

IP-UML: TOWARDS A METHODOLOGY FOR QUALITY IMPROVEMENT BASED ON THE IP-MAP FRAMEWORK

(Research Paper)

Monica Scannapieco

Universita' di Roma, "la Sapienza", Rome, Italy

IASI-CNR, Rome, Italy

monscan@dis.uniroma1.it

Barbara Pernici

Politecnico di Milano, Milan, Italy

barbara.pernici@polimi.it

Elizabeth Pierce

Indiana University of Pennsylvania, USA

empierce@grove.iup.edu

Abstract. In this paper, a UML profile for data quality is proposed with the aim of supporting quality improvement inside a single organization. The profile is based on the IP-MAP framework and has the advantage to give a formal definition to the main concepts related to the management of quality improvement, as well as to organize such concepts into a set of models useful to realize a software system. A methodology to improve data quality is also discussed. Specifically, a process to produce the UML artifacts designed by the data quality profile is specified and the initial steps of a pattern-based technique to design quality improvement are also presented.

Key Words: Data Quality, Improvement, Methodology, UML, IP-MAP

1 INTRODUCTION

Data quality improvement inside a single organization is an issue inherently very complex. The proof of this complexity can be found in the various methodologies dealing with this issue that have been published both by researchers ([14], [10]) and practitioners ([3], [8]). When adopting one of these methodologies, one of the major problems is in the lack of a modeling language that can help throughout all the methodology phases. Such a language should be formal enough to allow a unique interpretation of the language constructs, i.e. a precise semantics should be defined for it. At the same time, it should be easily understood also by people who do not have scientific and technical skills; this is because one of the most important activities when designing for quality improvement is the interaction with customers in order to find out the actual quality requirements. In the software engineering area, a language that has both characteristics is the Unified Modeling Language (UML). In fact, UML has a defined semantics, and perhaps the principal cause of its success lies in its understandability even for non-technical people. However, the UML semantics is intentionally general. This is because different application domains and

systems need different specialization and the applicability of UML would have been constrained if it could not support such diversity. The specialization of UML to different domains has already been realized by the proposals of different *profiles*, i.e. extensions of UML for specific purposes. Some examples of these profiles can be found in [6], and are the “UML Profile for Software Development Processes” and the “UML Profile for Business Modeling”.

The goal of this paper is to suggest a *UML profile for data quality*. This profile is inspired by the concepts of the IP-MAP framework, proposed in [10]. The IP-MAP is an extension of the Information Manufacturing System (IMS) proposed by Ballou et al. in [1]. In our opinion, this framework has the major advantage of combining both a data analysis and a process analysis in order to assess the quality of data. Data are considered as a product, i.e. Information Product (IP), and the processes creating data are analyzed in order to point out quality bottlenecks in manufacturing information products. The IP-MAP framework already proposes a modeling formalism, which is based on Data Flow Diagrams. The use of UML instead of such a formalism can provide the following advantages: UML is a *standard language* that is widely diffused and a lot of CASE tools for it have been realized; UML is a language thought for *analysis, design and implementation* artifacts, this means the possibility of bridging the gap between the initial phases of an improvement process and the actual implementation, by using the same language in all the phases of the process; finally, the *expressive power* of UML is greater, especially with reference to the process modeling constructs, as detailed in the following of the paper.

In addition to a UML profile for data quality, we also propose a *methodology* that is based on the IP-MAP framework but differs from this framework because: (i) it specifies the artifacts to produce during the improvement process in terms of diagrams drawn by using the UML elements defined in the profile for data quality; (ii) it uses the IP-MAPs not only in order to assess quality and *think* of improvement actions but also as a diagrammatic way to *design* and *realize* improvement actions.



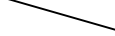





The structure of the paper is as follows. In Section 2, a brief overview of the IP-MAP framework and of UML extension mechanisms is described. In Section 3, the UML profile for data quality is proposed. Section 4 describes a UML and IP-MAP based methodology for quality improvement. In Section 5, an architecture supporting the methodology is defined. Finally, Section 6 concludes the paper by drawing future work.

2 IP-MAP AND UML BASICS

2.1 IP-MAP Framework

An Information Production Map (IP-MAP) is a graphical model designed to help people to comprehend, evaluate, and describe how an information product such as an invoice, customer order, or prescription is assembled. The IP-MAP is aimed at creating a systematic representation for capturing the details associated with the manufacture of an information product. IP-MAPs are designed to help analysts visualize the information production process, identify ownership of the process phases, understand information and organizational boundaries, and estimate time and quality metrics associated with the current production process.

There are eight types of construct blocks that can be used to form the IP-MAP. Each construct block is identified by a unique and non-null name and is further described by a set of attributes (metadata). The content of this metadata varies depending on the type of construct block.

1. **Source (raw input data) block:**  This block is used to represent the source of each raw (input) data that must be available in order to produce the information product expected by the consumer.
2. **Customer (output) block:**  This block is used to represent the consumer of the information product. The consumer specifies in this block the data elements that constitute the "finished" information product.
3. **Data Quality block:**  This block is used to represent the checks for data quality on those data items that are essential in producing a "defect-free" information product.
4. **Processing block:**  This block is used to represent any manipulations, calculations, or combinations involving some or all of the raw input data items or component data items required to ultimately produce the information block. When the processing block is used for the specific purpose of cleansing or correcting input data items then this block becomes known as the data correction block.
5. **Data Storage block:**  This block is used to represent data items in storage files or databases so that they can be available for further processing.
6. **Decision block:**  In some complex information manufacturing systems, depending on the value of some particular data items(s), it may be necessary to direct the data items to a different set of blocks downstream for further processing. In such cases, a decision block is used to capture the different conditions to be evaluated and the corresponding procedures for handling the incoming data items based on the evaluation.
7. **Business Boundary block:**  The business boundary block is used to specify the movement of the information product across departmental or organizational boundaries.
8. **Information System Boundary block:**  This block is used to reflect the changes to the raw input data items or component data items as they move from one information system to another type of information system. These system changes could be intra or inter-business units. There are circumstances where the raw input data items or component data items go through both a business boundary and a system boundary change. The **combined business-information system boundary block** is defined for this purpose.

2.2 UML and Extension Mechanisms

The Unified Modeling Language (UML) is used in many different application domains and is increasingly becoming the *de facto* standard language for object-oriented analysis and design. An interesting overview of the UML origin is described in [5]; now UML is at the version 1.4 (May 2001) [6].

UML definition is very general so that it can be used in every kind of application domain; in addition, it can be extended in order to cope with peculiar aspects of specific systems and domains. Some standard extension mechanisms are provided, namely: constraints, tag definitions and tagged values, and stereotypes. The specification of UML analysis and design elements is based on the notion of *model element*, defined as an abstraction drawn from the system being modeled; the principal model elements are classes and relationships.

A *constraint* is a semantic restriction that can be attached to a model element; it can be expressed: (*i*) in

an informal language, when the interpretation must be done by a human; *(ii)* in a formal language, in order to be automatically interpreted. In UML diagrams, constraints are enclosed in braces.

A *tag definition* specifies new kinds of properties that may be attached to model elements.

A *tagged value* specifies the actual values of tags of individual model elements.

A *stereotype* is a model element that characterizes model elements through a precise semantics.

According to the UML 1.4 specification [6], “a coherent set of such extensions, defined for a specific purpose, constitutes a *UML profile*”.

In the following of this paper, we focus on stereotypes and constraints as extension mechanisms, in order to define a UML profile for data quality improvement inside a single organization. The profile, described in Section 3, is based on the IP-MAP framework and its main objective is to give a formal definition to the main concepts related to the management of quality improvement, as well as to organize such concepts in a set of models useful to realize a system for quality improvement.

2.3 A Running Example

We consider as an explanatory example the same of [7]. For the purposes of this example, the information product is a mailing label. A school called Big State University uses these mailing labels to send out publications to its alumni. Incorrect or out-of-date mailing labels are a problem for the university. After the end of each academic year, data (including current address information) about graduating seniors are taken from the Big State University's active student database and transferred to the Alumni database. Alumni are encouraged to send name/address corrections and changes to Alumni Affairs so that their address information can be kept up to date. The secretary at Alumni Affairs records this information into the Alumni database on a weekly basis. Unfortunately, only about 1 in 10 alumni remember to inform Big State University of their name and address changes. When it is time for Big State University to send out an alumni publication, Alumni Affairs runs a program to create a list of mailing labels, which are then pasted onto the outgoing publication by the University Mail Service. The goal of Big State University is to improve the quality of mailing labels, in order to reduce the percentage of undelivered mail due to out-of-date or incorrect mailing labels.

3 DATA QUALITY PROFILE

In this section, the Unified Modeling Language is extended in order to allow modeling and realizing data quality improvement. The starting concepts are the ones defined in the IP-MAP framework; the result of such an extension is a UML profile called *Data Quality profile*. The Data Quality profile consists of three different models, namely: the Data Analysis Model described in Section 3.1, the Quality Analysis Model described in Section 3.2 and the Quality Design Model described in Section 3.3.

The three models derive from the idea of improving the quality of data by: *(i)* identifying which data are of interest, what is their composition and what is their derivation in the context of the production process (Data Analysis Model); *(ii)* identifying the quality requirements for each data type (Quality Analysis Model); and *(iii)* modeling data and processes together in order to verify the quality requirement satisfaction in the context of the processes producing data, as well as in order to model improvement actions in terms of the processes necessary to support quality improvement (Quality Design Model).

3.1 Data Analysis Model

The Data Analysis Model specifies which data are important for consumers, as their quality is critical for organizations success. It distinguishes among: information product, raw data that are parts of the information product and component data that are semi-processed information that contribute to the manufacture of the information product. Each of these elements is represented as a stereotyped UML class, namely:

- ❑ <<informationProduct>>. An Information Product class is a class labeled with this stereotype that represents an information product.
- ❑ <<rawData>>. A Raw Data class is a class labeled with this stereotype that represents a raw data related to a specific information product.
- ❑ <<componentData>> A Component Data class is a class labeled with this stereotype that represents a component data related to a specific information product.

We also introduce a further stereotyped class:

- ❑ <<qualityData>>. A Quality Data class is a class labeled with this stereotype that generalizes Information Product classes, Raw Data classes, and Component Data classes.

The relationships among these elements are shown in the schema depicted in Figure 1; this is a meta-schema, i.e. a schema defining the elements to be used in the UML schemas modeling the application systems. In the figure, an Information Product class is an *aggregation* of Raw Data classes and it has a *dependency* relationship with Component Data classes, which means that if component data change, therefore also the information product will change. A *dependency* relationship also connects the Component Data class with the Raw Data class. Moreover, a Quality Data class has a *generalization* relationship with Information Product classes, Raw Data classes and Component Data classes.

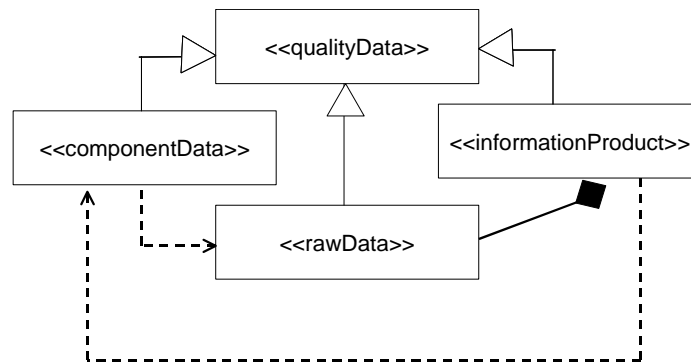


Figure 1: Classes in the Data Analysis Model.

Notice that information products, raw data and component data are all concepts defined in the IP-MAP framework; instead, the concept of quality data has been introduced in order to provide a higher abstraction level when generally referring to data “interesting” from a data quality perspective, i.e. for which quality requirements may be specified. Moreover, the UML formalization of such concepts also helps to understand the relationships and dependencies among the different elements.

3.2 Quality Analysis Model

The Quality Analysis Model consists of the modeling elements that allow for representing quality requirements of data.

A quality requirement can be related to one of the quality dimensions or characteristics that are typically defined for data quality. We consider the set of dimensions proposed in [13], and also adopted in the IP-

MAP framework. Such a set consists of four categories: accessibility, believability, interpretability and usefulness. Each category includes a set of dimensions for a total of 21 dimensions.

Our idea is to model the overall set of dimension-related requirements as a hierarchy of stereotyped classes, all of which are subclasses of a Quality Requirement class. The following stereotypes need to be introduced:

- ❑ <<qualityRequirement>>. A Quality Requirement class is a class labeled with this stereotype and generalizing the set of quality requirements that can be specified on a Quality Data class.
- ❑ <<qualityAssociation>>. A Quality Association is an association relationship labeled with this stereotype and associating a Quality Requirement class with a Quality Data class. Quality requirements on data need to be verified, such that, in case they are not satisfied, improvement actions can be taken; therefore a *constraint* is specifically introduced on the Quality Association.

The meta-schema defining the elements of the Quality Analysis model is shown in Figure 2; specifically, the hierarchy of stereotyped classes specializing a Quality Requirement class is depicted.

The specification of a distinct stereotype for each quality requirement has the advantage of clearly fixing the *types* of requirements that can be associated to data. Moreover, in future work the possibility of considering a *measure* class for each requirement (in a way similar to the one proposed in [12]) and the opportunity of formalizing constraints by using the Object Constraint Language (OCL)¹ [15] will be investigated.

An example of a diagram of the Quality Analysis Model related to our running example is shown in Figure 3; the accuracy and currency quality requirements are specified on the Quality Data classes *Mailing Label* and *Name*.

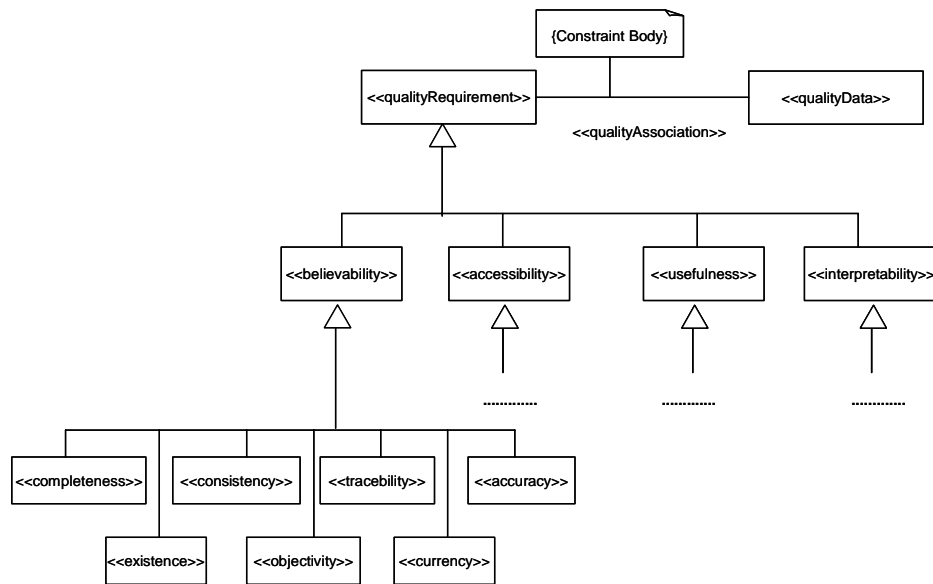


Figure 2: Classes and associations in the Quality Analysis Model.

3.3 Quality Design Model

The quality design model specifies IP-MAPs. An IP-MAP helps in understanding the details associated with the manufacture of an information product. It shows the processes managing data, as well as how

¹ The Object Constraint Language (OCL) is part of the UML and can be used to specify all kinds of constraints, pre- and post-conditions, guards etc. over the objects in the different models.

information products are produced starting from raw data and component data. Such a combined process and data analysis allows to achieve two objectives: (i) to detect quality potential problems and, after having checked the non-conformance to quality requirements, (ii) to introduce quality improvement actions.

The IP-MAP dynamic perspective, in which processes are described together with exchanged data, can be obtained by combining *UML activity diagrams* with *UML object flow diagrams*.

Activity diagrams are a special case of state diagrams in which “all (or at least most) of the states are action or subactivity states and in which all (or at least most) of the transitions are triggered by completion of the actions or subactivity in the source states” [9].

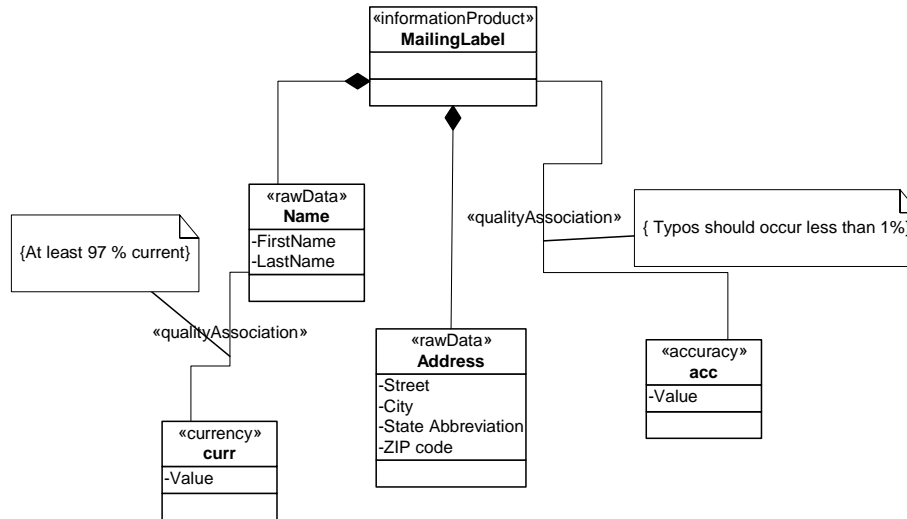


Figure 3: A diagram of the Quality Analysis Model of the running example.

Object flows are diagrams in which “objects that are input or output from an action may be shown as object symbols. A dashed arrow is drawn from an action state to an output object, and a dashed arrow is drawn from an input object to an action state” [9].

The following UML extensions need to be introduced, in order to represent IP-MAP elements:

- ❑ Stereotyped activities, to represent processing and data quality blocks.
- ❑ Stereotyped actors, to represent customer, source and data storage blocks.
- ❑ Stereotyped dependency relationships, in order to give a precise semantics to the relationships between some elements.

In Table 1, the detail of all the stereotypes with the associated descriptions is listed.

Notice that:

- ❑ the Decision Block, included in the IP-MAP specification, is simply the standard *decision construct* of UML activity diagrams.
- ❑ The Information System Boundary Block and the Business Boundary Block are represented by the standard concept of *swimlanes*².

The IP-MAP as described in [10] is very similar to Data Flow Diagrams. It is also based on the same approach as such a language, i.e. the specific focus on *data*. Instead, some characteristics specifically related to processes, such as process synchronization, also need to be captured when modeling for quality improvement. UML activity diagrams and object flows allow from one hand to maintain the IP-MAP idea of combining data and processes, but also enrich the process modeling power by exploiting the

² “Actions and subactivities may be organized into *swimlanes*. Swimlanes are used to organize responsibility for actions and subactivities” [9].

synchronization constructs included in the activity diagrams specification.

In Figure 4, an example of a diagram of the Quality Design Model related to the running example is shown.

Stereotype	Base Class	Description
<<processing>> Processing Activity	activity	It represents IP-MAP processing block.
<<quality>> Quality Activity	activity	It represents IP-MAP quality block.
<<customer>> Customer Actor	actor	It represents IP-MAP customer block.
<<source>> Source Actor	actor	It represents IP-MAP source block.
<<dataStorage>> Data Storage Actor	actor	It represents IP-MAP data storage block.
<<L/E>> Load/Extract Dependency	dependency	The two elements of the relationship have the role of Loader/Extractor and of the source from which loading/extracting

Table 1 : Stereotypes to model IP-MAP constructs.

4 IP-UML: TOWARDS A METHODOLOGY FOR DATA QUALITY IMPROVEMENT

In this section, we propose a methodology for data quality improvement, called *IP-UML*. A methodology needs to specify [11]:

- ❑ what modeling language should be used in describing the analysis and design work;
- ❑ a development process, that is a set of rules which defines how a development project should be carried out;
- ❑ some techniques concerning how to produce the project artifacts (e.g., analysis model, design model, documents, etc.).

IP-UML uses the Data Quality profile, defined in the previous section, as the modeling language. The process and a technique to design quality improvement actions are described in the two Sections 4.1 and 4.2, respectively.

4.1 A Process to Improve Data Quality

We propose a process consisting of three distinct phases: Data Analysis, Quality Analysis and Quality Improvement Design, as shown in Figure 5. The Data and Quality Analysis phases are inspired by the IP-

MAP framework, and are simply detailed by the specific UML artifacts that should be produced in each phase in conformance to the Data Quality profile. The Quality Improvement Design phase consists of two distinct sub-phases, namely: the Quality Verification phase and the Quality Improvement phase. The former is inspired by the IP-MAP framework; instead, the latter has been introduced with the specific aim of using the IP-MAPs also for explicitly modeling improvement processes.

4.1.1 Quality Analysis Phase

The Quality Analysis phase implies the specification of the required quality for the identified data in terms of quality dimensions. For each of these dimensions, the Data Quality profile introduces stereotyped classes associated to a Quality Data class, according to what defined in Section 3.2.

The result of this phase is a *Quality Analysis model*, which consists of a set of *class diagrams*, describing Quality Data classes (that may be information product or raw data) with the specified quality requirements.

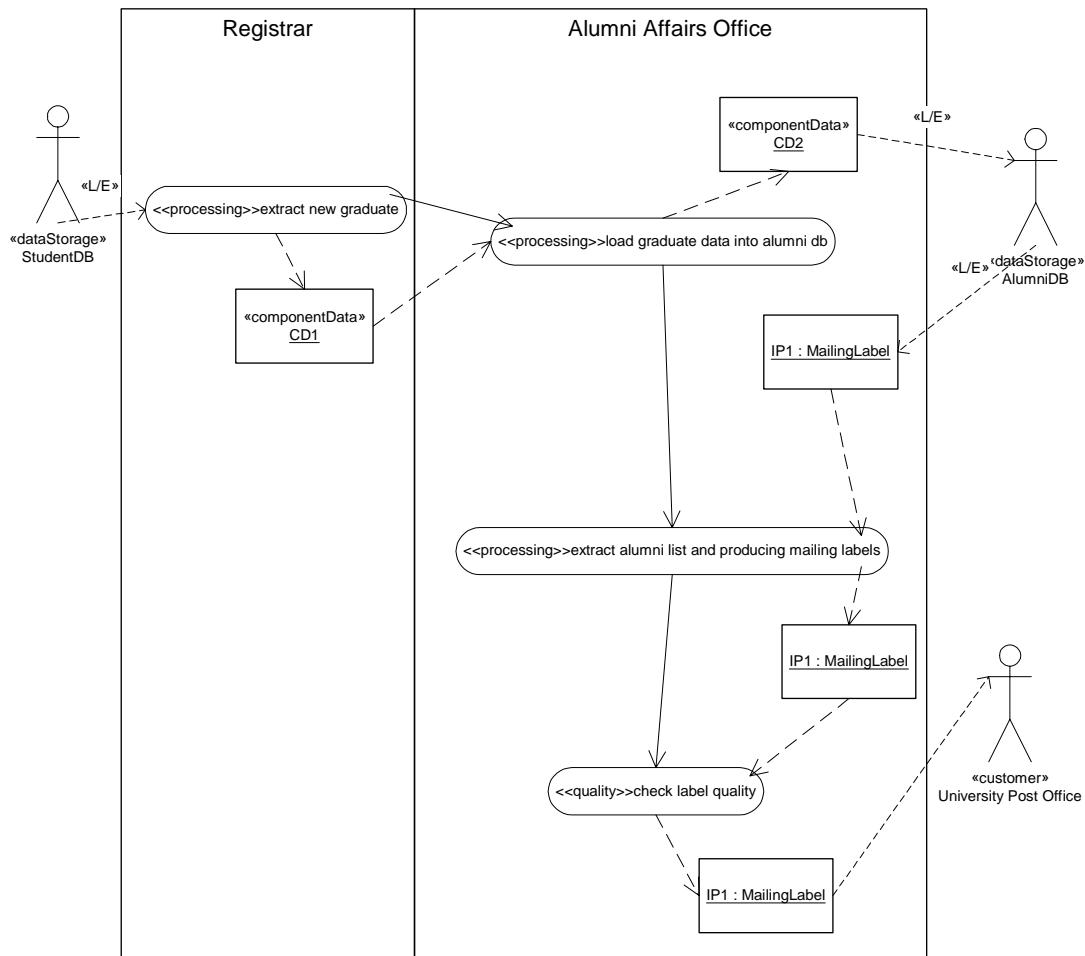


Figure 4: A diagram of the Quality Design Model of the running example.

4.1.2 Quality Improvement Design Phase

The Quality Improvement Design phase can be divided into two sub-phases:

- Quality Verification
- Quality Improvement

In the Quality Verification phase, the process by which the information product is manufactured is

described. Moreover, quality bottlenecks of such a process are detected and quality checks are introduced in order to verify the conformance to requirements. If such requirements are not satisfied, quality improvement actions need to be engaged. The result of this sub-phase is a set of *activity diagrams*, in which *object flows* are described in order to model data exchanges and manipulations. The activity diagrams are drawn by using the stereotyped activities, actors and relationships introduced in Section 3.3. In the Quality Improvement phase, improvement actions need to be defined in detail. Specifically, this phase implies the design of *processes realizing quality improvement*. Such a design can be supported by *quality improvement patterns*, a technique to reuse previous design experiences in quality improvement, which is detailed in the next section. As in the Quality Verification phase, activity diagrams and object flows are drawn by using the constructs introduced in Section 3.3.

Diagrams of both the Quality Verification and the Quality Improvement phases constitute the *Quality Design Model*.

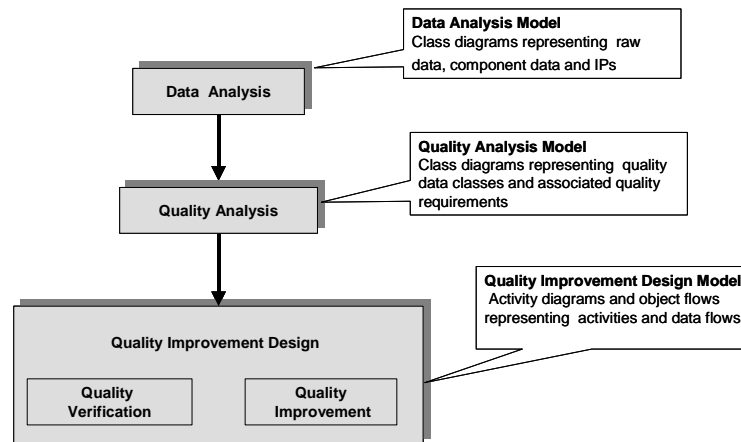


Figure 5: Phases of the process with artifacts.

4.2 Quality Improvement Patterns

Quality improvement is a complex activity that typically requires investments in terms of money and of people skills. The reuse of solutions and experiences can be very useful in supporting quality improvement, and can reduce time and costs considerably.

The basic idea is to provide some *quality improvement patterns* as a technique to be used to design quality activities. A design pattern generally consists of four elements [4]: (i) the name, necessary for having a vocabulary of patterns; (ii) the problem, describing when to apply the pattern; (iii) the solution, providing an abstract description of the pattern and (iv) the consequences, i.e. the results deriving from the pattern application.

In the following, we define an *Event Notification Pattern*. In future work, our idea is to support quality improvement design activities by a set as complete as possible of quality improvement patterns.

NAME: Event Notification Pattern

PROBLEM: In many cases, out-of-date information in a given data source is due to lack of communication with other data sources that store the up-to-date information.

SOLUTION: Each time an event in the real world occurs the storage of which is in care of a specific organization, all the organizations that are interested in this event are to be *notified* about this. It is possible to realize this by an event-based notification mechanism based on the Publish and Subscribe

(P&S) paradigm [2]. According to this paradigm, a number of message channels are defined, each carrying messages of a specific type. Individual organizations may subscribe to these channels and start receiving the messages that are multi-cast over the channel, or send messages of their own. Using this event-based model, all the organizations can notify and be notified about events of common interest. When an organization subscribes to one of these channels, it is notified of the change and can promptly *react* to it by updating its own data.

CONSEQUENCES: This idea comes in very handy for quality improvement, because it allows a change in one data source to propagate quickly to other data sources that may be affected by that change, thereby improving the overall currency of the data.

EXAMPLE: Considering our running example, the currency of mailing labels can be improved by making the Alumni Affairs Office to “subscribe” to two types of events:

- Address Change Event
- Death Event

Assuming that two organizations, let’s say a City Council and a Mortuary, are responsible for the “publishing” of such events, the result is that each time a new event of the defined types is published by the City Council or the Mortuary, the Alumni Affairs Office is notified about this.

In Figure 6, a diagram shows the pattern applied to the running example; each processing activity is assigned to a specific organization by means of swimlanes and “join” transition are used in order to synchronize activities.

5 IMPLEMENTING QUALITY IMPROVEMENT

The quality improvement process can be usefully supported by the software architecture depicted in Figure 7.

The *Quality Modeler* is a UML CASE tool that supports the modeling activities of the IP-UML methodology. It allows modeling according to the Data Quality profile and by following the proposed process for quality improvement.

The *Quality Checker* is a module implementing the quality checks designed in the Quality Verification phase of the process for quality improvement. It interacts with:

- A *Quality Repository*, in which it reads the actual values of the quality requirements, specified by the customer and modeled by the Quality Analysis Model.
- A (set of) *Data Repository* in which data values populating the Data Analysis Model are stored.
- *External Quality Measurement tools* necessary to measure the actual quality dimension values of data.

The *Quality Checker* has the role of comparing the specified requirements with actual quality values. In all the cases in which the quality requirements are not satisfied, the *Quality Checker* sends feedbacks to the *Quality Modeler* in order to enact and guide the design of the quality improvement activities.

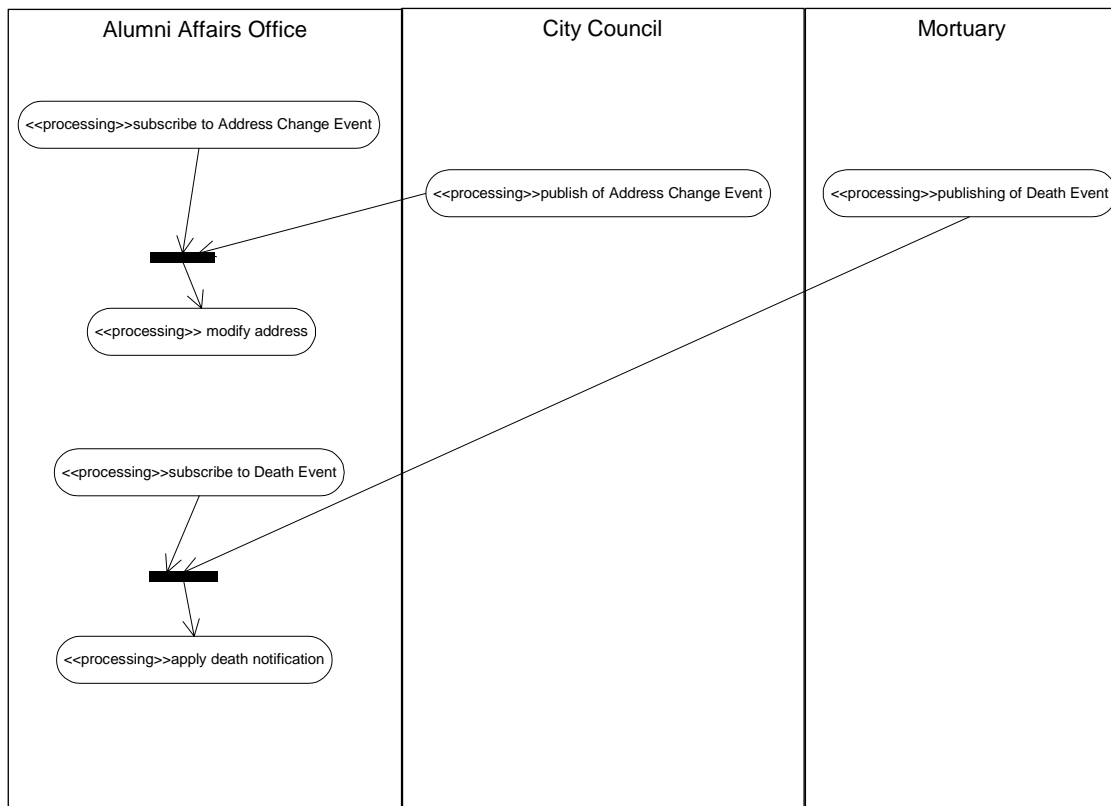


Figure 6: Example of the Event Notification Pattern instantiation.

6 CONCLUSIONS AND FUTURE WORK

The main contribution of this paper is in the proposal of a software engineering approach in order to improve data quality in a single organization. The proposed approach does not set aside the results already published in the data quality literature; instead it relies on one of them, namely the IP-MAP framework.

Specifically, in this paper a UML profile for data quality is proposed. Such a profile needs to be extended in different directions. A possible extension regards the modeling of metadata associated with the constructs in the IP-MAP framework. They may be very useful for resolving issues related to the measurement of the quality of the IP. From an architectural point of view, quality measurement tools exploiting the storage of such metadata could be designed. A further extension to the profile regards the modeling of possible metrics to use in the quality requirements specification and in the quality values measurement. Formalizing quality requirements by means of the Object Constraint Language could also be very useful to support the verification phase.

The IP-UML methodology, described in Section 4, also needs to be completed, its effectiveness must be proved and a complete set of quality improvement patterns will be designed and experimented. A prototype of the architecture proposed in Section 5 will also be implemented. The huge availability of UML case tools simplify the task of realizing the Quality Modeler, though remaining the need of implementing a plug-in specific for the data quality profile.

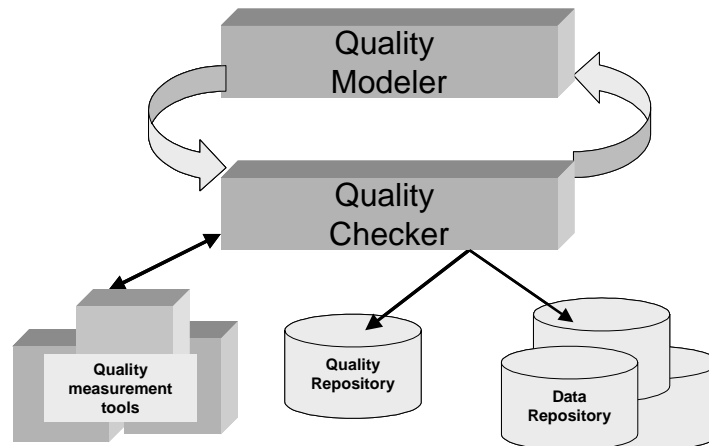


Figure 7: An architecture to support quality improvement.

REFERENCES

- [1] Ballou D. P., Wang R. Y., Pazer H. and Tayi G.K., “Modeling Information Manufacturing Systems to Determine Information Product Quality”, *Management Science*, 44(4), 1998.
- [2] Cugola G., Di Nitto E., Fuggetta A.: “Exploiting an event-based infrastructure to develop complex distributed systems”. In Proceedings of the *20th International Conference on Software Engineering (ICSE 98)*, Kyoto, Japan, 1998.
- [3] English L.: *Improving Data Warehouse and Business Information Quality*, Wiley & Sons, 1999.
- [4] Gamma E., Helm R., Johnson R., Vlissides J.: *Design Patterns- Elements of Reusable Object Oriented Software*. Addison Wesley, 1995.
- [5] Kobryn, C.: UML 2001: “A Standardization Odyssey”. *Communications of the ACM*, 42(10), 1999.
- [6] OMG: *Unified Modeling Language Specification*, Version 1.4, September 2001.
- [7] Pierce E. M.: “Using Control Matrices to Evaluate Information Product Maps”. In Proceedings of the *7th Conference on Information Quality*. Boston, MA, 2001.
- [8] Redman T.C.: *Data Quality for the Information Age*. Artech House, 1996.
- [9] Rumbaugh J., Jacobson I., Booch G.: *The Unified Modeling Language Reference Manual*. Addison Wesley, 1998.
- [10] Shankaranarayan G., Wang R. Y. and Ziad M.: “Modeling the Manufacture of an Information Product with IP-MAP”. In Proceedings of the *6th International Conference on Information Quality*, Boston, MA, 2000.
- [11] Stevens P., Pooley R.: *Using UML. Software Engineering with Objects and Components*. Addison Wesley, 1999.
- [12] Storey V.C., Wang R.Y.: “An Analysis of Quality Requirements in Database Design.”. In Proceedings of the *4th Conference on Information Quality*. Boston, MA, 1998.
- [13] Wang R.Y., Strong D.M.: “Beyond Accuracy: What Data Quality Means to Data Consumers”. *Journal of Management Information Systems*, 12(4), 1996.
- [14] Wang R.Y.: “A Product Perspective on Total Data Quality Management”. *Communication of the ACM*, 41(2), 1998.
- [15] Warmer J. B., Kleppe A.G.: *The Object Constraint Language-Precise Modeling with UML*. Addison Wesley, 1999.