# Quality Management in MSIS

**Adriana Marotta**
Université de Versailles-St-Quentin-en-Yvelines
Laboratoire PRISM

**Abstract**

This paper presents our first incursion in the problem of quality management in Multi-Source Information System (MSIS). We state the problem and experiment with the definition and classification of quality properties. We also experiment with a solution for the problem of quality evaluation in a MSIS considering a few selected properties.

## 1. Introduction

We consider a Multi-Source Information System (MSIS) as an Information System where exist a set of different User Views and a set of heterogeneous and autonomous Information Sources. Figure 1 shows the architecture of this system. There are three layers : source, mediation and application. The source layer contains each source with its associated wrapper, which translates queries and queries' responses that pass through it. The mediator layer has in charge the transformation and integration of the information obtained from the sources, according to the requirements coming from the application layer. The application layer provides the user views to the user applications through execution of queries over the mediation layer.

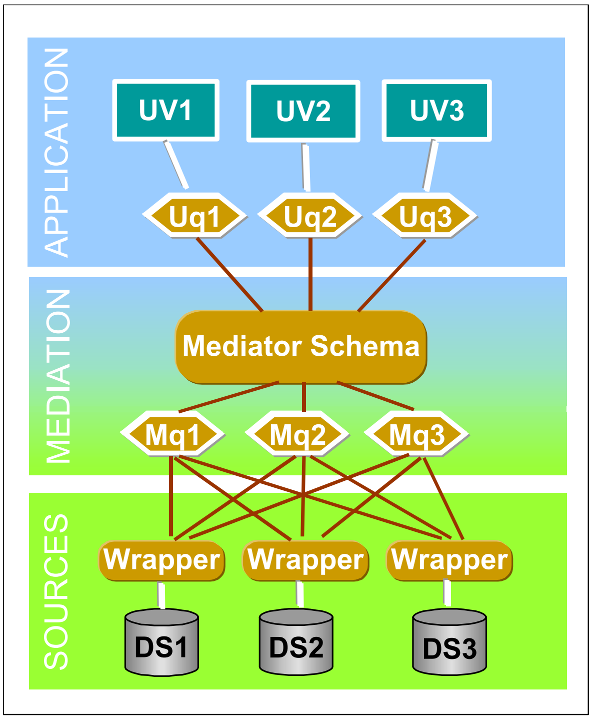


**Figure 1: MSIS Architecture**

In this kind of contexts the mediator layer is a compromise between user requirements and information existing in the sources. There are many works that address the different problems involved in these systems, such as sources' data integration, data cleaning, optimisation of views.

In this work we address the problem of quality in a MSIS.

There are some properties that are related to the general result obtained at the user side. They refer, for example, to the performance of the queries, the accessibility to the information, the completeness of the information, etc. We call these *quality properties*. Quality properties may have a great influence on the design of the different parts of an MSIS. In the present work we are interested in introducing the management of quality properties to MSISs.

In a MSIS architecture we find two aspects of quality: the quality that exists in the sources and the quality that is required by the user at the user views. This means that we have "actual" and "expected" quality values. In addition, for defining quality properties we must take into account that the vision of the user for establishing his quality requirements is usually different from the vision of DBAs for declaring existing quality values, leading to different quality criteria. In fact, one user-required property can be achieved by the combination of several source properties.

Quality properties and requirements may be used for evaluating the quality of the MSIS and also for making design decisions at the different layers of the architecture. In order to address these problems several subproblems must be solved, such as the propagation of quality properties of the sources to the mediation and user views layers, the propagation of the user requirements down to the mediation and source layers, and the conversion between the different quality criteria.

We believe that quality management in MSIS is a very wide problem. Our goal in this work is to present an overview of the general problem and possible solutions, and to focus on quality evaluation, not addressing the problem of quality impact on design.

In Section 2 we state the problem of quality management, in Section 3 we present a first experience defining some quality properties and solving the problem of quality evaluation in a reduced context, and in Section 4 we present the conclusions.

## 2. Problem Statement

In this section we analyze different kinds of quality factors as well as the layers of the MSIS where they may be considered. We also present an overview of the problem of quality management in a MSIS, considering two possible goals: quality evaluation and quality impact on system design.

### 2.1 Quality Properties and Requirements

In general, user quality requirements are not necessarily expressed in the same language as the quality properties that may be satisfied by a source. We believe that the user viewpoint for establishing his quality needs is completely different from the system viewpoint for declaring the properties of the data sources. For example, a user may require that a view has an *accessibility* of 8 (considering a pre-established scale of 1..10). This means that the sources that participate in this user view must satisfy certain values for the properties: *availability*, *locatability*, *connectivity* and *privileges*, which are the ones that contribute to the *accessibility* property.

Therefore, we distinguish two categories for the quality criteria: (1) user viewpoint and (2) system viewpoint. At the same time, we distinguish two categories for quality values (or expressions): (a) actual values (which we also call properties) and (b) expected values (which we also call requirements). We will show this through an example. Suppose a user view relation has a requirement of *accessibility*. This is an expected value and it is expressed with user viewpoint criterion. The value for *connectivity* that exists in one of the sources that contributes to that relation is an actual value and belongs to system viewpoint criterion. Suppose we obtain, from this value, the value of *accessibility* that offers the view relation. This is an actual value for a user viewpoint criterion property. Finally, suppose we obtain, from the *accessibility* requirement, a constraint over the *connectivity* of the source. This is an expected value (in fact, it generates a set of expected values) and it belongs to system viewpoint criterion.

Actual values are always generated from the quality measures at the sources, while expected values are generated from the quality user requirements. However, actual and expected values may correspond to category (1) or (2) and may be calculated over different layers of our architecture. In Figure 2 we show the possible combinations of the different categories of quality properties at the layers where they may be established. Note that some of the combinations have no sense, for example, an actual or expected value for a user view corresponding to the system viewpoint category.

**User Views:**

| | **ACTUAL** | **EXPECTED** |
|---|---|---|
| **USER viewpoint** | X | X |
| **SYSTEM viewpoint** | | |

**Mediator:**

| | **ACTUAL** | **EXPECTED** |
|---|---|---|
| **USER viewpoint** | X | X |
| **SYSTEM viewpoint** | X | X |

**Sources:**

| | **ACTUAL** | **EXPECTED** |
|---|---|---|
| **USER viewpoint** | | |
| **SYSTEM viewpoint** | X | X |

**Figure 2: Possible combinations of quality criteria**

## 2.2 Quality Management

Quality management may be included in MSISs with the goal **of quality evaluation/checking/validation** or with the goal of **impacting the design of the system.**

Before commenting the processes of quality evaluation and quality impact on the system design, it is necessary to explain the idea of *propagation* of quality properties and quality requirements, and the idea of *conversion* between quality criteria categories.

Quality properties are measured at the source objects, obtaining the actual source quality values. Then actual quality values can be derived at the corresponding mediator objects and user views objects. These derivations are done taking into account the queries that generate each object. For example, if a mediator relation R is derived from source relations R1 and R2, as R1 |><| R2, there must exist a function "Merge" that obtains the quality values of R from the quality values of R1 and R2. Mediator objects' quality values are derived from source objects' quality values, and user views objects' quality values are calculated from mediator objects' quality values. Therefore, we say we *propagate* the source quality properties when we derive the actual quality values for the objects of the mediator or user views. In [NLF99] they present a quality model to calculate a quality value for a plan, only considering join operators. The mechanism they propose could be adapted to our context and used for deriving the actual quality values at the mediator and user views layers.

Also quality requirements can be propagated from user view objects to the mediator and source objects. However, this propagation is not as direct as the previously explained. A user view object may be obtained from several mediator objects, therefore when we propagate the quality requirements of a user view object to the mediator, we obtain an inequations system that throws constraint expressions. These constraints relate the properties of the different mediator objects that participate in the query that generates the user view object. Analogously, we can propagate mediator objects quality requirements to source objects.

Figure 3 illustrates the idea of propagation.

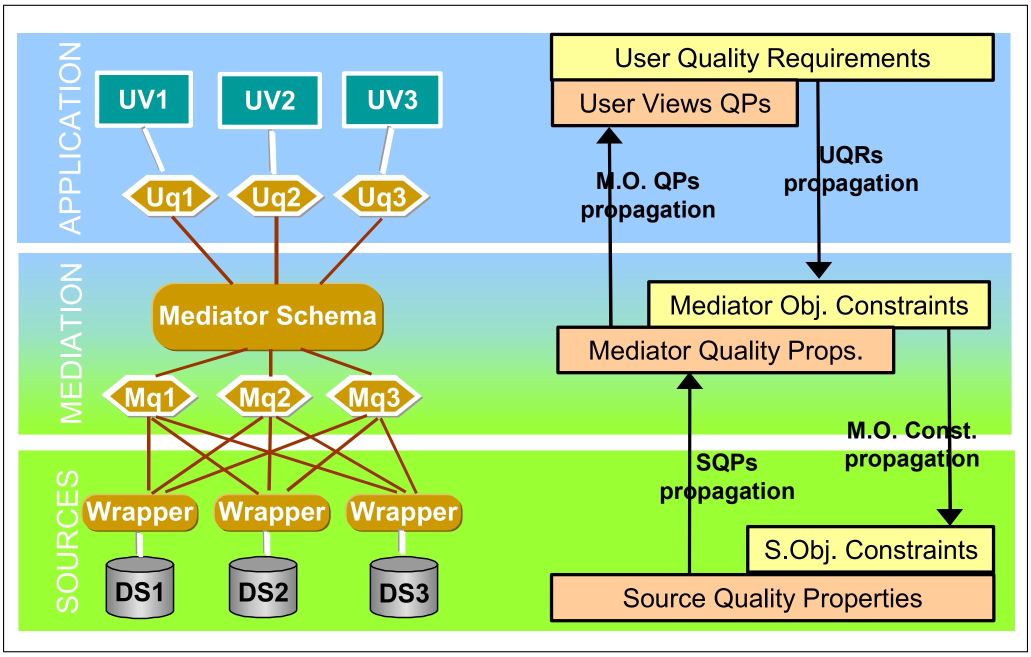


**Figure 3: Quality propagation**

In order to compare actual and expected quality values it is necessary to *convert* properties/requirements of the category *user viewpoint* into properties/requirements of the category *system viewpoint* or vice versa. This conversion is done at different layers of the system and with different direction according to the process that is being executed. The dependencies between the properties/requirements of the two categories must be specified. Such a dependency could be, for example, one that establishes that *accessibility* is equivalent

to a combination of *availability*, *locatability*, *connectivity* and *privileges*, some of them multiplied by a weight.
*accessibility = (availability * w1 + locatability * w2 + connectivity * w3) * privileges*

**Quality evaluation** may be addressed with two different approaches. (1) We may propagate source quality properties to the user views in order to confront them with user views quality requirements and give a diagnostic of the system quality. This is a bottom-up approach. (2) We may propagate user quality requirements down to the sources and evaluate which values or range of values for the source quality properties are necessary in order to satisfy user quality requirements. This a top-down approach.

Quality management may strongly **impact the design of the system**, since the design decisions that are made at the different layers can improve the quality obtained at the user views. In the following we show this through an example, in a simplified scenario. Consider a user view relation R(A,B,C), where the user poses as quality requirement the condition: *freshness(R) <= 5*. R is generated from the union of the source relations R1(A,B,C) and R2(A,B,C) (see Figure 4). Suppose that we have the following measures at the sources: *timeliness(R1) = 2* and *timeliness(R2) = 5*, and the conversion between quality criteria for this property is directly: *freshness = timeliness*. The actual value of *timeliness* for R may be calculated with the following formula:
*timeliness(R) = max (timeliness(R1), timeliness(R2) )*
In this case the existing design allows user requirements satisfaction.
However, in the case of Figure 5 the designer cannot apply the same operator (union) if he wants to satisfy the user requirements. Therefore, he must discard Source 2. On the other hand, he may consider the possibility of “negotiating” the required values with the user, or the actual values with the DBA of Source 2.

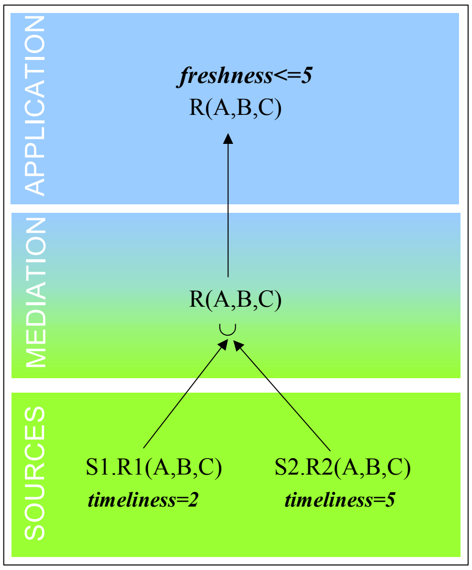


**Figure 4: Example. Impact on design.**

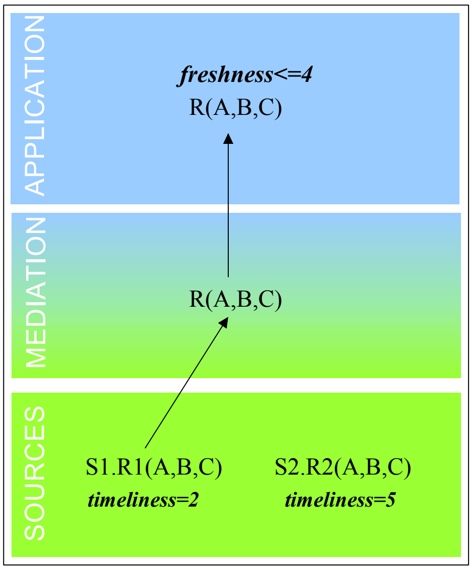


**Figure 5: Example. Impact on design.**

We believe that quality management may impact the following design problems in a MSIS:

- Source selection
- Mediator queries’ plan selection
- Mediator Schema design
- Mediator object selection for each User View
- User queries’ plan selection

- User View Schemas' design

In order to evaluate this impact, we need to propagate SQPs to Mediator and User Views and to propagate User Quality Requirements (UQR) from User Views down to Mediator and Sources. The latter, as said before, generates inequations systems that give constraint expressions.

For addressing the different design problems we take into account different properties, requirements and propagations. This can be seen in Figure 3. For the problem of source selection we consider the source object constraints and the source quality properties. For the problem of mediator queries' plan selection and mediator schema design we consider the mediator objects constraints and mediator quality properties. For the problem of user queries' plan selection and user views' schema design we consider the user quality requirements and the user views quality properties.

In the rest of the paper we concentrate in the problem of **quality evaluation**. We experiment with certain quality properties and propagation rules.

# 3. Experimentation

In this section we present a set of quality properties we have studied, we propose a mechanism for quality evaluation in a MSIS, and we show an example.

## 3.1 Quality Properties

We have selected a set of quality properties for the user viewpoint category and a set for the system viewpoint category. Following the proposal of [LSKW01], we classify the system viewpoint properties in 4 groups: Intrinsic (properties of the information itself), Contextual (properties that must been evaluated in the context of a task), Representational (properties related to the way information is represented) and Accessibility (properties related to the computer system that provides the information). We also define 2 groups for classifying user viewpoint properties: Content (refers to the quality of the information) and Operational (refers to the quality of the mechanisms that enable to access the data). Figure 6 shows the quality properties and their classification.

| | | |
|---|---|---|
| **System viewpoint** | **Intrinsic** | Accuracy |
| | | Believability |
| | | Reputation |
| | | Consistency |
| | | Granularity |
| | | Completeness |
| | | Unambiguity |
| | | Record Quantity |
| | **Contextual** | Relevance |
| | | Horizontal Fitness |
| | | Vertical Fitness |
| | | Timeliness |
| | **Representational** | Understandability |
| | | Concise representation |
| | | Sintax standardisation |
| | | Semantics |
| | **Accessibility** | Privileges |
| | | Availability |

| | | |
|---|---|---|
| | | Assistance |
| | | Locatabillity |
| | | Response Time |
| | | Connectivity |
| **User viewpoint** | **Content** | Correctness |
| | | Volume |
| | | Usefulness |
| | | Completeness |
| | | Freshness |
| | **Operational** | Performance |
| | | Accessibility |
| | | Easy of Use |

**Figure 6: Quality Properties**

Properties of the user viewpoint category are achieved by a combination of properties of the system viewpoint category. In Figure 7 we show for each property of the first category, the properties of the second that correspond with it.

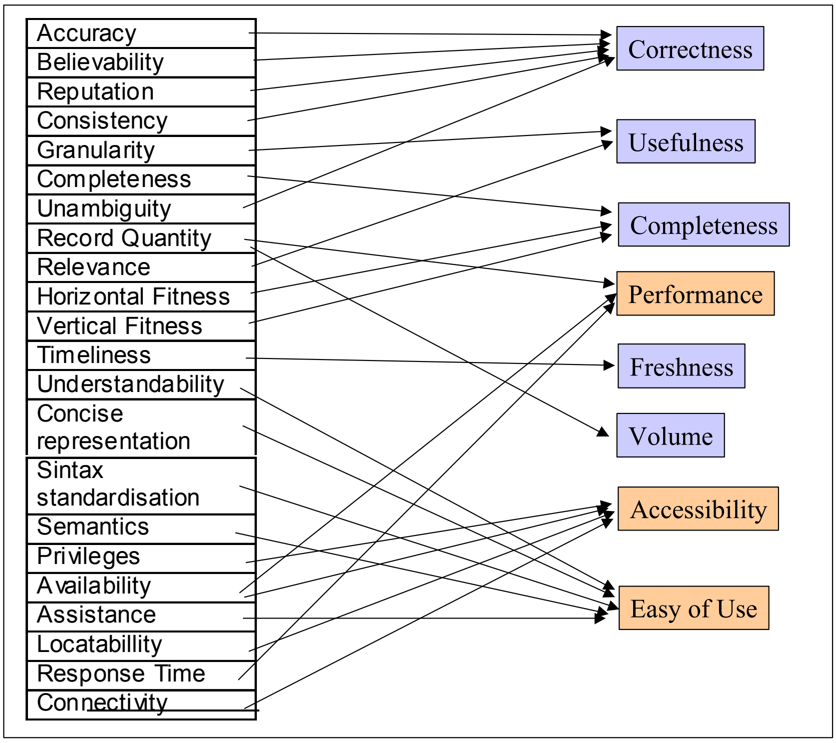


**Figure 7: Quality Properties Correspondence**

In the rest of this section we work with a subset of the properties: Accessibility, Freshness and Completeness, with their correspondent system viewpoint properties (see Figure 8).

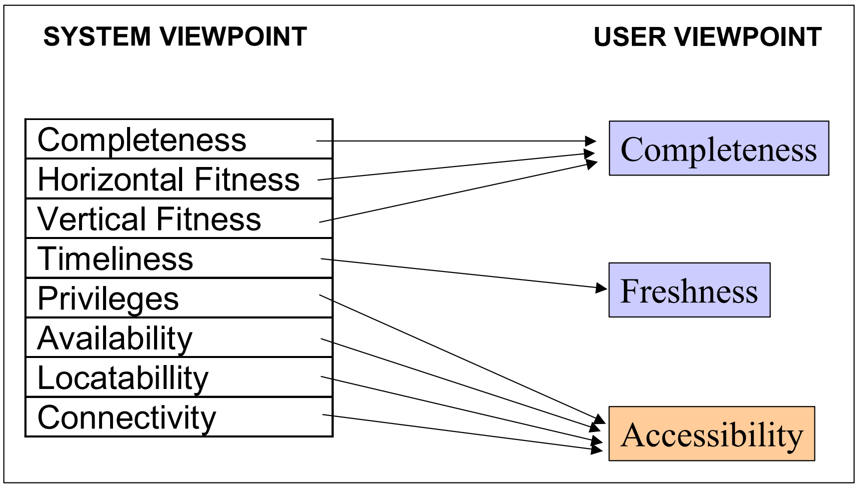


**Figure 8: Selected Quality Properties**

The user may be interested in having an idea of how complete and fresh is the information he obtains through the user views, and how easy is to retrieve the data coming from the sources. This is expressed by the *User Viewpoint Properties*: Completeness, Freshness and Accessibility.

In the following we give an idea of the meaning of each of the *System Viewpoint Properties*:

*Completeness* represents the percentage of data with respect to the real world.
*Horizontal fitness* is the percentage of not-null values for each attribute.
*Vertical fitness* is the percentage of matching between source attributes and required ones.

*Timeliness* is the update frequency at the sources measured in days.

*Privileges* expresses if certain user has the privilege for accessing certain data or not.
*Availability* is the percentage of time the source is accessible.
*Locatability* expresses how near is located the source from the user views (an index between 1 and 10).
*Connectivity* expresses the average amount of time that is necessary for connecting to a source (an index between 1 and 10).

## 3.2 Quality Evaluation

We focus on one of the approaches of quality evaluation. Our objective is to give a diagnostic of the system quality, we apply the bottom-up approach (Section 2.2).

The process of quality evaluation consists of the following steps:

1) Propagation of the Source Quality Properties (SQP) (actual values) to the user views.
We divide this step in three sub-steps. The first is the propagation of the SQP to the mediator objects. This is done through the mediator queries and Mediator Quality Properties (MQP) are obtained. The second is the propagation of the MQP to the user views objects. This is done through the user views' queries and User View Quality Properties (UVQP) are obtained.

Obtained MQP and UVQP are *actual values*, and belong to the category *system viewpoint*. The third sub-step is the conversion of the UVQP to the category *user viewpoint*.

2) Comparison between actual and expected values.
For each user view object and quality property we can obtain a proximity degree, which shows the proximity between the actual and expected value.

3) Calculation of a global quality value for each User View.
A global quality value for each view would give a kind of diagnostic of the quality achieved for the view with the present system conditions.

**Notation**

We define the following sets for specifying our solution.
*Relation Sets:*

URels = { R / R is a User View Relation }
MRels = { R / R is a Mediator Relation }
SRels = { R / R is a Source Relation }

*Relation Requirements Sets:*

URelReq = { <UR, Req, Val> / UR ∈ URels ∧ Req is a quality requirement ∧ Val ∈ Integer }

*Relation Properties Sets:*

URelPropUs = { <UR, Prop, Val> / UR ∈ URels ∧ Prop is a user viewpoint property ∧ Val ∈ Integer }
URelPropSys = { <UR, Prop, Val> / UR ∈ URels ∧ Prop is a system viewpoint property ∧ Val ∈ Integer }
MRelProp = { <MR, Prop, Val> / MR ∈ MRels ∧ Prop is a system viewpoint property ∧ Val ∈ Integer }
SRelProp = { <SR, Prop, Val> / SR ∈ SRels ∧ Prop is a system viewpoint property ∧ Val ∈ Integer }

*Procedure QueryPropagate:*

This procedure will be applied for propagating source quality properties to the mediator, and for propagating mediator quality properties to the user views.

For this procedure we adapt the proposal of [NLF99] to our context. They propagate properties through the query plans in order to deduce the quality of them. They consider a plan as a binary tree with QCAs (query correspondence assertions) as leaves and join-operators as inner nodes. They propose to use a function *Merge* for obtaining the property value of a relation that is the result of a join, from the property values of the participating relations.

In our case the binary tree corresponds to the mediator/user views queries and it has source/mediator relations as leaves. We also consider only join-operators. We must define a *Merge* function for each property we have.

**Propagation of the source quality properties**

The following is a pseudo-code of the propagation algorithm.

```
Propagation

For each R ∈ MRels do
        QueryPropagate (R)
For each R ∈ URels do
        QueryPropagate (R)
For each R ∈ URels do
        Derive_Accessibility (R, w_A, w_L, w_C, w_P)
        Derive_Completeness (R, w_C, w_H, w_V)
        Derive_Freshness (R, w)


Derive_Accessibility (R, w_A, w_L, w_C, w_P)
acc = 0
For each t ∈ UrelPropSys and t.UR = R do
        If t.Prop = "Availability" then
                acc = acc + t.Val * w_A
        Else if t.Prop = "Locatability" then
                acc = acc + t.Val * w_L
        Else if t.Prop = "Connectivity" then
                acc = acc + (10 - t.Val) * w_C
        Else if t.Prop = "Privileges" then
                acc = acc * t.Val
        EndIf
EndFor
Insert_in_UrelPropUs (<R, "Accessibility", acc>)

Derive_Completeness (R, w_C, w_H, w_V)
comp = 0
For each t ∈ UrelPropSys and t.UR = R do
        If t.Prop = "Completeness" then
                comp = comp + t.Val * w_C
        Else if t.Prop = "Horizontal Fitness" then
                comp = comp + t.Val * w_H
        Else if t.Prop = "Vertical Fitness" then
                comp = comp + t.Val * w_V
        EndIf
EndFor
Insert_in_UrelPropUs (<R, "Completeness", comp>)

Derive_Freshness (R, w)
fres = 0
For each t ∈ UrelPropSys and t.UR = R do
        If t.Prop = "Timeliness" then
                fres =  t.Val * w
        EndIf
```

EndFor
Insert_in_UrelPropUs (<R, "Freshness", fres>)

### 3.3 Example

In this section we show a simple example where there are two user views derived from two different sources. We evaluate the quality taking into account only one requirement over one of the user view relations.

We show how we propagate the source quality properties, how we convert them from system viewpoint category to user viewpoint category, and how we compare them with the quality requirement.

The following are the schemas of the sources, mediator and user views.

**Sources**

*Source 1*
Physicians (name, address, telephone, speciality)
Specialities-Diseases (speciality, disease)
Treatments (treatment-name, description)

*Source 2*
Physicians (name, age, speciality)
Treatments (treatment-name, description)
Diseases-Treatments (disease, treatment-name)

**Mediator**

Physicians (name, address, telephone, age, speciality)
Specialities-Diseases (speciality, disease)
Treatments (treatment-name, description)
Diseases-Treatments (disease, treatment-name)

**User Views**

*User View 1*
Physician-Diseases (name, address, telephone, speciality, disease)

*User View 2*
Speciality-Treatment (speciality, disease, treatment-name)

The following are the source quality properties and user requirements that are relevant for the quality evaluation of requirement *Accessibility* for user relation *Physician-Diseases*.

**Source Quality Properties**

| | Availability | Locatability | Connectivity | Privileges |
|---|---|---|---|---|
| Source1.Physicians | 8 | 8 | 2 | 1 |
| Source1.Specialities-Diseases | 8 | 8 | 2 | 1 |
| Source2.Physicians | 3 | 5 | 3 | 1 |

**User View Quality Requirements**

Physician-Diseases: Accessibility: 6

Figure 8 shows the scenario of the example.

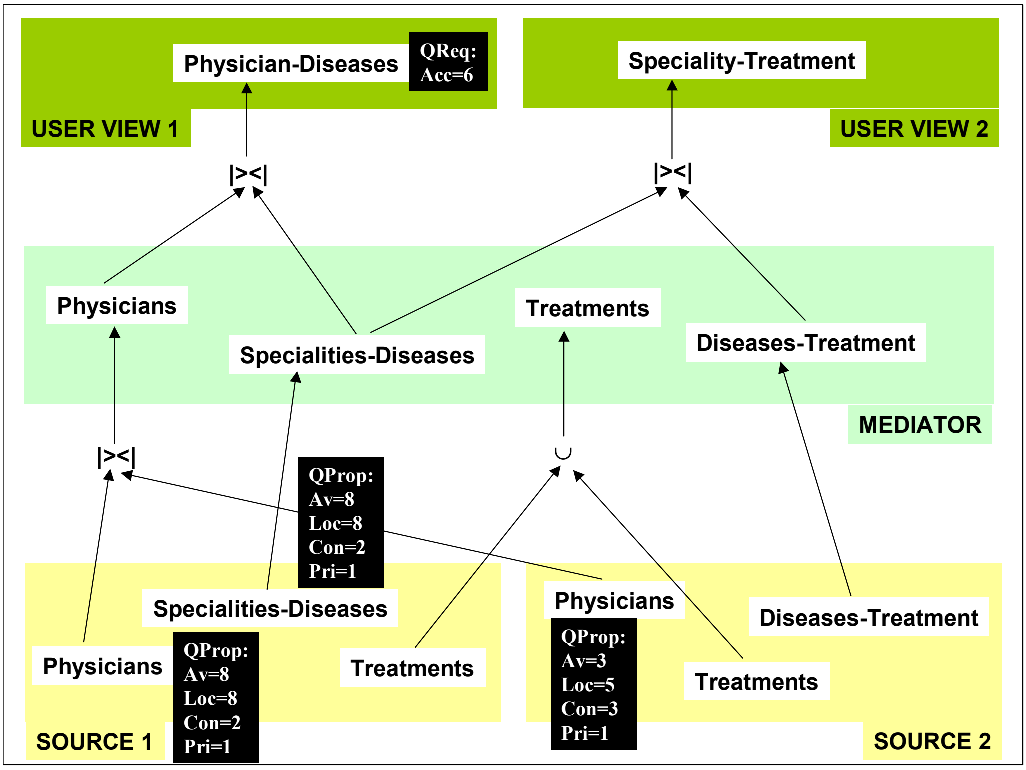


**Figure 8 : Example**

Now we present the process for quality evaluation of requirement *Accessibility* in user relation *Physician-Diseases*.

### Propagation

As we said in the previous section we apply the idea of [NLF99] for propagating the properties from the sources to the mediator and from the mediator to the user views, through the queries.

We need to define a *Merge* function for each quality property, which is applied in order to propagate the properties through a join operator.

*MergeAvailability* (Av1, Av2) = (Av1 * Av2) / 10
*MergeLocatability* (Loc1, Loc2) = (Loc1 + Loc2) / 2
*MergeConnectivity* (Con1, Con2) = Max(Con1, Con2)
*MergePrivileges* (Pri1, Pri2) = if (Pri1=0 or Pri2=0) then 0 else 1

*Mediator Quality Properties*

| | Availability | Locatability | Connectivity | Privileges |
|---|---|---|---|---|
| Physicians | 2,4 | 6,5 | 3 | 1 |
| Specialities-Diseases | 8 | 8 | 2 | 1 |

*User View Quality Properties*

| | Availability | Locatability | Connectivity | Privileges |
|---|---|---|---|---|
| Physician-Diseases | 1,7 | 7,25 | 3 | 1 |

**Conversion**

The DBA must decide which weights are assigned to each property for deriving the accessibility actual value of the user view relation.
For example:

Accessibility =

(Availability * 0.40 + Locatability * 0.20 + (10 – Connectivity) * 0.40) * Privileges

*User View Quality Properties*

Physician-Diseases:
Accessibility = (1,7 * 0,40 + 7,25 * 0,20 + 7 * 0,40) * 1 = 4,93

**Comparison**

*User View Quality Properties*

Physician-Diseases: Accessibility: 4,93

*User View Quality Requirements*

Physician-Diseases: Accessibility: 6

**Proximity degree (0..1): 0,80**

## 4. Conclusions

In this work we intend to state the problem of quality management in MSIS. We also approximate to a solution for quality evaluation. We experiment with some quality properties, proposing a classification for them, and a mechanism for deducing the quality offered by the system.

The present work shows a general overview of the problem of quality properties in a MSIS context. We are planning to continue by a deeper study of a few specific properties, including how they are measured, propagated, etc.

## References


**[NLF99]** Felix Naumann, Ulf Leser, Johann Christoph Freytag. *Quality-driven Integration of Heterogenous Information Systems.* VLDB 1999: 447-458

**[LSKW01]** Y. W. Lee, D. M. Strong, B. K. Kahn, R. Y. Wang. *AIMQ: A Methodology for Information Quality Assessment.* Forthcoming in Information & Management, published by Elsevier Science (North Holland). (Accepted in November 2001)