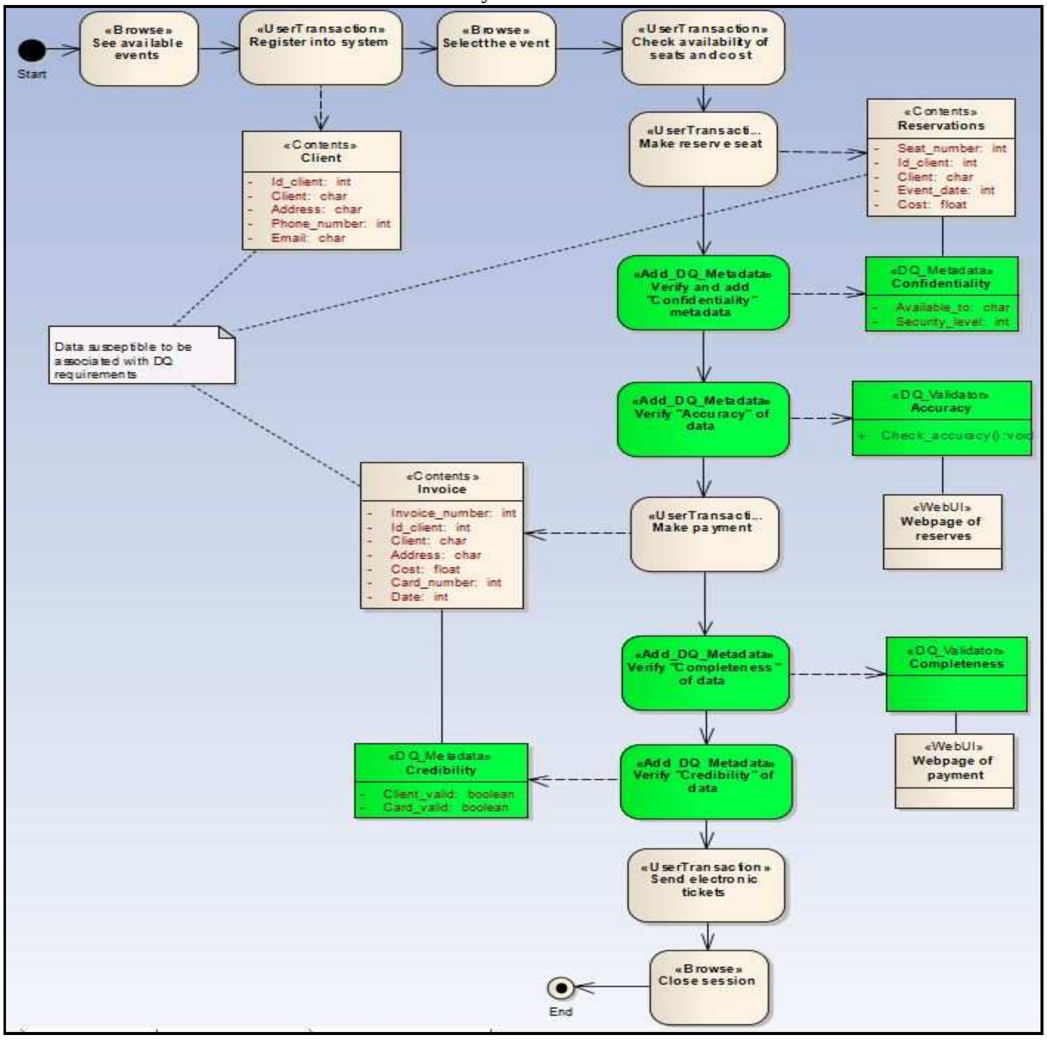
Start
«Browse» See available events
«UserTransaction» Register into system
«Browse» Selectthe event
«UserTransaction» Check availability of seats and cost
«Contents» Client
- Id_client: int
- Client: char
- Address: char
- Phone_number: int
- Email: char
«UserTransacti... Make reserve seat
«Contents» Reservations
- Seat_number: int
- Id_client: int
- Client: char
- Event_date: int
- Cost: float
«Add_DQ_Metadata» Verify and add "Confidentiality" metadata
«DQ_Metadata» Confidentiality
- Available_to: char
- Security_level: int
Data susceptible to be associated with DQ requirements
«Add_DQ_Metadata» Verify "Accuracy" of data
«DQ_Validator» Accuracy
+ Check_accuracy() :void
«Contents» Invoice
- Invoice_number: int
- Id_client: int
- Client: char
- Address: char
- Cost: float
- Card_number: int
- Date: int
«UserTransacti... Make payment
«WebUI» Webpage of reserves
«Add_DQ_Metadata» Verify "Completeness" of data
«DQ_Validator» Completeness
«WebUI» Webpage of payment
«DQ_Metadata» Credibility
- Client_valid: boolean
- Card_valid: boolean
«Add_DQ_Metadata» Verify "Credibility" of data
«UserTransaction» Send electronic tickets
«Browse» Close session
End

Figure 4. Activity diagram with DQ management.

6. ACKNOWLEDMENTS

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 supported by the Consejería de Educación y Ciencia of Junta de Comunidades de Castilla-La Mancha.

7. REFERENCES

[1] Phan, D.D. and D.R. Vogel, *A model of customer relationship management and business intelligence systems for catalogue and online retailers.* Information & Management, 2010. 47(2): p. 69-77.

[2] Bertino, E., A. Maurino, and M. Scannapieco, *Guest Editors' Introduction: Data Quality in the Internet Era*. 2010. p. 11-13.

[3] Caro, A., et al., *A proposal for a set of attributes relevant for Web Portal Data Quality.* Software Quality Journal, 2008.

[4] Batini, C., et al. *A Framework and a Methodology for Data Quality Assessment and Monitoring*. in *12th International Conference on Information Quality*. 2007. MIT, Cambridge, MA.

[5] Ballou, D.P. and H.L. Pazer, *Modeling Completeness versus Consistency Tradeoffs in Information Decision Contexts* IEEE Transactions on Knowledge and Data Engineering 2003 15 (1): p. 240-243

[6] Kahn, B.K., D.M. Strong, and R.Y. Wang, *Information Quality Benchmarks: Product and Service Performance.* Communications of the ACM, 2002. **45**(4ve): p. 184-192.

[7] Pipino, L., Y. Lee, and R. Wang, *Data Quality Assessment.* Communications of the ACM, 2002. **45**(4): p. 211-218.

[8] Scannapieco, M. and L. Berti-Équille, *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, 2006. 35(2): p. 50-52.

[9] Shankaranayanan, G. and Y. Cai. *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)*. 2005. Big Island, HI, USA: IEEE Computer Society.

[10] Eppler, M. and M. Helfert. *A Classification and Analysis of Data Quality Costs*. in *International Conference on Information Quality*. 2004. MIT, Cambridge, MA, USA.

[11] Laudon, K.C., *Data Quality and Due Process in Large Interorganizational Record System.* Communications of the ACM, 1986. 29(1): p. 4-11.

[12] Wang, R., V. Storey, and C. Firth, *A Framework for Analysis of Data Quality Research.* IEEE Transactions on Knowledge and Data Engineering, 1995. **7**(4).

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

[14] Guerra-García, C., I. Caballero, and M. Piattini, *A Survey on How to Manage Specific Data Quality Requirements during Information System Development.* Lecture Notes in Computer Science, 2011(Evaluation of Novel Approaches to Software Engineering).

[15] Ge, M. and M. Helfert. *A Review of Information Quality Research*. in *International Conference on Information Quality*. 2007. MIT, Cambridge, MA, USA.

[16] Strong, D.M., Y.W. Lee, and R.Y. Wang, *Data Quality in Context.* Communications of the ACM, 1997. 40(5): p. 103-110.

[17] Escalona, M.J. and G. Aragón, *NDT. A Model-Driven Approach for Web Requirements.* IEEE Trans. Softw. Eng., 2008. 34(3): p. 377-390.

[18] Guerra-García, C., I. Caballero, and M. Piattini. *DQ-VORD: A Methodology for Managing and Integrating Data Quality Requirements into Software Requirement Specification*. in *IADIS International Conference WWW/INTERNET 2009*. 2009. Rome, Italy.

[19] Batini, C., et al., *Methodologies for data quality assessment and improvement.* ACM Computing Surveys, 2009. Vol. 41, No. 3.

[20] Lee, Y.W., et al., *Journey to Data Quality*. 2006, Cambridge, MA, USA: Massachussets Institute of Technology.

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

[22] Koch, N. and A. Kraus, *The Expressive Power of UML-based Web Engineering*, in *Second Int. Workshop on Web-oriented Software Technology (IWWOST ´02)*. 2002: Málaga, Spain. p. 105-119.

[23] Ceri, S., P. Fraternali, and A. Bongio, *Web Modeling Language (WebML): a modeling language for designing Web sites.* Computer Networks, 2000. 33(1-6): p. 137-157.

[24] Escalona, M.J. and N. Koch, *Metamodeling the Requirements of Web Systems*, in *Web Information Systems and Technologies*, S.B. Heidelberg, Editor. 2006. p. 267-280.

[25] Meliá, S. and J. Gómez, *Applying Transformations to Model Driven Development of Web applications*, in *Perspectives in Conceptual Modeling*, S.B. Heidelberg, Editor. 2005. p. 63-73.

[26] Escalona, M.J. and N. Koch, *Requirements Engineering for Web Applications: A Comparative Study.* Journal on Web Engineering, 2004. 2: p. 193-212.

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

[28] OMG. *Unified Modeling Language: Superstructure. Versión 2.0*. 2005; Available from: <http://www.omg.org/docs/formal/05-07-04.pdf>.

[29] OMG, *OCL 2.0 Specification. Version 2.0.* 2005, Object Management Group (OMG). p. 185.

[30] OMG. *MOF QVT Final Adopted Specification*. 2008; Available from: http://www.omg.org/spec/QVT/1.0/ [Accessed in January, 2011].

[31] Guerra-García, C., I. Caballero, and M. Piattini. *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*. 2010. Athens, Greece.

[32] Becker, D., J. Jaster, and J. Kuperman. *Flexible and Generic Data Quality Metadata Exchange*. in *International Conference on Information Quality, ICIQ ´09*. 2009.

[33] Becker, D., W. McMullen, and K. Hetherington-Young. *A Flexible and Generic Data Quality Metamodel*. in *International Conference on Information Quality*. 2007.

[34] Caballero, I., et al. *A Data Quality Measurement Information Model based on ISO/IEC 15939*. in *12th International Conference on Information Quality*. 2007. MIT, Cambridge, MA.

[35] Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process.* 1999: Reading (MA): Addison-Wesley.

[36] Bézivin, J., *In Search of a Basic Principle for Model Driven Engineering.* UPGRADE, Novática., 2004. Vol. 2(No.2): p. 21-24.