DAQ_UWE: A FRAMEWORK FOR DESIGNING DATA QUALITY AWARE WEB APPLICATIONS

(RESEARCH IN PROGRESS)

César Guerra-García
Polytechnic University of San Luis Potosí, México
cesar.guerra@upslp.edu.mx

Ismael Caballero
University of Castilla-La Mancha, Spain
ismael.caballero@uclm.es

Laure Berti-Équille
IRD-Institut de Recherche pour le Développement, France
laure.berti@irisa.fr

Mario Piattini
University of Castilla-La Mancha, Spain
mario.piattini@uclm.es

Abstract: The use of Web applications in order to provide data with an acceptable level of quality is currently of paramount importance for any enterprise that wishes its business processes to succeed. The adequate management of the corresponding data resources through the introduction of all those aspects whose aim is to monitor the levels of quality for the task in hand is therefore essential. We claim that the introduction of such elements and mechanisms should take place during the Web application development process. To the best of our knowledge, there is still a lack of methodological and technological proposals with which to design data quality aware applications in the field of Web application development. Based principally on the benefits provided by the Model Driven Web Engineering (*MDWE*), this paper proposes a metamodel and a UML profile (*DAQ_UWE*) for the management of Data Quality elements in the design of Web applications. The main objective is to provide the designer with the necessary tools to design Web applications, thus preventing data quality problems and ensuring data quality through design.

Key Words: Data Quality Awareness, Web Design, Web Engineering, Model Driven Web Engineering, Data Quality by Design

1 INTRODUCTION

It is possible to state that an information system is as good as the data within that system. An increasing number of organizations are rediscovering this fact, once they have begun to modernize older systems into new ones like Web applications [12]. In today's world, new information technologies play an important role in the complexity of information sharing. The Internet and the new tools for User Interface (UI) design have resulted in companies being closely interwoven with each other (B2B) and/or with the consumer (B2C). With the simultaneous transfer of information constantly taking place, it is more important than ever for companies to be able to store and display data with adequate levels of quality [2, 5].

Some of the characteristics required for data quality (DQ) are that the data must be complete, correct, useful and accessible among systems [5]. If data does not fulfill these requirements, organizations will suffer financially from errors, duplicates and inconsistencies. For example, the scenarios described by Levis [35] are: ill-defined processes, deficient design applications, redundant databases and defective data design. This will result in extra costs, rework, and wasted time, in addition to a loss of system and company credibility and reliability. Various organizations

therefore continue to spend large amounts of money on data quality projects and initiatives [8, 16, 42]. These problems also cause different kind of damage to organizations, with an increasingly high cost in economical terms [16, 33, 44]. Once organizations are aware of the situation, they are willing to eradicate these kinds of problems.

Many companies currently manage most of their business and organizational processes through Web applications, thus generating, transferring and interchanging of a large amount of data whose quality needs to be managed. These Web applications have certain characteristics that make them different from common desktop applications, such as the ability to distribute the persistence of data, the data they offer (content) which originate from different sources, access to the different scenarios in which they offer information (navigation) and how the users of the Web applications are provided with information (functionalities) [1]. These unique characteristics of Web applications have forced software engineering to work on new methodologies and technologies that can cope with these new requirements, and Web developers therefore need to adopt such methodologies and technologies [7, 9, 11].

On the other hand, Internet has grown considerably in the last few years, leading to new research works that search for those methodologies and technologies which are usable in this field, and which are mainly related to functional modeling. A new area of development of Web applications known as Web Engineering [36] has consequently emerged. The paradigm of Model Driven Engineering (MDE) is also a solution that has been widely applied in Web development, obtaining successful results [37]. The application of MDE to Web Engineering has generated a new discipline called Model Driven Web Engineering (*MDWE*). MDWE proposes the depiction of concepts using metamodels, and supports the development process with a set of transformations and relations between these concepts. This approach leads to agile developments and assures consistency between models [17].

Most of the research work conducted in this area has principally been oriented towards the analysis and design phases, and various programming languages, methods, tools and design patterns for Web modeling have been proposed in this context [19]. Almost all of them offer specific processes to support the systematic and semi-automatic development of these applications. However, after performing a systematic literature review, we discovered that there is no work that covers the issues related to the management of data quality during the steps involved in designing and developing Web applications [24]. The only work that covers aspects of data quality is the work by Guerra-García et al. in [22], in which the authors propose a metamodel and a UML profile with which to capture and specify DQ requirements (*DQ_WebRE*), but not in Web application design, which is currently still lacking.
Upon observing the lack of proposals for managing DQ characteristics during the design step of Web applications, the need to model different kinds of elements that are the foundations for DQ management became our research and practitioner challenge. The main contribution of this work, both in Web Engineering and in the Data Quality area, is the proposal of an *extended metamodel and a UML profile to incorporate aspects for DQ management during the **design stage***, through various UML class diagrams, all of which is focused on representing different specific views (*presentation*, *process* and *navigation*). A set of mapping rules is also proposed, thus collaborating in the better design of *Data-Quality aware Web applications*.
We claim that, as Information Systems in general and Web applications in particular, are built to satisfy certain business requirements, if they were to be *designed* to also satisfy specific DQ requirements it would be possible to assure users that data quality problems could be avoided or at least mitigated. The goal of our research is therefore to provide developers with the suitable modeling tools for use in the design of elements that would be responsible for managing the quality of the data of Web applications.
The remainder of the paper is structured as follows: the second section provides a brief description of the state-of-the-art as regards DQ and Web Engineering. Section 3 presents the extended metamodel and the *DAQ_UWE profile* proposed for designing DQ elements, along with their mapping rules. Section 4 describes the usage of the *DAQ_UWE* profile through a case study, showing different design models. Finally, Section 5 concludes the paper and presents our future work.

2 RELATED AREAS

2.1 Data Quality

In spite of the existence of the various definitions of the term Data Quality [4], it is recently been recognized that the most widely accepted definition of the term Data Quality is "*fitness for use*" [21]. This means that a user evaluates the quality of a set of data for a particular task carried out in a specific context, according to a set of criteria or dimensions of Data Quality, thus determining whether these data could be used for that purpose [43].

One of the most interesting strategies for the study of data quality in a specific context is to divide it into several characteristics known as *data quality dimensions* [34]. DQ is therefore recognized as a multidimensional concept: in order to measure the DQ level of a piece of data, it is necessary to identify various DQ dimensions that best represent the user's data quality requirements. The set of data quality dimensions identified is known as a "*DQ model*" [34].

Although many DQ models have been proposed by different authors, most of them are fairly domain dependent with reduced applicability [6]. DQ professionals have to choose from the data quality dimensions that appear in the model, those that best fit the user's perception. In order to obtain the most independent perspective possible, we have chosen the generic DQ model proposed in the ISO/IEC 25012 standard [26] for our research.

This international standard defines fifteen data quality characteristics from two perspectives:

- *Inherent*: this refers to the extent to which the 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 attained and preserved within a computer system when data is used under specified conditions.

The data quality characteristics proposed by the ISO/IEC 25012 standard [26] are: *Accuracy*, *Completeness*, *Consistency*, *Credibility*, *Currentness*, *Accessibility*, *Compliance*, *Confidentiality*, *Efficiency*, *Precision*, *Traceability*, *Understandability*, *Availability*, *Portability* and *Recoverability*. It is worth mentioning that each definition of the characteristics is very generic, so they should be adapted according to the context and DQ requirements of each user.

As mentioned previously, no study in literature considers certain specific aspects or characteristics of DQ during the design stage of Web application development. The only related work is presented by Guerra-García et al. [22], in which an extended metamodel and a UML profile (*DQ_WebRE*) are proposed for the *DQ requirements specification* in Web application development.

This work proposes the usage of use cases and activity diagrams. The analyst is able to use some particular elements which are in charge of specifying and managing the aspects related to the DQ requirements specifications (e.g. "*InformationCase*", "*DQ_Requirement*", "*DQ_Metadata*", "*DQ_Validator*", etc.), in addition to linking them to each element focused on the specification of Web application functionalities (use cases).

Having considered the development stages defined in the Unified Development Process (*UDP*) [27], and taking the previous proposal on dealing with *DQ requirements and analysis* as a starting point, the objective of work presented herein is to continue this approach and offer the necessary elements for the *management of DQ in the design stage*, thus allowing developers to be aware of the DQ requirements that should be considered, and to continue their implementation first in the design stage, and later in coding in the Web application. An in-depth description of each of the new elements proposed for the management of DQ in the design stage is shown in Section 3.

2.2 Web Engineering

After reviewing the different methodologies that support the modeling in the design phase of Web applications, we found the following proposals: OOWS [20], UWE [28], WebML [10], W2000 [3], WSDM [14], SOD-M [13] and WebSA [36].

In [20], Fons et al. propose *OOWS* (Object Oriented Web Solutions) as an extension of an object-oriented software production method (OO-Method, [41]); OOWS provides a full method that defines a set of activities with which to correctly specify the functional, navigational and presentational characteristics of Web applications. This method is comprised of two main steps- *system specification* and *solution development*. The first step concerns the specification of the system functional requirements, while the second step proposes a strategy oriented towards generating the software components that constitute the solution (the final software product).

Koch and Kraus's proposal *UWE* [28] covers the special aspects of Web application design: these authors define a set of special views graphically represented by UML diagrams, such as a navigation model and a presentation model, through the use of UML profiles. They additionally show how to use different kinds of static diagrams to model the static aspects of the Web applications. The use of other types of diagrams is similarly encouraged in the design of the

dynamic behavior of the Web Application: e.g. UML 2.0 activity diagrams to model the tasks. It is worth noting that the UWE proposal provides a detailed description of the concepts used to design Web applications and their semantics. This proposal is based on a common metamodel and also includes tools to support the design and the semi-automatic generation of Web applications.

WebML [10] is a high level language for the specification of 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.

The proposal by Baresi et al. in [3] enriches the latest version of HDM (*Hypertext Design Model*) with concepts and notations for modeling richer information structures, along with operations and services which are accessible through the Web. It takes advantage of the UML language and its customizability, giving *W2000* a standard graphical syntax. The authors have additionally developed a CASE toolbox that assists Web application designers during the entire development process: from conceptual design to the semi-automatic generation of final applications.

The *WSDM* (*Web Site Design Method*) proposal [14] is focused on the management of requirements that should be carried out by using certain techniques, such as concept maps and the data dictionary for the definition of specific functional and security requirements. This approach considers the specification of requirements through a textual form, thus allowing it to produce some precision errors.

The *SOD-M* proposal [13] 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 [36], Meliá proposes a generic design approach named *WebSA*. This is based on the MDA (Model Driven Architecture) paradigm [38], 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. WebSA basically provides the designer with a set of models that are focused on architecture and transformation models in order to specify Web applications.

All of these methodologies are principally focused on how to identify and define the functional aspects, related principally to navigation, conceptual and presentation models. However, none of them deals with data quality issues. Only the proposal by Ceri et al. (WebML) [10] mentions some specific objectives of information that should be considered when designing Web application. However, these authors do not study this in any depth, and nor do they consider any specification of DQ characteristics.

3 DAQ_UWE: EXTENDING UWE METAMODEL TO SUPPORT THE DESIGN OF DATA QUALITY AWARE WEB APPLICATIONS

After conducting a comparative study of the different proposals, we decided to take the *UWE* proposal as a basis for our work, which was first introduced by Koch and Kraus in [28] and was updated in [29] [30], in which the authors proposed a metamodel for representing concepts and relationships of Web Engineering. This metamodel constitutes the basis for defining a UML profile for *Web Application Design (UWE)*, with a particular focus on systematic design, personalization and semi-automatic generation. UWE defines special views graphically represented by UML class diagrams, which are organized into four different packages: *Navigation*, *Presentation*, *Process* and *Content*. A complete description of the elements used in each package is shown in [30]. We decided to consider the *UWE* proposal as a platform for adding the DQ elements, since it provides a good flexibility in its metamodel and its UML profile for Web design. All of this will allow us to model the DQ elements and relate them to each of the elements defined in the UWE metamodel.

One of the main motivations of our investigation is to provide the designers and developers of Web applications with the DQ elements needed in the *design stage*, in order to better specify, in a clear and intuitive manner, the DQ characteristics that should be implemented in the Web application. These elements will be related to the different elements of Web design.

This section therefore describes our proposal, which consists of extending the UWE metamodel for the integration of those concepts that are considered essential for the management of DQ in the design stage. We first reviewed the

main proposals with regard to DQ modeling as presented in [25], and we considered the following key concepts in addition to three new elements as part of the process package: "*DQProcessClass*", "*DQProcessProperty*" and "*DQLink*", along with three new elements in the content package: "*DQDim*", "*DQDimProperty*" and "*DQ_Constraint*", and finally one element in the presentation package: "*UIE_DQVerifier*".

- ***DQProcessClass.*** This represents a specific process that is responsible for managing the DQ metadata which is associated with the data that will be handled in each "*ProcessClass*". In other words, the *DQProcessClass* will be able to use the "*DQDim*" class, which specifies the DQ metadata of each DQ dimension defined.
- ***DQProcessProperty.*** A *DQProcessProperty* is owned by a *DQProcessClass* and it is used to specify the DQ dimensions that should be met by each piece of data managed in the related *ProcessClass*.
- ***DQLink.*** This represents a special link, and is used to connect *ProcessClass* elements with *DQProcessClass* elements.
- ***DQDim.*** This represents a content element; this metaclass corresponds to all of the DQ dimensions: it will be in charge of saving the different DQ metadata (*DQDimProperty*).
- ***DQDimProperty.*** This is owned by a DQ dimension and it is used to define the DQ metadata related to each DQ dimension (*DQDim*).
- ***DQ_Constraint.*** This represents a content element. This metaclass corresponds to all of the "*Constraints*" that should be defined and associated with the different DQ dimensions, in addition to its corresponding bounds (e.g. "*upper_bound*" and "*lower_bound"*).
- ***UIE_DQVerifier.*** This metaclass represents a class which is responsible for specifying and verifying that the data managed by each element in the user interface meets the DQ dimensions (*DQDim*) specified.

Table 2 (Appendix A) shows an analogous specification to UML[40] of each new DQ stereotyped element, in which each stereotype specification contains: name, base class, description, constrains, used in, tagged values (optional) and notation (Icon).
Bearing in mind the advantages of the MDWE paradigm, and in order to attain a better understanding of how a designer can use each new DQ element, we propose a set of mapping rules (see Table 1).

DQ_WebRE element	DAQ_UWE element	Description	Used in
Add_DQ_Metadata	DQProcessClass	Each "*Add_DQ_Metadata*" element will be transformed into a "*DQProcessClass*" element. The relation of an element of the "*UserTransaction*" type with another element of the "*Add_DQ_Metadata*" type will be converted into a specific relation of the "*DQLink*" type, between "*ProcessClass*" and "*DQProcessClass*" type elements. The name of the "*DQ_Metadata*" and "*DQ_Validator*" type elements related to "*Add_DQ_Metadata*", will be converted into attributes of the "*DQDim*" type in the "*DQProcessClass*" element.	Navigation model
DQ_Metadata	DQDim	Each "*DQ_Metadata*" element will be transformed into a "*DQDim*" element, in addition to copying its respective defined attributes (*DQDim properties*). Each relation between elements of the "*DQ_Metadata*" and "*Contents*" types will be converted into a relation between a "*DQDim*" element and a "*ContentClass*" element (the class in which data is normally stored).	Content model
DQ_Validator	UIE_DQVerifier	Each "*DQ_Validator*" element will be transformed into a "*UIE_DQVerifier*" element. A relation between the "*DQValidator*" and "*WebIU*" type elements will be converted into a relation between "*UIE_DQVerifier*" and "*PresentationPage*" type elements; this relationship could become more specific to particular elements within the *PresentationPage*, e.g. *presentationGroup*, *textInput*, etc.	Presentation model
DQ_Validator	DQDim	Each "*DQ_Validator*" element will be transformed into a "*DQDim*" element. This element will be in charge of storing the validation operations, and will represent them in the Content model (If this "*DQDim*"element has been defined previously, it will only be necessary to add the validation operation).	Content model
DQConstraint	DQ_Constraint	Each "*DQConstraint*" element will be transformed into another "*DQ_Constraint*" element. This element will be in charge of storing the specific data to which the constraints will be related, in addition to the variables used to define the maximum and minimum bounds ("*upper_bound*" and "*lower_bound*"). A relation between the "*DQ_Validator*" and "*DQConstraint*" elements will be converted into a relation between "*DQDim*" and "*DQ_Constraint*" type elements in the Content model.	Content model

Table 1. Proposals of mapping rules from "*DQ_WebRE"* to "*DAQ_UWE"* elements

These proposals for mapping rules should be taken in account by the designer when modeling the corresponding design diagram (*content*, *navigation*, *presentation*). It is worth noting that all these DQ stereotyped elements can be obtained from the different elements modeled in the activity diagram using the *DQ_WebRE profile* [22].
In order to support our approach, we propose a UML profile for Web application design which is augmented with data quality (*DAQ_UWE*). The profile was implemented in the Enterprise Architect tool, which supports the definition of UML-based profiles, thus allowing its use in practically any other type of modeling tool. Once the *DAQ_UWE profile* has been implemented, we obtain a suitable work environment in which to model design diagrams with the new elements defined (see toolbox in Figure 3).

4 ILLUSTRATIVE CASE STUDY

The case study we are going to present is based on the *EasyChair Conference System* [15], one of the Web applications most frequenly used to support conference management. This application allows the paper to be submitted, the management and monitoring of Program Committee (PC) members, the assignment of papers to reviewers and many others features. 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).
We propose starting to model a specific use case diagram (see Figure 1) for each actor involved in the application, in addition to modeling the specific DQ requirements involved, by using the elements defined in *DQ_WebRE profile* proposed in [22] (Section 4.1); we shall then obtain the corresponding design diagrams by using the elements defined in the *DAQ_UWE profile* and considering the mapping rules proposed (Section 4.2). Finally, in Section 4.3 a brief example of instantiation is shown.

4.1 Capturing and modeling DQ requirements by means of DQ_WebRE

As developers, we are first interested in highlighting how to capture the main functionalities of the Web application, along with the principal DQ characteristics for the data used in the execution of the application.
By using the *DQ_WebRE* profile proposed in [22], the functionalities of each type of user (namely actor) are represented in a use case diagram. The analysts will first model the Author's main use cases (see Figure 1), and then, by means of activity diagrams, they will attempt to provide a more detailed description of each of the use cases identified. Owing to length restrictions, in this work, we shall focus solely on the "*New Submission*" use case that an "*Author*" may require of the EasyChair application.

Based on the work presented by Guerra-García et al. in [23], in which the authors conducted an analysis of which DQ characteristics are presumably linked directly to each of the functionalities owned by a Web application, and considering that the "*New Submission*" use case can be catalogued as a Web funcionality of the "*Content Management*" type, we can observe that, according to [23], this functionality could be related to the following DQ characteristics: Accuracy, Completeness, Consistency, Credibility, Currentness, Accessibility, Compliance, Confidentiality, Efficiency, Traceability, Understandability and Portability. It should be noted that the Analyst has the freedom to incorporate the DQ characteristics as is considered necessary according to the context of the application, in order to mitigate potential problems of quality in data that will be managed in each specific use case.

Bearing the above considerations in mind, we proceed to model the use case diagram (see Figure 1). We can observe the specification of the specific use cases "*Add information of each author*", "*Add paper information*" and "*Add name of file*" (stereotyped as "*InformationCase*"), which will be in charge of managing all the data involved (see comment in Figure 2) in the "*New Submission*" use case.

Once this has been done, it is posible to specify the DQ requirements that will be related to each of the *InformationCase* elements, and the following DQ requirements are defined: if we wish to guarantee the Confidentiality of data, the execution of the following DQ requirement should be take place: "*ensure that data will only be accessed by authorized users*". In addition, in order to guarantee the Completeness of data, the following DQ requirement should be executed: "*check that all data were completed*". The analyst will similarly be able to guarantee the characteristics of *Consistency*, *Currentness*, *Traceability* and *Compliance* of data, through the

specification of the following respective DQ requirements: "*verify that data of each author are coherent with author's data previously registered*", "*validate that data were updated before deadline*", "*check who is able to upload a file, and when*" and "*verify the type of extension of each file*". "V*erify the file size*", should also be executed with the aim of checking that the file entered is not empty and has not corrupted the DQ requirement of Accuracy. Finally, a DQ requirement of Credibility must be added in order to verify that the author´s data are the same in both the Web form and in the pdf file content for author name and affiliation.

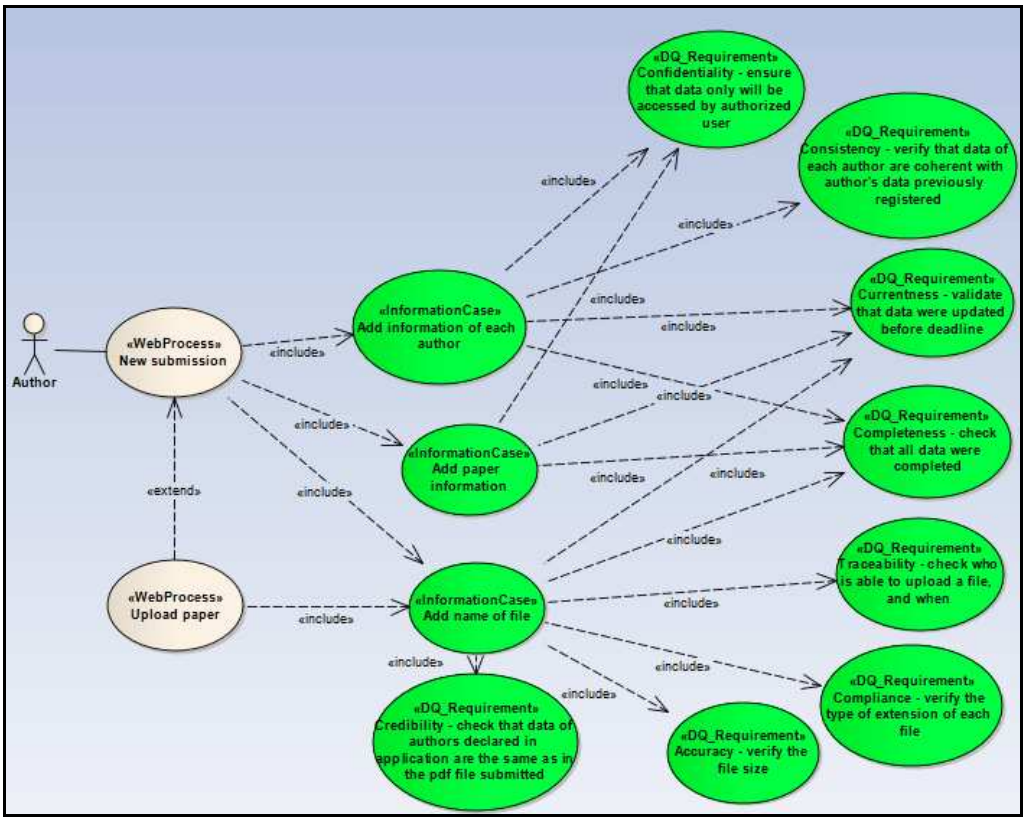

Figure 1. Use case diagram specifying DQ requirements

Having completed the use case diagram, and in order to widely detail the description of the "*New Submission*" use case, it is possible to draw an activity diagram using the corresponding elements (stereotyped) defined in the *DQ_WebRE profile*.

For this activity diagram (see Figure 2), the Analysts will be able to model the specific activities to meet the DQ 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 closely linked to the DQ requirements that each user defines for the data that will be managed in each "*InformationCase"* element.

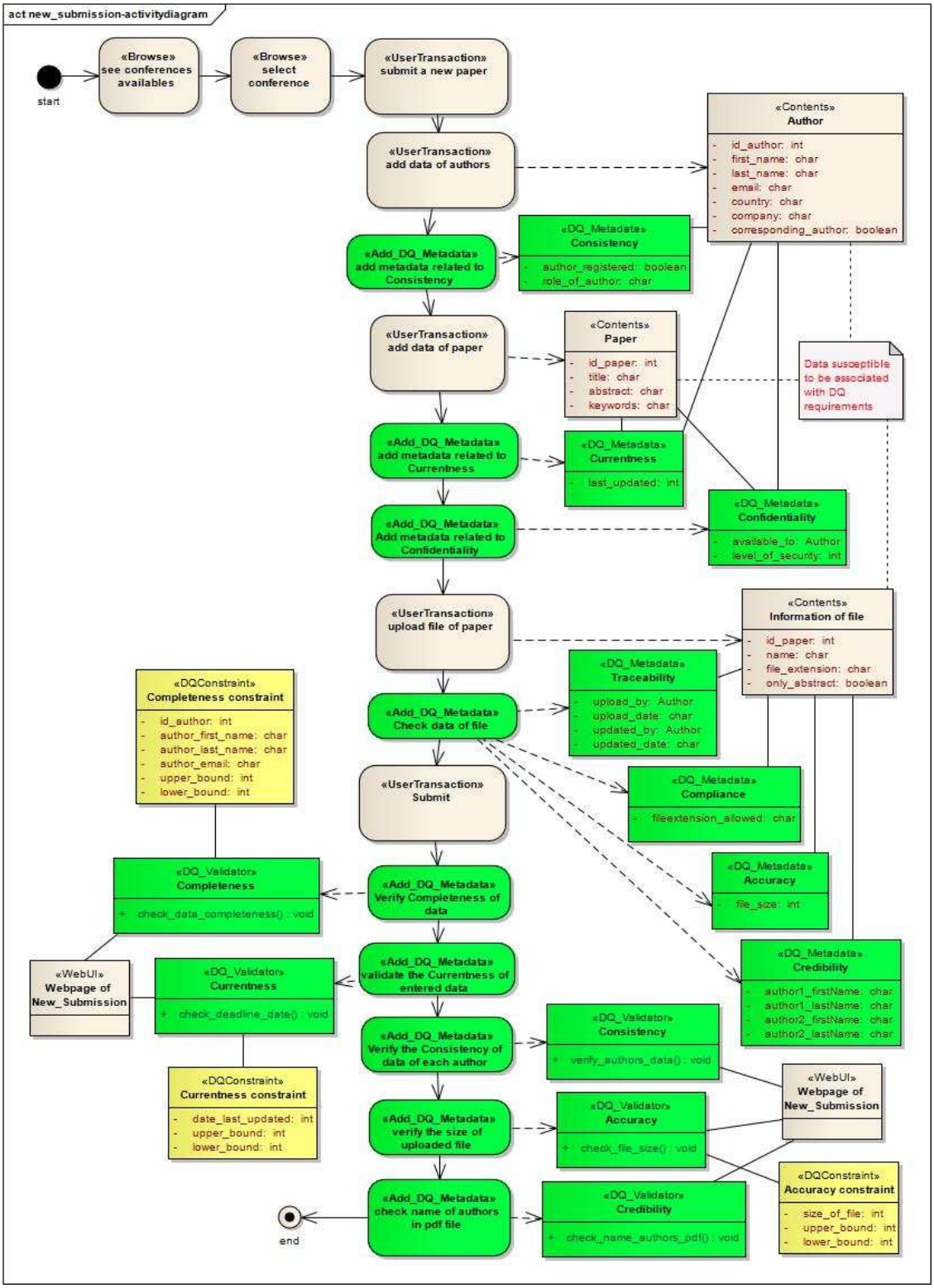

Figure 2. Activity diagram with Data Quality management

In this activity diagram, the first activities, "*add metadata related to Consistency*" and "*add metadata related to Currentness*" (stereotyped as "*Add_DQ_Metadata"*), will be responsible for capturing the metadata related to Consistency ("*author_registered", "role_of_author"*) and Currentness ("*last_updated"*). These metadata will be stored in an instance of the "*DQ_Metadata"* class, and will later be used to satisfy the DQ requirements of *Consistency* and *Currentness* respectively. On the other hand, the "*add metadata related to Confidentiality*" activity will capture the corresponding metadata related to "*available_to*" and "*level_of_security*", both of which are stored in a specific "*DQ_Metadata"* class. Note that all these "*DQ_Metadata"* classes are related to the *data* managed in the previous activities of "*add data of authors*" and "*add data of paper*" (stereotyped as "*UserTransaction*").
The "*Check data of file"* activity is similarly in charge of capturing the metadata related to fulfilling the specific DQ requirements of *Traceability* ("*upload_by*", "*upload_date*", "*updated_by*" and "*updated_date*"), *Compliance* ("*fileextension_allowed*"), *Accuracy* ("*file_size*") and *Credibility* ("*author1_firstName*", "*author1_lastName*", etc.).
Finally, the "*Verify Completeness of data*", "*Validate the Currentness of entered data*", "*Verify the size of uploaded file*" and "*check name of authors in pdf file*" activities will be responsible for adding the specific operations of "*check_data_completeness()*", "*check_deadline_date()*", "*check_file_size()*" and "*check_name of_authors in_pdf file*", in order to verify the Completeness, Currentness, Accuracy and Credibility of the data managed in the "*Webpage of New_Submission*" element (stereotyped as "*WebUI*").

The data managed within the "*Completeness constraint*" element (stereotyped as "*DQConstraint*") will be used to specify the data that should be required for completion, in addition to specifying the bound of completeness that each one must fulfill, through the values specified in "*upper_bound*" and "*lower_bound*". Moreover, the data stored in the "*Currentness constraint*" element will serve to define the valid bounds (through "*upper_bound*" and "*lower_bound*") and determine that the date of updated data ("*date_last_updated*") related to the *Author* is kept current. The "*Verify the Consistency of data of each author*" activity will be in charge of verifying these data, which will be entered through the "*Webpage of New_Submission*" *WebUI* stereotyped element.
Finally, the data stored in the "*Accuracy constraint*" element will be used to verify that the file entered is not empty, along with delimiting the maximum size allowed (e.g. quantity of kilobytes), through the values established in "*upper_bound*" and "*lower_bound*".

4.2 Modeling the design diagrams by using DAQ_UWE

To continue with the separation of UWE concerns, in this section we shall show the different design models for *content*, *navigation* and for *presentation* separately, always bearing in mind the mapping rules previously proposed. It is worth mentioning that the authors of the UWE metamodel have proposed a set of transformation rules (see [31, 32]) with which to obtain the various elements used in the design models, from the elements defined in the *WebRE* proposal [18] for requirements specification in Web applications.

By taking the stereotyped elements used in the previous activity diagram as a basis, and considering both our *mapping rules* as *UWE transformations*, the designer will be able to model the corresponding design models using the stereotyped elements defined in the *DAQ_UWE profile*, once each of the diagrams are shown.

The *Content model* is shown in Figure 3 with the classes defined for storing the principal information concerning "*Author"*, "*Paper"* and "*Information of file"* (stereotyped as "*ContentClass"*). This diagram also shows the classes related to the definition of each DQ dimension (stereotyped "*DQDim*"): *Completeness, Compliance, Confidentiality, Currentness, Accuracy, Consistency, Credibility* and *Traceability*, along with their attributes (of the "*DQDimProperty*" type) and the operations defined. It is also possible to model the "*currentness constraint*", "*accuracy constraint*" and "*completeness constraint*" (stereotyped as "*DQ_Constraint*") classes, which will store the data used to specify the constraints and bounds defined for the different DQ dimensions. This *Content model* is represented as a plain UML class diagram.

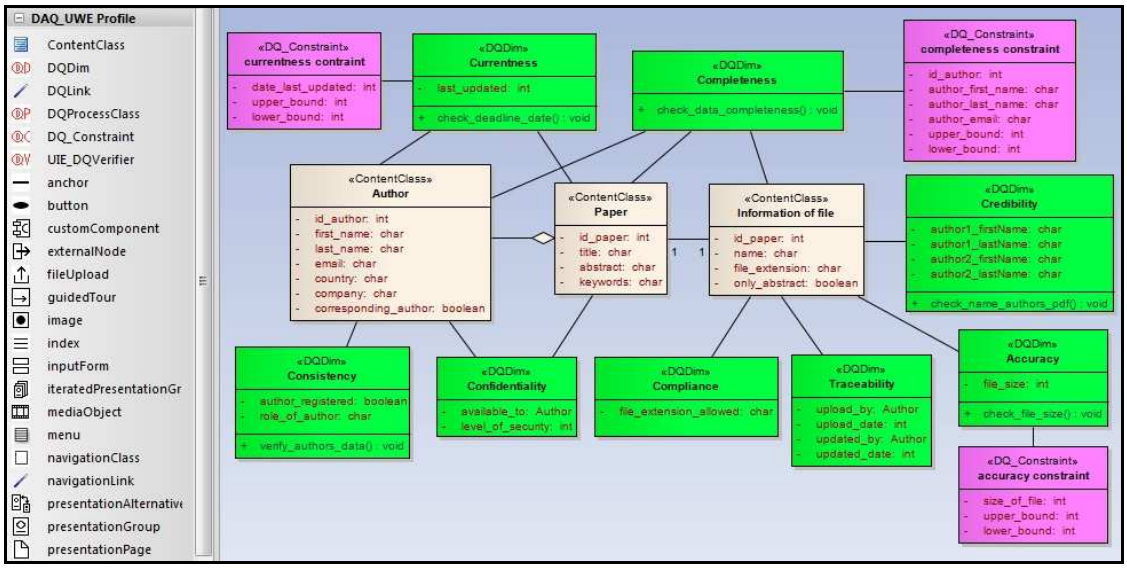

Figure 3. Content model with elements for DQ management

By making use of the stereotyped elements defined in the *DAQ_UWE* profile, it is also possible to model the hypertext structure using a *Navigation model* (see Figure 4). This diagram shows nodes and links (denominated as *Process nodes*, and integrated into the navigation flow through the use of process links). UWE distinguishes between different types of nodes, such as *navigation class, process class, menu, index, query*, etc. This design diagram shows a navigation class (“*see conferences available”),* an index class (“*select conference”)* and a menu class (“*submit a new paper*”). All the data entered in the Web application is modeled by means of the “*add data of authors*”, “*add data of paper*” and “*upload file of paper*” classes of the “*ProcessClass”* type.
The DQ management is similarly modeled with “*DQProcessClass”* (e.g. “*add metadata related to Consistency”*, “*add metadata related to confidentiality”*, “*check data of file”*, “*verify completeness of data*”, “*validate the currentness of entered data*”, “*add metadata related to currentness”*, “*verify the size of uploaded file*”, “*Verify the consistency of data of each author*” and “*check name of authors in pdf file*”) type elements, which are related to the different “*ProcessClass*” type classes through a specific “*DQLink”* relation. These *DQProcessClass* are in charge of indicating the DQ characteristics that will satisfy the data involved in each “*Process node”*.

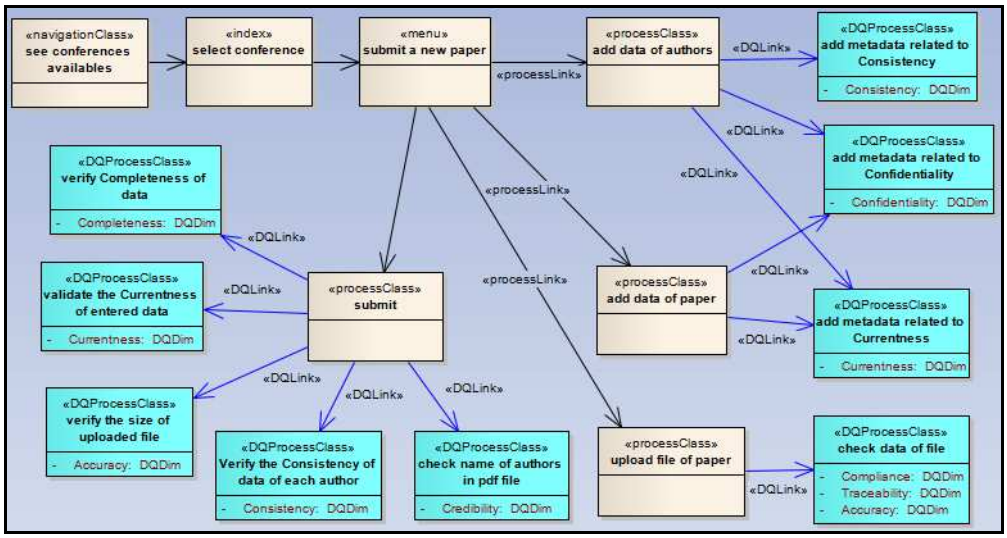

Figure 4. Navigation model with DQ management

Figure 5 shows the *Presentation model* of the running example. We have also used a design class diagram for the representation of presentation models. The container form is selected in order to provide a more intuitive representation of the pages. The presentation page "*submit a new paper*" contains three presentation groups: "*add data of authors"*, "*add data of paper"* and "*upload file of paper"*.
Different elements are shown for each presentation group. For instance, the following elements are modeled in the "*add data of authors*" Presentation Group: *first_name*, *last_name*, *email*, *country*, and *company* (of the "*textInput"* type) and *corresponding_author* ("*selection*" type), while the "*add data of paper*" Presentation Group contains the fields: "*title*", "*abstract*" and "*keywords"* of the "*textInput*" type. The "*upload file of paper*" Presentation Group type similarly contains the following elements: "*name*" ("*textInput*" type), "*only_abstract*" ("*selection*" type) and "*add file*" ("*fileUpload*" type).

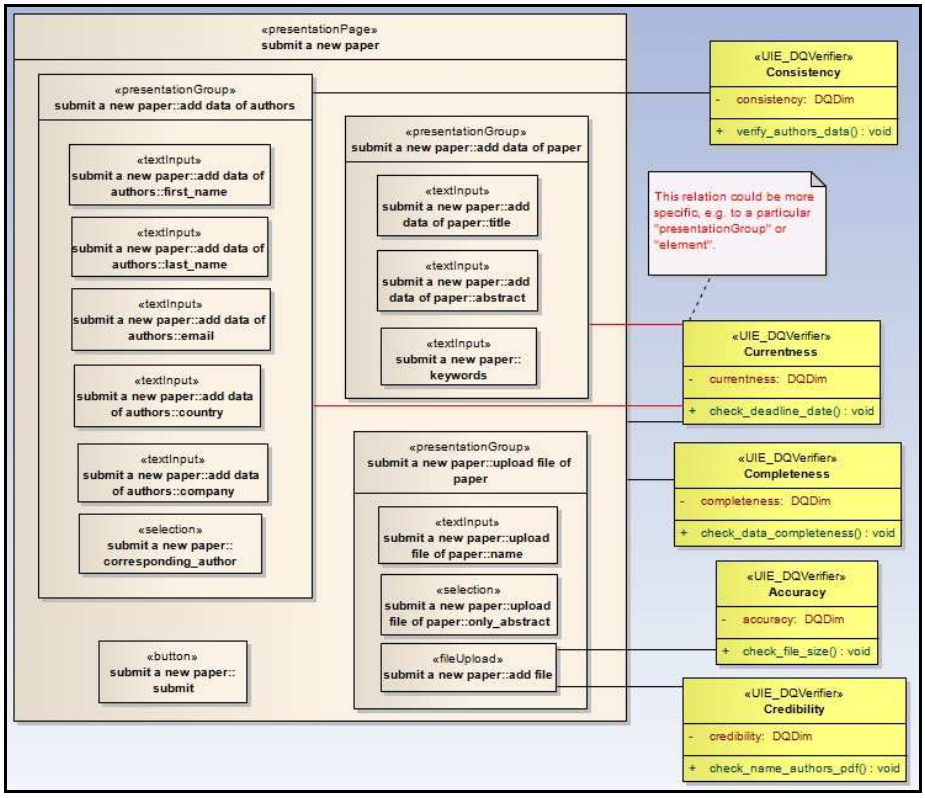

Figure 5. Presentation model including elements for DQ management

In this presentation model, we can see that the "*Completeness*" class (stereotyped as "*UIE_DQVerifier*") is related to the "*submit a new paper*" Presentation Page. This means that this class will be in charge of verifying the completeness (through the operation "*check_data_completeness*") of the data entered in each element contained in this group, thus representing the fulfillment of the DQ requirement of Completeness. The "*Currentness*" class (stereotyped as "*UIE_DQVerifier*") will similarly be in charge of executing the specific "*check_deadline_date*" operation in order to validate the currentness of each of the elements contained in the "*submit a new paper*" presentation page. The "*Consistency*" class will be responsible for verifying that all the data of each author that will be entered are maintained in a consistent manner, through the execution of the "*verify_authors_data*" operation. The "*Accuracy*" class will likewise be in charge of executing the "*check_file_size*" operation to check that the file size is in accordance with the bounds permitted (note that this element is related solely to a particular presentation element). Finally, the "*Credibility*" class will be responsible for verifying that the author´s name in the pdf file is the same as the name entered in the application.

It is worth noting that all the elements in charge of managing the DQ requirements (stereotyped as "*UIE_DQVerifier*") were derived from the previous activity diagram (Figure 2). At any given time, the Designer would be free to associate a specific DQ requirement with a particular presentation group, or even with a specific element (see attached note in Figure 5).

4.3 A brief instantiation example

After executing the "*New submission*" use case, the diagram shown in Figure 6 shows an example of an instantiation for the "*Content model*" with the data entered. This diagram shows that the DQ requirements previously established have been met. For example, the class responsible for storing the *completeness restrictions* contains detailed information concerning the required fields as being obligatory, in order to meet the DQ requirement of Completeness by 100%. Moreover, the data stored in the "*Confidentiality*" class shows that only the authors identified as "1" and "2" are able to observe the information regarding *paper 7*. Similarly, we can see the data stored in the "*Compliance*" class, thus defining that it is only possible to store files with the "*pdf*" type extension.
In addition, for the instance of the "*Traceability*" class we can see the data related to the author who first introduced the file, and the author that updated it last, thus maintaining the traceability requirement previously defined. The names of authors written in the pdf file are likewise stored in the instance of the "*Credibility*" class, and as long as these data are the same as those stored in the "*author1*" and "*author2*" classes, the credibility requirement is ensured.

In a similar way, there is a value stored in the "*Currentness*" class which represents the date of the last update of data from both the author and the paper. Upon analyzing the data stored in the "*constraints of currentness*" class, in which the range of days allowed (*from 0 to 365*) is also defined, we can see that the DQ requirement of Currentness has been fulfilled, since the data are maintained within the bounds permitted (this validation is carried out by using the "*check_deadline_date*" operation defined in the *Currentness* class). On the other hand, upon observing the values stored in the "*constraints of accuracy*" class, it will be noted that the accuracy constraint has been fulfilled, since the file size (700 Kb) remains between the bounds permitted (from 1 to 2000 Kb).

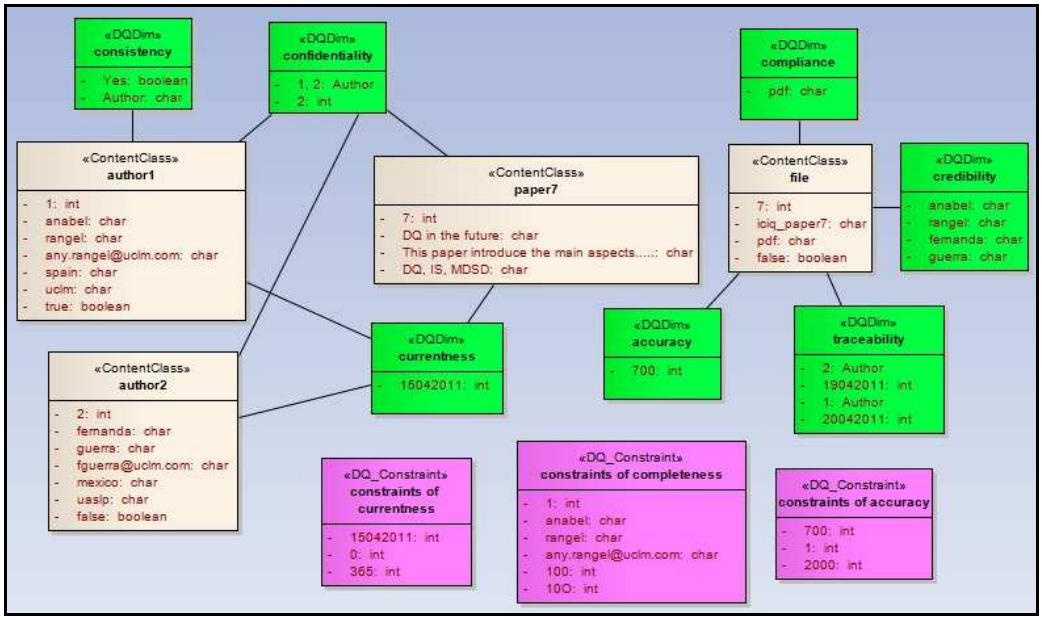

Figure 6. An example of instantiation "Content model"

5 CONCLUSIONS AND FUTURE WORK

In the last decade, Web applications have emerged as leading information resources for most organizations and enterprises. It is therefore essential that these applications provide data with appropriate quality levels for the purposes required by different users. However, none of the current Web design methodologies include any kind of elements or artifacts for the management of DQ characteristics.

A proper management of DQ elements during the design stage could help designers to erradicate some possible problems with the data, along with allowing programmers to code the appropriate mechanisms for a correct management of the quality of the data managed by the application.

In order to confront to this problem, and taking the model-driven Web engineering approach as a basis, we present an extended metamodel and a UML profile (*DAQ_UWE*) with which to manage the DQ elements in Web application design. The use of this profile will allow us to model the key elements related to data quality in the design stage in any kind of Web application development.

As future work we plan to implement our approach in various Web development projects, in order to obtain feedback to discover the advantages and benefits achieved, along with quantifiable project management data (e.g. time required for DQ specification implementation), which will allow us to improve our approach. We also anticipates the incorporation of model transformation rules through the *QVT* (Query/View/Transformation) language [39], taking the proposals of the aforementioned mapping rules as our basis. We additionally aim to develop a tool that will automatically support all the stages in the development cycle, from analysis and design to code generation. Our eventual goal is to develop Web applications more quickly, in turn ensuring the quality of the data they manage.

ACKNOWLEDGEMENTS

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

[1] Aguilar, J.A., et al., *An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering.* Universal Computer Science, 2010. 16: p. 2475-2494.

[2] Akoka, J., et al. *A Framework for Quality Evaluation in Data Integration Systems.* in *International Conference on Enterprise Information Systems, ICEIS*. 2007.

[3] Baresi, L., et al., *Meta-modeling Techniques Meet Web Application Design Tools*, in *Fundamental Approaches to Software Engineering*. 2002, Springer Berlin / Heidelberg. p. 182-206.

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

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

[6] Caballero, I., et al., *Tailoring Data Quality Models Using Social Network Preferences*, in *Database Systems for Advanced Applications*, L. Chen, et al., Editors. 2009, Springer Berlin / Heidelberg. p. 152-166.

[7] Cachero, C. and J. Gómez. *Advanced Conceptual Modeling of Web Applications: Embedding Operation*. in *21th International Conference on Conceptual Modeling*. 2002.

[8] Cai, Y. and G. Shankaranarayanan, *Managing data quality in inter-organisational data networks.* International Journal of Information Quality, 2007. 1(3): p. 254 - 271.

[9] Casteleyn, S., W.V. Woensel, and G.-J. Houben, *A semantics-based aspect-oriented approach to adaptation in web engineering*, in *Proceedings of the eighteenth conference on Hypertext and hypermedia*. 2007, ACM: Manchester, UK. p. 189-198.

[10] 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.

[11] Ceri, S. and I. Manolescu, *Constructing and integrating data-centric web applications: methods, tools, and techniques*, in *Proceedings of the 29th international conference on Very large data bases - Volume 29*. 2003, VLDB Endowment: Berlin, Germany. p. 1151-1151.

[12] Clement, D., S. Ben Hassine Guetari, and B. Laboisse. *Data Quality as a key sucess factor for migration projects.* in *The 15th. International Conference on Information Quality, ICIQ 2010*. 2010. Little Rock,

Arkansas.
[13] De Castro, V. and E. Marcos, *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, 2009. 18, No. 2: p. 225-260.
[14] De Troyer, O.M.F. and C.J. Leune, *WSDM: a user centered design method for Web sites.* Computer Networks and ISDN Systems, 1998. 30(1-7): p. 85-94.
[15] EasyChair. *EasyChair Conference System*. Available from: http://www.easychair.org/.
[16] Eppler, M. and M. Helfert. *A Classification and Analysis of Data Quality Costs*. in *International Conference on Information Quality*. 2004. MIT, Cambridge, MA, USA.
[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] 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.
[19] Escalona, M.J., et al., *The treatment of navigation in web engineering.* Advances in Engineering Software, 2007. 38(4): p. 267-282.
[20] Fons, J., et al., *Development of Web Applications from Web Enhanced Conceptual Schemas*, in *Conceptual Modeling - ER 2003*, I.-Y. Song, et al., Editors. 2003, Springer Berlin / Heidelberg. p. 232-245.
[21] Ge, M. and M. Helfert. *A Review of Information Quality Research*. in *International Conference on Information Quality*. 2007. MIT, Cambridge, MA, USA.
[22] Guerra-García, C., I. Caballero, and M. Piattini. *Capturing Data Quality Requirements for Web Applications by means of DQ_WebRE.* in *2nd International Workshop on Business intelligencE and the WEB, BEWEB 2011*. 2011. Uppsala, Sweden: ACM 978-1-4503-0610-2/11/03.
[23] 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 on WWW/INTERNET 2009*. 2009. Rome, Italy.
[24] 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): p. To be published.
[25] 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.
[26] ISO-25012, *ISO/IEC 25012: Software Engineering-Software product Quality Requirements and Evaluation (SQuaRE)-Data Quality Model.* 2008.
[27] Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process.* 1999: Reading (MA): Addison-Wesley.
[28] 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.
[29] Koch, N. and A. Kraus, *Towards a Common Metamodel for the Development of Web Applications*, in *Web Engineering*, L.N.i.C. Science, Editor. 2003, Springer Berlin / Heidelberg. p. 419-422.
[30] Koch, N. and C. Kroib, *UWE Metamodel and Profile. User Guide and Reference.*, T. Report, Editor. 2008, Institute for Informatics. Ludwig-Maximilians-Universitat Munchen (LMU), Germany.
[31] Koch, N., G. Zhang, and M.J. Escalona, *Model transformations from requirements to web system design*, in *Proceedings of the 6th international conference on Web engineering*. 2006, ACM: Palo Alto, California, USA.
[32] Kraus, A., A. Knapp, and N. Koch. *Model-Driven Generation of Web Applications in UWE*. in *3rd International Workshop on Model-Driven Web Engineering, MDWE*. 2007. Italy.
[33] Laudon, K.C., *Data Quality and Due Process in Large Interorganizational Record System.* Communications of the ACM, 1986. 29(1): p. 4-11.
[34] Lee, Y.W., et al., *Journey to Data Quality*. 2006, Cambridge, MA, USA: Massachussets Institute of Technology.
[35] Levis, M., M. Helfert, and M. Brady. *Information Quality Management: Review of an Evolving Research Area*. in *ICIQ'07*. 2007. MIT, Cambridge, MA, USA.
[36] 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.
[37] Moreno, N. and A. Vallecillo, *Towards interoperable Web engineering methods.* Journal of American Society for Information Science and Technology, 2008. 59(7): p. 1073-1092.

[38] OMG, *MDA Guide Version 1.0.1*. 2003, Object Management Group. p. 62.
[39] OMG. *MOF QVT Final Adopted Specification*. 2008; Available from: http://www.omg.org/spec/QVT/1.0/ [Accessed in January, 2011].
[40] OMG. *Unified Modeling Language: Superstructure. Versión 2.0*. 2005; Available from: <http://www.omg.org/docs/formal/05-07-04.pdf>.
[41] Pastor, O., et al., *The OO-Method Aproach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming*. Information Systems, 2001. 26: p. 507-534.
[42] 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.
[43] Strong, D.M., Y.W. Lee, and R.Y. Wang, *Data Quality in Context*. Communications of the ACM, 1997. 40(5): p. 103-110.
[44] 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).

APPENDIX A

Name	**UIE_DQVerifier**	Icon	
Base Class	Class		
Description	This metaclass represents a class responsible for specifying and verifying that the data managed by each element in the user interface meet the DQ dimensions (*DQDim*) specified.		
Constraints	- Must be related to at least one "*UIElement*" type element.		
Used in	Presentation model.		
Name	**DQProcessClass**	Icon	
Base Class	navigationNode::Class		
Description	It represents a specific process responsible for the management of DQ metadata associated with the data that will be handled in each "*ProcessClass*". In other words, the *DQProcessClass* will be able to use the "*DQDim*" class, which specifies the DQ metadata of each DQ dimension defined.		
Constraints	- Must be related to at least one "*ProcessClass*" type element.		
Used in	Navigation / Process model.		
Name	**DQProcessProperty**		
Base Class	navigationProperty::Property		
Description	A *DQProcessProperty* is owned by a *DQProcessClass* and is used to specify the DQ characteristics (dimensions) that should be met by each piece of data managed in the *ProcessClass* related.		
Used in	Navigation / Process model.		
Name	**DQLink**		
Base Class	link::Association		
Description	It represents a special link and is used to connect *ProcessClass* elements with *DQProcessClass* elements.		
Used in	Navigation model.		
Name	**DQDim**	**Icon**	
Base Class	Class		
Description	It represents a content element. This metaclass corresponds to all of the DQ dimension. It will be responsible for saving the different DQ metadata (*DQDimProperty*).		
Used in	Content model.		
Name	**DQDimProperty**		
Base Class	Property		
Description	A *DQDimProperty* is owned by a DQ dimension and is used to define the DQ metadata related to each DQ dimension (*DQDim*).		
Used in	Content model.		
Name	**DQ_Constraint**	**Icon**	
Base Class	Class		
Description	It represents a content element. This metaclass corresponds to all of the "Constraints" that should be defined and associated with the different DQ dimensions, besides corresponding to bounds (e.g. "*upper_bound*" and "*lower_bound*").		
Used in	Content model.		

Table 2. Stereotypes specification for DQ design elements