A proposal to consider aspects of quality in the software development

CCésar Arturo Guerra García
Information Technology Department
San Luis Potosí, México.

Marco Cardenas Juarez
Faculty of Sciences San Luis Potosí,
México.

Ismael Caballero
Department of Information Technologies
Ciudad Real, Spain.

José Reyes Juárez Ramírez
Autonomous University of Baja California
Tijuana, México.

***Abstract*—Users need trusting in data managed by software applications that are part of Information Systems (IS), which supposes that organizations should assuring adequate levels of quality in data that are managed in their IS. Therefore, the fact that an IS can manage data with an adequate level of quality should be a basic requirement for all organizations. In order to reach this basic requirement some aspects and elements related with data quality (DQ) should be taken in account from the earliest stages of development of software applications, i.e. "*data quality by design*". Since DQ is considered a multidimensional and largely context-dependent concept, managing all specific requirements is a complex task. The main goal of this paper is to introduce a specific methodology, which is aimed to identifying and eliciting DQ requirements coming from different viewpoints of users. These specific requirements will be used as normal requirements (both functional and non-functional) during the development of IS awareness of data quality.**

Keywords—requirements specification; information system development; data quality;

I. Introduction

Many progresses in the field of software quality management at a product and process levels have been done during last decades. As a prove, it is worth highlighting the existence of multiple international standards addressing specifics issues by means of some specific quality models (ISO 9126, ISO 25000, IEEE 1061-1998), software process maturity models (e.g. CMMI and ISO 15504), and standards related to the software verification and validation (IEEE 1012, IEEE 1028, ISO 12207, etc.).

These standards have been widely used in industry for more than twenty years [1]. However, the perception of quality in software depends also on the quality of the data that manages [2-6], to endow these claims, it is worthy to bring here the conclusions raised from some reports [7, 8] coming from some consultancies and vendors that have studied the severe negative impact of inadequate levels of data quality on companies in which software in Information Systems was successfully running [9-11]. As a consequence the importance of the data quality field has grown dramatically. Indeed, this growth is widely accepted, that the new family of standards *ISO/IEC 25000 Software Product Quality Requirements and Evaluation*, which includes the need for dealing with data quality (DQ) as a part to be considered when assessing the level of quality of a software product: "*The target computer system also includes computer hardware, non-target software products, non-target data, and the target data, which is the subject of the data quality model"* [12].

This implies that organizations need to take into account data quality concerns when develop the various software since data is a critical factor [13-15]. In order to do so, we pose such DQ concerns should be addressed from the earliest stages of the software development, as it would be any other software requirements. This would require specific mechanisms and artifacts. Unfortunately, and to the best of our knowledge [16, 17], there are no significant proposals specifically aimed to addressing DQ concerns into the process of developing software.

As researchers in Software Engineering for Data Quality (SE4DQ) as a specialization of "*data quality by design*", our motivation is to provide the necessary mechanisms and artifacts that can help engineers to develop software being aware of such DQ concerns.

To do so, we show the concept of Data Quality Software Requirement (DQSR) as a way to realize the Data Quality Requirement (DQR) into the software [17]. A DQSR is defined as a *software requirement* aimed to satisfying a DQR. The rationale about this definition is the following: we wanted to capture the DQRs that best fit to the data used in each use scenario, and afterwards, derive the corresponding DQSRs that will complement the normal software requirements (both functional and no functional) associated to each one of those use scenarios.

Addressing several Data Quality Software Requirements into a software development process can be a complex task given the existence of serious dependencies and users; impacts and contradictory overlapping effects on both data and process models could also appear. Being conscious of such complexity, and in order to optimize the developing efforts, we introduce DAQURES, *a Methodology for Project Management of Data Quality Requirements Specification*, which is the main contribution of this paper.

The remainder of the paper is organized as follows. Section II provides some concepts on data quality to better understand the paper. DAQURES methodology is presented in Section III. In section IV a brief description of elements and artifacts identified in the methodology is done. Finally, in section V some conclusions are drawn and future work is suggested.

II. Managing DQ Requirements

A. Data Quality Concepts

There are many definitions about the concept of Data Quality [18]; however, the vast majority of authors agree on the view of *fitness for use*: a user has a particular perception of the level of quality of a piece of data should it fit for the purpose of an specific use in a particular task, then the piece of data can be said as having quality [19]. This perception is necessarily multidimensional, because it could be broken down into several criteria, commonly known as Data Quality Dimensions [5]. Each one of the users playing a specific role within the organization by using the software should be able to identify several DQ dimensions that are relevant when assessing the level of quality of the set of data he/she needs to use for the task at hand. The set of DQ dimensions identified by a user will constitute a *Data Quality Requirement*, [17]. It will be necessary to capture the corresponding DQRs for each one of the use scenario and for each one of the users in such scenario.

It is not easy to identify the best fitting DQRs for a purpose [20] since there exists many highly context-dependent ***data quality model*** [21] for the various context: e.g. for medical environments [22, 23], military [24], decision support systems [25, 26], web applications [27, 28], small business [29], and cooperative systems [30]. However, there are some DQ models considered as standard, as the generic one provided by [19].

For the purposes of our investigation we define a DQSR as "*those software requirements originated from one DQR that will allow deriving the necessary features in the software being development to support the required DQ characteristics"*. Here, a DQ characteristic should be understood as the set of features of the software that specifically supports the specific DQR.

The set of such DQ characteristics for software is introduced in a generic way by ISO/IEC 25012 standard [31]. This standard proposes fifteen DQ characteristics from two perspectives: *Inherent* and *System Dependent*. For example, when the DQ characteristic of "Accuracy" appears in the standard, we interpret it as the set of features of the software aimed to warranty that the data being used will be accurate for the specific use in the specific context.

To achieve this warranty, sometimes, the DQSRs will derive into new functionalities that complement existing functional requirement (e.g. when a user is fulfilling a form, some verification can be added to warrantee that all values have been provided for mandatory fields), or other times, they will derive into new non-functional requirements (e.g. choosing the adequate data type to warrantee stored values are accurate enough) or even that they could involve both kind of them.

B. Managing some Requirements for an Information System

The success on the development of a software depends largely on the good elicitation and specification of software requirements by the systems analyst [32-36]. Besides that, as Rizwan and Hussain mentioned [37], it is very important to model both the *requirements* and *data* involved at the moment of creating a Software Requirements Specification (SRS) document. And, it should be taken into account that many users will use the produced software, and the same functionalities of software must meet all of the intended requirements for all of the possible users. In this sense, the method of *Viewpoint-Oriented Requirements Definition (VORD)* as a means of specifying requirements is well recognized [38-41], even when this method is only functionality-oriented. In VORD, the concept of *viewpoint* is analogous to role of users or clients in software. The system delivers functionalities (or services) to viewpoints and the viewpoints pass control information and associated parameters to the system [42]. Thereby, a *viewpoint* is an entity outside the software that generates a requirement (i.e. requirement source).

Each viewpoint has a specific relation with the proposed software; this relation is based upon viewpoint's needs and interactions with the system. Thus, the main focus of VORD is to capture and organize not only global but also the specific requirements provided by each viewpoint. VORD takes into account the different viewpoints, in order to structure and organize the requirements during the elicitation process. The key point of VORD is that it takes in account the existence of many perspectives and provides a framework to manage all different requirements, besides discovering possible conflicts that could appear among proposed requirements by different viewpoints. Given the focus and the main features and advantages of VORD method, we decided to use it as base to develop the DAQURES methodology.

The main steps defined in VORD method [39] are showed in Fig. 1.

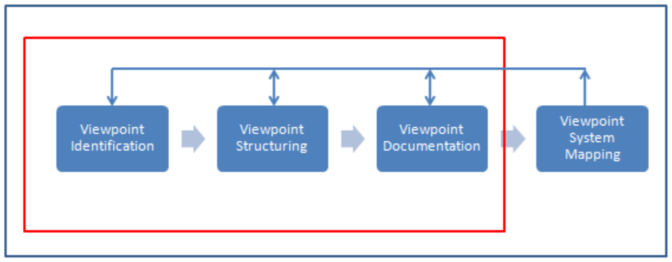

Fig. 1. Steps defined in VORD method.

III. Methodology Supporting Management of DQ Requirements Specification

A. *Objective of Methodology*

The main objective of the *DAQURES methodology* is to guide software developers during the specification and elicitation of Data Quality Requirements. This will require the identification of the data to be used in each one of the possible scenarios, and the corresponding related Data Quality Requirements, which data must meet. For this, it is necessary to consider that each one of the software functionalities will be used by user playing a specific role (called "*viewpoints"* following VORD method nomenclature) who state specific DQRs for the data he uses. All of these stated DQRs should be gathered, and classified and prioritized, and conveniently transformed into Data Quality Software Requirements to be introduced as regular software requirements.

The full description of the DAQURES methodology will be completed with the next information regarding to:

- The different roles involved in the methodology and the activities in which there is a communication between them.
- The artifacts and products that are managed (inputs and outputs of activities, deliverables, etc.).
- The set of activities and steps necessary to carry out a right project management for DQ requirements specification. This description also includes the identification of the involved roles and products to be used and obtained in each activity.
- The catalog of techniques and tools that can be apply during the execution of the different activities of the methodology.

B. *Description of Activities and Steps of Methodology*

This subsection presents the six main activities of DAQURES methodology. These activities are proposed taking in mind the steps defined in the VORD method (showed previously in Fig. 1).

<u>1 PPDQ. Planning of the Project for DQ Software Requirements Specification</u>

This activity pursues the agreement on the DQ Software Requirements Specification and a Plan of Project of Data Quality Requirements Specification. This activity consists of five distinct steps that are shown in Tables I and II, along with the input and output products, and the roles involved for them.

TABLE I. Products, techniques and roles involved for each step defined in activity "Planning of the Project of DQ Software Requirements Specification".

Step	***PPDQ.1. Agreement on a Project for DQ requirements specification***	***PPDQ.2. Celebration of the kickoff meeting of the Project for DQ requirements specification***
Input products	-Specification of the needs expressed by the Customer Enterprise.	- Contract of DQ Software Requirements Specification. - Meeting agenda.
Output products	- Contract of DQ Software Requirements Specification.	- Presentation of project kickoff. - Meeting Minutes.
Techniques or tools	- Interviews. - Meetings.	- Brainstorming. - Meetings.
Roles involved	-Head of Development. -Chief Architect.	- Head of Development. - Chief Architect. - Specification Team. - Analysts / Developers.

TABLE II. Products, techniques and roles involved for "Planning of the Project of DQ Software Requirements Specification".

PPDQ.3. Planning of Project for DQ requirements specification	***PPDQ.4. Approval of the planning of Project for DQ requirements specification***	***PPDQ.5. Configuration management of Project for DQ requirements specification***
- Contract of DQ Software Requirements Specification. - Meeting Minute.	- Plan of Project of DQ Requirements Specification *preliminary version.*	- Each one of document generated during the activities of project.
-Plan of Project of DQ Requirements Specification *preliminary version.*	- Plan of Project of DQ Requirements Specification *approved by both parts*.	- New version of each document generated.
- Meeting. - Observation. -Documentation study. - Software tools: word processors, CASE tools, MS Project, etc.	- Meeting. - Interpersonal negotiation techniques.	- Meeting. -Documentation Study. - Expert Judgment. - Software tools: word processors, CASE tools, MS Project, etc. - Specific software tools for configuration management.
- Chief Architect. - Specification Team.	- Chief Architect. - Head of Development.	-Chief Architect. - DQ Analyst.

Once approved by both parts (e.g. *Customer and Consultant Enterprise*) the first two deliverables documents "*Contract of DQ Software Requirements Specification*" and "*Plan of Project of DQ Requirements Specification*", it is time to go ahead with the activities of elicitation, analysis and specification of DQ requirements.

<u>2 IISV. Identification of Information System Viewpoints</u>

This activity is focused on discovering the distinct *viewpoints* (users) that are going to use the specific software functionalities of the application, besides the identification of the corresponding software functionalities, as well as the DQ requirements (*dimensions*) related. This activity consists of three distinct steps, for each one is showed the intended objective, the input and output products, techniques and tools used and the roles involved (see Table III and IV).

TABLE III. Products, techniques and roles involved for each step defined in activity "*Identification of Information System Viewpoints*".

Step	***ISV.1. Identification of system viewpoints***	***IISV.2. Identification of the most common software functionalities requiring to be complemented with specific DQ requirements***
Input products	- Document of Software Requirements Specification	- List of viewpoints identified able to propose DQ requirements to the system. - List of all functionalities to provide the system. - Document of Software Requirements Specification.

Output products	- List of viewpoints identified able to propose DQ requirements to the system	- List of software functionalities selected that requires satisfying specific DQ requirements from each viewpoint.
Techniques or tools	- Interviews. - Documentation Study. - Brainstorming. - Observation.	- Interviews. - Documentation Study. - Questionnaires. - Brainstorming
Roles involved	- Chief Architect. - DQ Analyst. - Users. - Analyst/Developers.	- Chief Architect. - DQ Analyst. - Users. - Analyst/Developers

TABLE IV. PRODUCTS, TECHNIQUES AND ROLES INVOLVED FOR "*IDENTIFICATION OF INFORMATION SYSTEM VIEWPOINTS*"

Step	***IISV.3. Identification of DQ Requirements for each software functionality***
Input products	- List of viewpoints identified able to propose DQ requirements to the system. - List of software functionalities selected that requires satisfying specific DQ requirements from each viewpoint. - Suitable Data Quality Model.
Output products	- DQR: Instantiation of DQ model for the particular software functionalities under study (including an interpretation on how to achieve each DQ Dimensions for each software functionality).
Techniques or tools	- Interviews. - Work sessions. - Brainstorming.
Roles involved	- Chief Architect. - DQ Analyst. - Users.

3 VS. Viewpoint Structuring

This activity encompasses grouping the functionalities and viewpoints related in a hierarchy, with the objective of organizing them appropriately and resolving possible conflicts between the functionalities and the different viewpoints.

This activity is composed of two steps, the Table V shows their objectives, their input and output products, as well as usable techniques and tools.

TABLE V. PRODUCTS, TECHNIQUES AND ROLES INVOLVED FOR EACH STEP DEFINED IN ACTIVITY "*VIEWPOINT STRUCTURING*".

Step	***VS.1. Classification of Viewpoints***	***VS.2. Classification of DQ requirements***
Input Products	-List of viewpoints identified being able to propose DQ requirements to the system. - Instantiation of DQ model for the particular software functionalities under study.	-List of viewpoints identified able to propose DQ requirements to the system. - Instantiation of DQ model for the particular software functionalities under study. -Hierarchized list of viewpoints and related functionalities.
Output Products	-Hierarchized list of viewpoints and related functionalities.	-List of classified DQ requirements.
Techniques or tools	-Manual of DQ best practices. - Work sessions. -Opinion and Expert Judgment.	- Manual of DQ best practices. - Work sessions. - Expert Judgment.
Roles involved	- Chief Architect. - DQ Analyst. - Use Case Engineer.	- Chief Architect. - DQ Analyst. - Use Case Engineer.

4 DV. Documentation of the Viewpoints

The main goal of this activity is to refine the description of the viewpoints and their software functionalities, besides adding a deep description of the different DQ requirements previously gathered based on the interpretation done for each DQ dimension. This activity is composed of two main steps on which indicate the sought objectives, input and output products, as well as techniques and tools used (see Table VI).

TABLE VI. PRODUCTS, TECHNIQUES AND ROLES INVOLVED FOR EACH STEP DEFINED IN ACTIVITY "*DOCUMENTATION OF THE VIEWPOINTS*".

Step	***DV.1. Documentation of viewpoints***	***DV.2. Documentation of DQ requirements***
Input Products	- List of software functionalities selected to satisfy the specific DQ requirements from each viewpoint. - Hierarchized list of viewpoints and related functionalities. - List of classified DQ requirements. - Document of Software Requirements Specification.	- List of classified DQ requirements. - Hierarchized list of viewpoints and related functionalities. - Document of Software Requirements Specification. - First version of "*Document of System Requirements Specification augmented with DQ Requirements Specification".*
Output Products	- First version of "*Document of System Requirements Specification augmented with DQ Requirements Specification".*	- "*Extended Document of System Requirements Specification augmented with DQ Requirements Specification".*
Techniques or tools	- Software tools: word processors, CASE tools, etc. - Work sessions. - Expert Judgment.	- Software tools: word processors, CASE tools, etc. - Work sessions. - Expert Judgment.
Roles involved	- Chief Architect. - DQ Analyst. - Use Case Engineer.	- Chief Architect. - DQ Analyst. - Use Case Engineer.

5 LSF. Layout of Software Functionalities and DQ Requirements

The main goal of this activity is to extend the modeling of the Software functionalities with those DQSR, which have generated complementing software functionalities. To do so *use case diagrams* or *information case diagrams* can be used [43].

This activity is composed of three different steps on which indicate the sought objectives, input and output products, roles involved, techniques and tools used (see Table VII and VIII).

TABLE VII. PRODUCTS, TECHNIQUES AND ROLES INVOLVED FOR EACH STEP DEFINED IN ACTIVITY "*LAYOUT OF SOFTWARE FUNCTIONALITIES AND DQ REQUIREMENTS*".

Step	***LSF.1. Modeling of software functionalities.***	***LSF.2. Modeling of DQ software requirements.***
Input Products	"*Extended Document of System Requirements Specification augmented with DQ Requirements Specification".*	-"*Extended Document of System Requirements Specification augmented with DQ Requirements Specification.* - Use Case Diagram of IS.
Output Products	Use Case Diagram of the Information System.	Final version of *"Document of System Requirements*

		Specification augmented with DQ Requirements Specification".
Techniques or tools	- Tools for modeling (Rational Rose, Visual Paradigm, Poseidon, ArgoUML, etc.).	- Tools for modeling (Rational Rose, Visual Paradigm, Poseidon, ArgoUML, etc.).
Roles involved	- Use Case Engineer. - Analyst/Developers.	- DQ Analyst. - Use Case Engineer. - Analyst/Developers.

TABLE VIII. PRODUCTS, TECHNIQUES AND ROLES INVOLVED "*LAYOUT OF SOFTWARE FUNCTIONALITIES AND DQ REQUIREMENTS*"

Step	***LSF.3. Validation of results obtained from DAQURES methodology.***
Input products	-Final version of *"Document of System Requirements Specification augmented with DQ Requirements Specification".*
Output products	-Approved final version of "*Document of System Requirements Specification augmented with DQ Requirements Specification".*
Techniques or tools	- Work sessions. - Interpersonal negotiation techniques.
Roles involved	- Chief Architect. - DQ Analyst. - Head of Development. - DQA Engineer.

6 CPDQ. Closing of the Project for DQ Software Requirements Specification.

In this activity, the final report of project is done, besides to carry out the presentation of results to the people involved by Customer Enterprise. This activity is composed of two distinct steps each of which shows the objectives sought, input and output products, the techniques and tools used; besides of roles involved (see Table IX).

TABLE IX. PRODUCTS, TECHNIQUES AND ROLES INVOLVED FOR EACH STEP DEFINED IN ACTIVITY "*CLOSING OF PROJECT OF DQ SOFTWARE REQUIREMENTS SPECIFICATION*".

Step	***CPDQ.1. Elaborating of Final Report of Project for DQ Requirements Specification***	***CPDQ.2. Results presenting of Project for DQ Requirements Specification***
Input Products	-Approved final version of "*Document of System Requirements Specification augmented with DQ Requirements Specification".*	- Final Report of Project of DQ Requirements Specification.
Output Products	- Final Report of Project of DQ Requirements Specification.	- Presenting of results.
Techniques or tools	- Work sessions. - Brainstorming. - Expert Judgment. -Software tools: word processors, CASE tools, etc.	- Meeting. -Software tools: word processors, Power Point, Excel, etc.
Roles involved	- Chief Architect. - DQ Analyst. - DQA Engineer.	- Chief Architect. - DQ Analyst. - DQA Engineer. - Head of Development. - Analyst/Developers. - Users.

IV. CATALOGUE OF ELEMENTS IDENTIFIED

In this section we detailed the elements identified in the activities of the methodology, these elements could be of input, output or both, being besides some of them final deliverables.

Contract of DQ Software Requirements Specification. This document represents the first deliverable (output document given to the Customer Enterprise) of *DAQURES*. It sets out the conditions under which will be made the "Project of DQ Software Requirements Specification", therefore, is a public document for both the Consultant and the Customer Enterprise, and it will require approval by both parties before starting the project.

Plan of Project of DQ Requirements Specification. This document represents the second deliverable of the DAQURES Methodology. The first version of this document is realized previous to start of project, to be included as part of the technical proposal of contract. Being this document a deliverable, the *Plan of Project of DQ Requirements Specification* is public for both the Customer Enterprise and the Consultant Enterprise. The *Plan of Project of DQ Requirements Specification* will be subject to changes during all the project, and on it should be registered the delays and changes of planning made during the project, as well as the changes that can appear in the Specification Team or in the necessities of Customer Enterprise.

Catalogue of specification techniques. This document normally is property of Consultant Enterprise and contains the specification techniques available to be selected in each activities and tasks of DQ requirements specification. Some of these techniques are: interviews, documentation study, brainstorm, questionnaires, work sessions, expert judgment, DQ best practices manual, etc.

Catalogue of specification tools. This document is property of Consultant Enterprise and contains the description of set of tools available to carry out the project. These tools are mentioned in each activities of the DAQURES Methodology, some of them are: Web modelling tools, UML modelling tools, word processors, etc.

Document of System Requirements Specification augmented with DQ Requirements Specification. This artifact represents the third deliverable of the DAQURES methodology, and as such, it should be available for both Enterprises (*Customer and Consultant*). This artifact will be generated as final result once carried out all activities and tasks of DAQURES, will be formed by a document which collects all the DQ requirements associated to each one of functionalities to be implemented into the system. An important characteristic of this document is the necessity of a feedback by the *Customer Enterprise*. Once this document is delivered to the Customer Enterprise, it will dispose of time to study the results and report to the Consultant Enterprise any defect or incoherence detected, in order to clarify possible doubts and improve the document. For this is used the document of "*Modification Request*".

Modification Request. This artifact represents the document whereby the Customer Enterprise once revised some of the deliverables documents, manifests its opinion and possible changes regarding to the obtained results in that documents. This document will be delivered to the Consultant Enterprise, and it will check the document and will update it if necessary, according to the suggested changes. Once modified the document, it will be again delivered to the Customer

Enterprise, who will be able to request a new meeting or solicit a new *Modification Request*, acting just as the first time. The result of this *Modification Request* will be the refinement of this document, as well as the improvement of the own process.

Internal information of Project of DQ Requirements Specification. During the execution of the DAQURES methodology are generated, besides of the own deliverables, a set of documentation related with the own application and execution of the methodology. The output products gotten in each activities and tasks of the methodology will mainly constitute this documentation.

Final Report of Project of DQ Requirements Specification. This artifact will be generated as final result of project. This document gathers the conclusions once analyzed the reached results. By other hand, the *Final Report* is also composed of a presentation of results, which will be directed primarily to the Head of Development and Analysts/Developers, and if necessary to some Directive member of the Customer Enterprise.

V. Conclusions

Through the last two decades, the variety and complexity of software has grown dramatically, all of them focused to satisfy any kind of business processes into the organizations and enterprises. Likewise, the data used by software has become into a key assets in any type of organization. However, people using day by day the software have to be sure that data they are consuming has an adequate level of quality for the use they require. Unfortunately, there are no proposals that include issues that would address the management of DQ software requirements into the Information System development. A first approach to addressing this problem is shown in this paper, which presents a methodology for projects management of data quality requirements specification. This methodology could help developers of Information Systems, to carry out a proper elicitation and specification of specific Data Quality requirements defined by different types of roles (named *viewpoints*) that interact with an Information System. It can be understood as a guide that analysts can follow at the moment of writing a Requirements Specification document complemented with management of Data Quality. The purpose of specifying data quality requirements from the initial stage, it is easier for developers to be aware of the quality of data that needs to be implemented for each functionality throughout the Information System development process. As part of our future work, and considering the advantages provided by the Model Driven Architecture approach, mainly focusing on their capabilities of abstraction and modeling characteristics, it will be possible to ensure a much easier integration of our results in the DQ aware-IS development with other tools and methodologies of software development.

References

1. Díaz-Ley, M., F. García, and M. Piattini, *MIS-PyME software measurement capability maturity model - Supporting the definition of software measurement programs and capability determination.* Advances in Engineering Software, 2010. **41**(10-11): p. 1223-1237.
2. Caballero, I., et al., *IQM3: Information Quality Maturity Model.* Journal of Universal Computer Science, 2008. **14**: p. 1-29.
3. Guerra-García, C., et al. *DAQ_UWE: A Framework for Designing Data Quality Aware Web Applications*. in *International Conference of Information Quality, ICIQ´11*. 2011. Adelaide, Australia.
4. Maydanchik, A., *Data Quality Assessment*, ed. T. Publications. 2007.
5. Wang, R.Y., *A Product Perspective on Total Data Quality Management.* Communications of the ACM, 1998. **41**(2): p. 58-65.
6. Wang, S. and H. Wang, *Information quality chain analysis for total information quality management.* International Journal of Information, 2008. **2**(1): p. 4-15.
7. Karr, A.F., A.P. Sanil, and D.L. Banks, *Data quality: A statistical perspective.* Statistical Methodology, 2006. **3**(2): p. 137-173.
8. Russom, P. *Taking Data Quality to the Enterprise through Data Governance*. 2006.
9. Oracle, *State of the Data Integration Market 2009-2009*, 2008. p. 1-29.
10. Thi, T.T.P., et al., *InfoGuard: A Process-Centric Rule-Based Approach for Managing Information Quality*, in *European Research Consortium for Informatics and Mathematics ERCIM*2010. p. 55-56.
11. ThomsonReuters and Lepus *Thomson Reuters and Lepus Survey reveals Data Quality and Consistency Key to Risk Management and Transparency*. 2010.
12. ISO/IEC, *ISO/IEC 25000 Software and system engineering – Software product Quality Requirements and Evaluation (SQuaRE) –Guide to SQuaRE*, 2005, International Organization for Standarization: Geneva, Switzerland.
13. Akoka, J., et al. *A Framework for Quality Evaluation in Data Integration Systems*. in *International Conference on Enterprise Information Systems, ICEIS*. 2007.
14. Otto, B., Y. Lee, and I. Caballero, *Information and data quality in networked business.* Electronic Markets, 2011: p. 1-3.
15. Otto, B. and A. Schmidt. *Enterprise Master Data Architecture: Design Decisions and Options*. in *International Conference on Information Quality, ICIQ*. 2010. Little Rock, USA.
16. Guerra-García, C., I. Caballero, and M. Piattini, *A Survey on How to Manage Specific Data Quality Requirements during Information System Development.* Communications in Computer and Information Science, 2011. **230**(Evaluation of Novel Approaches to Software Engineering): p. 16-30.
17. Guerra-García, C., I. Caballero, and M. Piattini, *Capturing data quality requirements for web applications by means of DQ_WebRE.* Information Systems Frontiers, 2013.
18. Batini, C., et al., *Methodologies for data quality assessment and improvement.* ACM Computing Surveys, 2009. **Vol. 41, No. 3**.
19. Strong, D.M., Y.W. Lee, and R.Y. Wang, *Data Quality in Context.* Communications of the ACM, 1997. **40**(5): p. 103-110.
20. Wand, Y. and R.Y. Wang, *Anchoring Data Quality Dimensions in Ontological Foundations.* Communications of the ACM, 1996. **39**(11): p. 86-95.
21. Batini, C. and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. 2006, Berlin: Springer-Verlag Berlin Heidelberg.
22. Davidson, B., Y.W. Lee, and R.Y. Wang, *Developing data productions maps:meeting patient discharge data submission requirements.* International Journal of Healthcare Technology and Management, 2004. **6**(2): p. 223-240.
23. Kosar, D. *Developing a Framework to Manage Data Quality in Healthcare* in *Fourth International Conference on Information Quality (ICIQ'99)*. 1999. MIT, Cambridge, MA, USA.

24. Burzynski, T. *Establishing the Environment for Implementation of a Data Quality Management Culture in the Military Health System*. in *Third International Conference on Information Quality (ICIQ'98)*. 1998. MIT, Cambridge, MA, USA.

25. Gendron, M. and M.J. D'Onofrio. *Formulation of a Decision Support Model Using Quality Attributes*. in *Seventh International Conference on Information Quality (ICIQ'02)*. 2002. MIT, Cambridge, MA, USA.

26. Shankaranarayan, G., M. Ziad, and R.Y. Wang, *Managing Data Quality in Dynamic Decision Environments: An Information Product Approach.* Journal of Database Management, 2003. **14**(4): p. 14-32.

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

28. Eppler, M. and P. Muenzenmayer. *Measuring Information Quality in the Web Context: A Survey of State-of-the-Art Instruments and an Application Methodology*. in *Proceeding of the Seventh International Conference on Information Quality*. 2002.

29. Leonowich-Graham, P. and M.J. Willshire. *A Data Quality Framework for Small Business*. in *Eighth International Conference on Information Quality (ICIQ'03)*. 2003. MIT, Cambridge, MA, USA.

30. Mecella, M., et al., *Managing Data Quality in Cooperative Information System*, in *CooPI/DOA/ODBASE*, R. Meersman and Z. Tari, Editors. 2002, Springer-LNCS 2519. p. 486-502.

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

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

33. Lowe, D. and J. Eklund, *Client Needs and the Design Process in Web Projects.* Journal of Web Engineering, Rinton Press, US, 2002. **Vol. 1, No. 1**.

34. Nicolás, J. and A. Toval, *On the generation of requirements specifications from software engineering models: A systematic literature review.* Inf. Softw. Technol., 2009. **51**(9): p. 1291-1307.

35. Nuseibeh, B. and S. Easterbrook, *Requirements engineering: a roadmap*, in *Proceedings of the Conference on The Future of Software Engineering* 2000, ACM: Limerick, Ireland p. 35-46

36. Pressman, R., *Software Engineering: a Practitioner's Approach. 5/e*. Fifth ed. 2001: McGraw-Hill.

37. Rizwan Jameel Qureshi, M. and S.A. Hussain, *An adaptive software development process model.* Advances in Engineering Software, 2008. **39**(8): p. 654-658.

38. Darke, P. and G. Shanks, *Stakeholder viewpoints in requirements definition: A framework for understanding viewpoint development approaches.* Requirements Engineering, 1996. **1**(2): p. 88-105.

39. Kotonya, G. and I. Sommerville, *Requirements engineering with viewpoints.* Software Engineering Journal, 1996.

40. Kotonya, G. and I. Sommerville, *Requirements Engineering. Processes and Techniques.*, ed. J.W. Sons. 2002.

41. Leite, J.C.S.d.P. and P.A. Freeman, *Requirements Validation Through Viewpoint Resolution.* IEEE Trans. Softw. Eng., 1991. **17**(12): p. 1253-1269.

42. Kotonya, G., *Practical Experience with Viewpoint-Oriented Requirements Specification*, in *Requirements Engineering*, S. London, Editor. 1999. p. 115-133.

43. 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.