Journal of Web Engineering, Vol. 1, No.1 (2002) 061-073

TOWARDS A REUSABLE REPOSITORY FOR WEB METRICS

LUIS OLSINA[1], GUILLERMO LAFUENTE[1], OSCAR PASTOR[2]

[1] *La Pampa National University*
olsinal@ing.unlpam.edu.ar

[2] *Valencia Polytechnic University*
opastor@ dsic.upv.es

Received: September 2, 2002

In this article we introduce a metric model as one of the building blocks for a repository of metrics. Particularly, starting from a conceptual model for metrics, we thoroughly discuss a catalogue template for product metrics instantiating it with some Web metrics. A catalogue of metrics basically allows tools, evaluators and other stakeholders to have a service and a consultation mechanism, which starts from a sound specification of the entity type, the attribute definition and motivation, the metric formula, criteria and protocols, among other template items. The metrics repository and the cataloguing tool can be appropriately used to support different quality assurance processes such as non-functional requirements specification, quality testing definition, etc. in different phases of the software life cycle. Effective and full-fledged quality assurance processes require not only strategic but also technological support as well.

Key words: Repository of Metrics, Quality framework, Web metrics, Cataloguing Tool
Communicated by: Y Deshpande

1 Introduction

Due to the unceasing growth of Web sites and applications (WebApps), developers and evaluators have interesting challenges not only from the development but also from the quality assurance point of view. As we know, the quality assurance is one of the challenging processes to Software Engineering as well as for the Web Engineering, as a new discipline[1]. Particularly, regarding the quality perspective, a clear definition and management of functional and non-functional requirements in order to specify, measure, control, and potentially improve the produced WebApps are largely needed. In addition, the support of automated processes and tools is also necessary in order to assist evaluators and other stakeholders in quality assurance activities. This way, providing a repository of metrics and a cataloguing tool will basically facilitate a consultation, retrieval and reuse mechanism, starting from a sound specification of the entity type, the attribute definition, objective and motivation, the metric formula, criteria, protocols (counting rules), metric unit, and scale type, among other elements.

As Web sites have grown both in interaction and functionality, they have changed just from being static, document-oriented pages to dynamic application-oriented pages with at least the complexity of traditional software applications. This makes the evaluation task and ultimately the quality assurance more challenging. Although there exists many design guidelines, heuristics[2] and metrics for the evaluation of WebApps[3,4,5] most of them lack a well-defined specification framework and, even worse, a strategy for consultation, retrieval and reuse.

Some initial efforts have been recently made to classify metrics for some entity type as for example metrics for software products. It is worthwhile to remark the initiative of the ISO 9126 standard[6] in the 9126-2 and 3 draft documents. However, most of the product metrics informed in drafts are either sufficiently generic or not customized for Web products. Furthermore, the paper-based template of information used to describe those metrics is not sufficiently complete (as we analyse later on), and is not intended to being automated by retrieval tools. On the other side, it is also worthy of mention the effort carried out by Kitchenham *et al.*[7] in the definition of a conceptual framework and an infrastructure (based on the Entity-Relationship model) to specify entities, attributes and relationships for measuring and instantiate software projects, with the purpose of analysing metrics and data sets in a consistent way.

This last framework serves as a starting point for our proposal, which we are trying to strengthen from the conceptual modeling point of view (using O-O approaches), as well as from the cataloguing technologies and offered services. Firstly, we propose a conceptual quality framework for entity types grounded in the ISO 9126-1 quality model, as well as a conceptual model for metrics. When a metric is defined and specified, it is necessary to previously know what attribute of what entity type it will quantify. Particularly, in these last years we were defining and specifying Web site metrics (considering a Web site or component of it as a product), mainly those that can be automated from the data collection point of view. Among the hundred and fifty automated web page and site metrics catalogued so far different categories were identified as *Link and Page Faults*, *Navigation*, *Information*, *Media*, *Size*, *Performance*, *Accessibility*, among others. We will describe and exemplify the catalogue template with some few Web metrics[4] focusing on the idea that the given results can be extended to any other metric belonging to the product entity type. Secondly, regarding the repository of metrics, we are designing and building a cataloguing tool which will basically provide a Web-based collaborative mechanism for discussing, agreeing, and adding candidate metrics to the repository on one hand, and a Web-based robust query functionality based on the Semantic Web approach and Web services for consultation and retrieval, on the other.

Unfortunately, in recent Web research initiatives, robust Web metric specifications and cataloguing environments as technological support for quality assurance processes have often been neglected.

The rest of this article proceeds as follows. In Section 2, we propose a conceptual framework of quality for entity types, quality models and metrics. In Section 3, starting from this conceptual quality framework and a conceptual model for the metrics domain, we thoroughly discuss a catalogue template for product metrics, and some design aspects of the cataloguing tool. In addition, in subsection 3.2, we comment some catalogued Web attributes and metrics, focusing the attention on those automated; in subsection 3.3, an architectural view of the cataloguing environment is depicted. Finally, concluding remarks and future works are drawn.

2 A Conceptual Framework of Quality

The integral evaluation of attributes for different entities is a difficult endeavor not only in the Software Engineering field but also in the Web Engineering. In its broadest sense, it is difficult to consider all the characteristics and mandatory or desirable attributes of a process, resource, product (like a WebApp), or product in use, if there are no sound quality framework, models and methods that allow evaluators specify systematically those characteristics and attributes.

As a first step, the definition of a conceptual framework of quality that serves as a guideline in the classification process of entities, models and associated metrics should be considered necessary. As shown in Figure 1, the core relationships among the quality factors (from which attributes and metrics can be derived) is tried to be captured at a high level of abstraction, in consideration of the entities that can intervene in a quality assurance project. Some aspects of the conceptual framework of quality are next described.

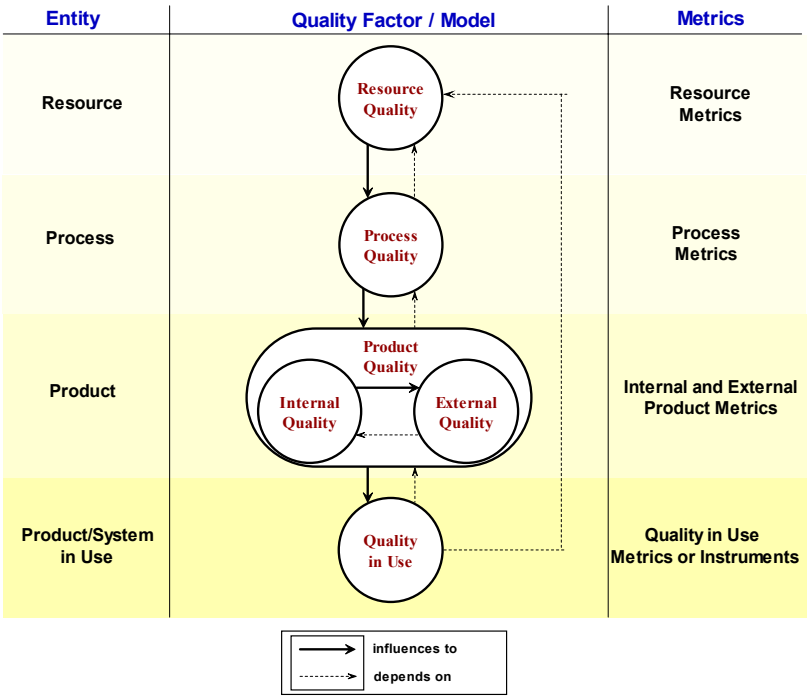

Figure 1. Conceptual framework of quality regarding different entity types and potential quality models of interest to Software and Web Engineering fields.

In that schema, enhanced from the ISO/IEC 9126-1 quality model framework[6], we implicitly observe that each quality factor (e.g. product quality) belongs to an entity of the empirical domain. Because an entity can only be measured through its attributes, it is necessary to define metrics of entity attributes in order to be able to analyse and surmise from numbers. In addition, one or more quality models can model each quality factor.

For this conceptual framework, the following factors have been kept in mind: *Quality of Resource*, *Quality of Process*, *Quality of Product* and *Quality in Use*. In figure 1, we can observe that the resource quality potentially contributes to improving process quality, and that the process quality influences the product quality, and this in turn, influences the quality in use. It is important to highlight that the evaluation of the quality in use can provide feedback for improving a product, and the evaluation of a product can give feedback for improving the process quality.

In addition, all the above factors can contribute to the *Quality of a Project* as a whole.

Next, a short description of each one of these factors involved in the conceptual framework is given. By means of the *Quality of Resource* factor, a quality model to measure human, or technological resources, etc. can be specified, which can influence the quality of processes. By means

of the *Quality of Process* factor you can specify a quality model (e.g., ISO/IEC 15504, or other) to measure different aspects of a process. In the same way, you can use a model for the *Quality of Product* factor. Our proposal of product quality based on the documented experience in the literature[6,8] models it considering the internal and external quality of a product. The internal quality is measured through internal metrics of a product; that is to say, they measure aspects of the internal view of a product without considering its behavior and environment. The external quality is measured through external metrics where the product is generally in operation state; here the important thing is the set of characteristics and attributes that influence the external view of a product, generally being in a simulated execution environment. Lastly, by means of the *Quality in Use* factor, the users' perceptions and reactions interacting with the real product in specific scenarios of use can be measured, considering specific user profiles.

For instance, for the *Quality of Product* factor we can choose the ISO 9126-1 quality model. This standard prescribes six well-known characteristics as well as a set of subcharacteristics for each one. The hierarchical model can be specified as a tree composed of characteristics, subcharacteristics, and attributes.

Let us consider the *Orphan Page Count* attribute, which measures (counts) pages having no return link to the site where are included in. *Orphan Page Count* attribute can obviously be measured as a direct metric in the absolute scale. A possible indirect metric (regarding internal or external quality depending on the case) is: X = *OrphanPageCount* / *PageCount*. Now, how do we relate this metric to a characteristic or sub-characteristic? For instance, intuition, experience, and ultimately empirical studies can draw this attribute as being associated to the *Reliability* characteristic.

We made different case studies customizing this kind of hierarchical model to WebApps[9,10,11].

In the next section, based on an UML-based conceptual model for metrics, the different template items that should be considered for cataloguing purposes are analysed.

3 Towards a Repository of Metrics for the Cataloguing Environment

For the above conceptual framework, it would be necessary to have a formal conceptual model that allows documenting and accessing metrics involved in the quality factor for each entity. In the end, a cataloguing environment can offer users consultation mechanisms from an on-line repository, amongst other functionalities. Users may consult and filter information with the purpose of obtaining the desired metrics in a quick and efficient way. For instance, they could query for the product entity and, specifically, for the Web site sub-entity, what automated metrics exist for broken links (let it be internal or external to the site), and for a potential beneficiary as a maintainer.

Figure 2, shows an UML-based conceptual model for the domain of metrics specifying the main classes and relationships useful to define the metadata for cataloguing purposes. In addition, this model can be enlarged in order to store instances of metrics with theirs values for different entities in specific software projects[7]. A thorough discussion of this conceptual model will be given in a future work, however, description of it are given in the next section.

3.1 Analyzing the template items

Starting from the components of the above conceptual model, some template items for the quality factor are illustrated in order to build a catalogue of metrics (that we are formally specifying it in XML

and RDF languages[12]). Table 1 shows the template items instantiated with an automated Web metric: The *Orphan Page Count* metric.

The *Name* item (from the *Attribute* class) serves to identify in natural language the attribute title in the catalogue. The *Keywords* item serves to indicate all those key words that are related with the attribute, including possible *Alias*, as illustrated in Table 1. It might be useful searching in the catalogue for alternative names and key words.

In the *Definition* element the attribute is described in a succinct and unambiguous way. The *Objective/Motivation* item is intended to help users identifying clearly which the purpose of the attribute is, and which its utility is, i.e., for what it serves.

The *Level of Independence of the Application Domain* item indicates the independence degree of the attribute and it can serve as a guide to see whether it can be reusable in different application domains or not. A possible ordinal scale can be: TI = Totally Independent; PD = Partially Dependent; and TD = Totally Dependent. For instance, *Broken Links* and *Orphan Pages* attributes can be considered as totally independent for different WebApps domains.

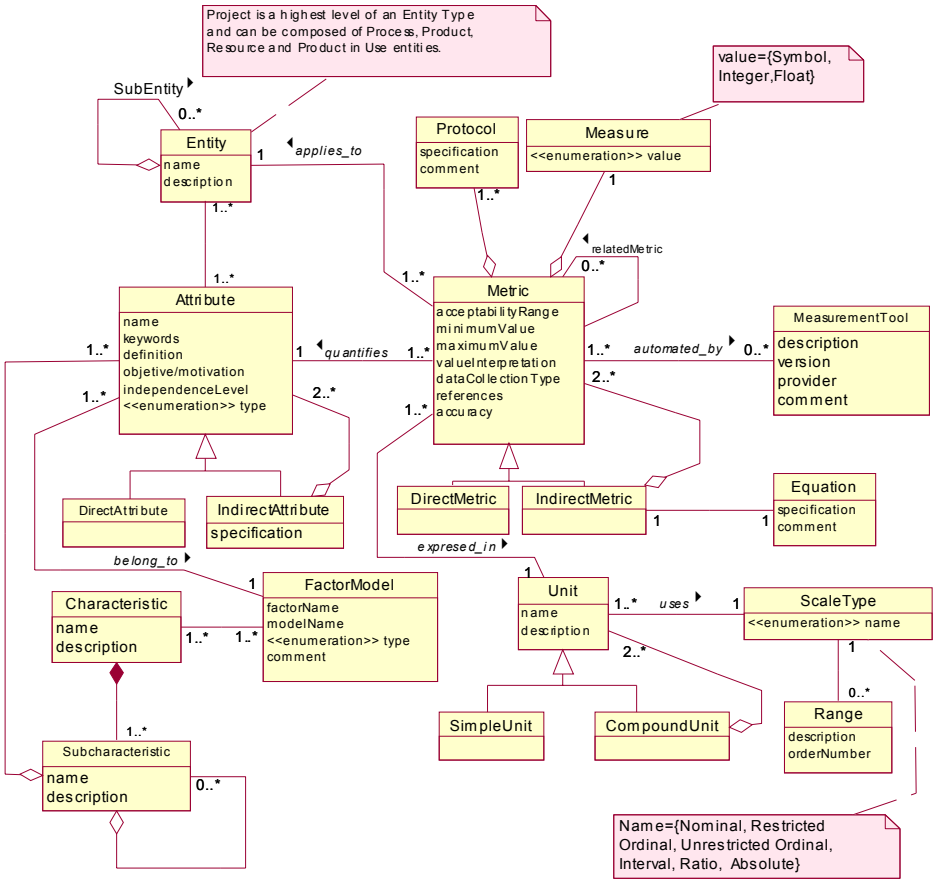

Figure 2. Conceptual model for the domain of metrics.

The *Attribute Type* item helps to know if the metric corresponds to an internal or external attribute as commented in Section 2.

The *Entity* element indicates the entity type (i.e., project, product, process, etc.). For the above example, the entity corresponds to the product whose *Sub-entity* is a Web site or page (or component of it). The entity types at a high level of abstraction are those specified in the conceptual framework presented in Section 2, meanwhile an entity can be hierarchically broken down in sub-entities as modeled in Fig. 2.

The *Quality Model* item indicates the stated quality model to which the attribute is specified, if any. The objective of the *Characteristic* sub-item is to indicate the higher-level characteristic to which the attribute is related. In our example, the *Orphan Page Count* attribute corresponds to the *Reliability* characteristic for the ISO 9126-1 quality model. In turn, for a given attribute, the related *Subcharacteristic* belonging to the characteristic can be included as well (in Fig. 2, we have specified the *Factor Model* class, because quality is one of the factors; other, can be the cost factor).

In order to help users searching and browsing the catalogue, the *Potential Categories* item serves as a way to classify metrics (such as metrics for Length and Size, Complexity, Faults, etc.). For instance, the Failure and Fault category can be split into the Link, or Page *Subcategories*.

The *Equation* item specifies how the X's variable value can be obtained (X is the independent variable of the metric). For a metric of a direct attribute, X is computed directly as is the case for the *Orphan Page Count*. For a metric of an indirect attribute, X is calculated from a mathematical model (i.e., an equation that model dependencies of attributes and / or parameters). The *Associated Metrics* item is provided for indirect metrics, and it is useful to link to the metric templates of those variables that are involved in the metric calculation. For instance, the formula of the *Percentage of Orphan Pages* attribute can be expressed as: X = (*OrphanPageCount* / *PageCount*) * 100; where *Orphan Page Count* and *Page Count* are associated and catalogued metrics. On the other hand, *Related Metrics* item is considered, mainly for browsing purposes (see Fig. 2).

The *Protocol* element explains and specifies the procedural mechanism and counting rules to be followed and applied to the respective metric. In case of automated metrics (as it is that of our example) the application algorithm might be specified. This serves as a guide for data collection and computation processes. In addition, in this item or in the *Comment* one, information about the protocol with the purpose of guaranteeing repeatability and reproducibility in the evaluation process[13] can be added. Also other *Alternative Procedures* can be specified accordingly.

The purpose of the *Interpretation of Measured Value* item (the *valueInterpretation* field of the *Metric* class in Fig. 2) is that of helping stakeholders to understand the obtained value, e.g. the closer to zero is the better. Besides, the catalogue should contain information about the *Unit* in which the metric is expressed (e.g. LOC, number of pages, etc.), and the used *Scale Type* such as nominal, ordinal, interval, ratio, or absolute. The scale type defines what admissible transformations are possible, in addition to the type of mathematical operations and statistical analyses that can be carried out with numbers[8,14]. The overall *Accuracy* of the metric can also be determined considering, for instance, the average hit and miss accuracies.

For the *Data Collection Type* item, how to collect the data and perform the corresponding calculation (automatically, semiautomatically, or manually) can be indicated. As a subsidiary element the identification of *Measurement Tools* can be added as well as its URLs or complementary references (the tool version and supplier).

Table 1. Template items for cataloguing product metrics exemplified by the *Orphan Page Count* metric.

Attribute Name (Title)	**Orphan Page Count**			
Keywords / Alias	Dead-end Page Count, Orphan Page, Isolated Page, Web Metric			
Attribute Definition	An orphan page has no internal link to the site where is included in (or it has all internal links broken). Although can have some external links, these will not allow to navigate inside internal pages of the site.			
Objective / Motivation	Count the number of pages that have no internal links to the Web site where they are included in. When a visitor accesses an orphan page through an external URL, he/she is unable to navigate inside the site. This kind of page has no internal navigational functionality and its utility depends rather on its content exclusively (see the Observation item).			
Level of Independence	TI (Totally Independent of the domain)			
Entity Type	Product			
	Subentity	Web site or page		
Quality Model	ISO/IEC 9126-1			
	Characteristic	*Reliability*	**Subcharacteristic**	*Maturity*
Potential Categories	Failures and Faults	**Subcategories**	Page	
Equation	X= #OP (Number of Orphan Pages)			
	Associated Metrics	None, because it is a direct metric		
Related Metrics	None			
Attribute Type	External or internal, depending on the process lifecycle.			
Protocol	From a starting URL (of a given site page), recursively analyse all the pages, considering exclusively those that have at least an internal and not broken link, following this generic algorithm: *Preconditions* Starting from the initial URL to analyze = URL_j ; Orphan_page = 0; j: 1.. Page Count *Orphan_pages (URLj): #Orphan_pages* *For each* page (URL_j) not previously analysed *If* ¬∃ an internal(URL_{ji}) not broken *then* #Orphan_pages = 1 + Orphan_pages (URL_{j+1}) *else* #Orphan_pages = Orphan_pages (URL_{j+1}) *end if* *end for each* *end*			
	Comment	In order to perform the calculation the email address link (or other resources) isn't computed.		
	Alternative Procedure	None		
Interpretation of Measured Value	X > = 0, the closer to zero the better			
Unit	Number of orphan pages			
Scale Type	Absolute (it is a counting)			
Accuracy	To be determined from a site sample			
Data Collection Type	Automated	**Measurement Tool**	LIFT Onsite, WebMA, among others	
Potential Processes of Use	Development, Testing, Integration, Maintenance			
Potential Beneficiaries	Developer, Tester, Maintainer			
References	J. Nielsen, wwww.useit.com/alertbox/9605.html.			
Observations/Comments	This metric can give stakeholders useful information both in the development and maintenance phase indicating the absence of page links that allow smooth site navigation.			

The *Potential Processes of Use* and *Potential Beneficiaries* items indicate in what processes the metric could be used and to what participant roles can be targeted. The *References* element can contain bibliographical or URL resources, where additional and authoritative information of the given metric can be consulted.

Finally, there are some fields not shown in the catalogue template that should be part of it, as for example, the metric creation and update dates in the repository.

3.2 Some catalogued web metrics.

So far, we have catalogued about a hundred and fifty Web metrics where data gathering can be automated. As previously remarked, the attributes of an entity may be categorized as direct or indirect attributes. In addition, according to the data collection type, attributes can be partially or totally automated. Precisely, the X value of an attribute could be determined by a manual or automated process, i.e., by means of a measurement tool. Although in case studies we performed[9,10] many attributes values were gathered just observationally (because there was not another way to do it, as for example for *Table of Contents*, *Site Map*, etc.), the automatic data collection and metric calculation were in many cases the only way to obtain reliable and efficient values. This was the case for attributes such as *Broken Link Count*, *Quick Access Pages*, *Orphan Page Count*, among others. We are going to discuss some few automated metrics by tools implicitly highlighting some catalogue template items, for space reasons.

3.2.1 Broken link count. This indirect attribute represents the total number of broken links both internal and external to the site, not including dynamically generated pages and links as shown in the procedure in Figure 3. It is important to know if a broken link is internal or external to the site because a broken internal link is likely caused by carelessness or by an extreme complexity in the structure, meanwhile the other, is caused by an external and uncontrollable environment.

On the other hand, this attribute does not take into account the distinction among broken links to identical URLs; so all broken links are counted.

Figure 3, shows the application procedure that automates the *Broken Link Count* metric. The WebMA_Tool[4] implements this algorithm and stores the current and destination URLs that would lead either to the internal or external broken link. This allows ulterior analyses and corrections. According to the returned HTTP state code, a broken link will be detected by this code; likewise, depending on the returned state code other link failures and metrics can be determined.

This attribute influences the quality of Web sites. From the visitor point of view, the bigger the number of broken links is, the lesser the reliability on the site is. From the maintainer point of view, the distinction between external and internal links (broken or not) is relevant, as commented above.

3.2.2 Link count. The total (absolute) number of links of static pages of a site can be collected automatically. By reusing the algorithm shown in Fig. 3, the total number of internal and external links can be calculated (this can be enhanced to take into account both textual and graphical links). Sometimes, when just internal links are considered, the metric in the literature is called *Connectivity*[3].

```
Preconditions
   Starting at the initial URL of the Website to analyse URL= URLi
   #Broken_links = 0; #Internal_broken_links = 0; #External_broken_links = 0;
   j: 1..PageCount.

Broken_links (URLj):#Broken_links
For each link (URLji)of page with URL= URLj not previously analysed
   If (URLji) is broken then
      If (URLji)is internal then
      #Internal_broken_links = #Internal_broken_links + 1
      Else
      #External_broken_links = #External_broken_links + 1
      End if
   Else
      If (URLji) is internal then
         #Broken_links = #Internal_broken_links + #External_broken_links +
         Broken_links ()
      End if
End if
End for each
    Return (#Broken_links)
End
```

Figure 3. Specification of the algorithm that automates the *Broken Link Count* metric for static pages. This algorithm changes considerably for dynamic pages.

3.2.3 Percentage of broken links. Using the above metrics (broken link count, and link count), the percentage is calculated by the following formula:

$$PercentageBrokenLinks = 100 * \frac{\#InternalBrokenLinks + \#ExternalBrokenLinks}{LinkCount}$$

This attribute may be considered domain independent. Besides, the metric of this attribute shows to some extent how reliable a site is (the reader can also consider the quotient between the number of external broken links and the total number of external links, likewise can be done for internal links). A more careful study on the quality impact of these metrics would envisage the importance of broken links relating to their location in the more relevant or visited pages of a site (this gives place to the definition of the *Frequency of Broken Links per Hit Pages* indirect attribute).

3.2.4 Number of different broken links. This metric is obtained analysing the distinct URLs used in the Broken Link Count metric. Internal and external broken links to the same resource are just counted once. It can show useful information for the maintenance phase helping in the analysis of the impact on changes. In this case, the procedure to data collection and computation as specified in Fig.3, has a slight change, i.e., just checking if the considered URL was visited before or not.

3.2.5 Percentage of different broken links. Instead to the *Percentage of Broken Links* metric, the relation is established according to the non-repeated links. The percentage is calculated as follows:

$$PercentageDifferentBrokenLinks = 100 * \frac{\#DifferentBrokenLinks}{DifferentLinkCount}$$

As the reader can consider, combining distinct metrics useful information can be drawn. For instance, we can see what is the level of link redundancy regarding the quotient between the *Different*

Link Count and the *Link Count* (either internal or external or both). The equation is similar as shown to the 3.2.8 attribute.

3.2.6 Image count. It helps to measure the amount of provided visual information (in the same way, we can alternatively measure the *Media Count* metric, where the number of media files is considered). The existence of images in a page is checked through the IMG property that is supported by the HTML code.

3.2.7 Different image count. This direct measure counts the non-repeated images in the site.

3.2.8 Percentage of image redundancy. The relation between the amount of different images and the image count in a site can be posed as shown in the next formula. An image repetition may be interpreted as the level of redundancy of visual information.

$$PercentageImageRedundancy = 100 * \left(1 - \frac{DifferentImageCount}{ImageCount}\right)$$

3.2.9 Quick access pages. The download time (T) is related to the size of a page (τ) and the speed in the established connection line (c).

$$\mathrm{T}_{Download} = f(\tau, c)$$

This time is directly proportional to the page size and inversely proportional to the speed of a given connection line. A function may be created in order to classify pages as quick or slow access pages, according to a minimum threshold of time (e.g. 10 seconds) for a given speed of a connection line. (Readers can find information about recommended page sizes depending on the speed of communication lines for example in[2]).

$$g(\mathrm{T}_{Download}) = \begin{cases} QuickAccess & \mathrm{T}_{Download} < \mathrm{T}_{\max} \\ SlowAccess & \mathrm{T}_{Download} \geq \mathrm{T}_{\max} \end{cases}$$

This criteria is a simple way to measure the performance (or predict it at design time), however, it does not reflect the actual or perceived performance, as the reader might surmise.

Finally, in Table 2, some few other attributes to entity types described in Section 2 are illustrated, which can be part of a repository of metrics.

3.3 Some design issues for the cataloguing environment.

Regarding the repository of metrics, we are designing and building a cataloguing environment that basically will provide a Web-based collaborative mechanism for discussing, agreeing, and adding approved metrics to the repository on one hand, and a Web-based robust query functionality (based on Semantic Web principles) for consultation and reuse, on the other hand. These subsystems are outlined in Figure 4.

From the point of view of the design of users for the cataloguing environment, four user's role types with different responsibilities and access privileges were considered, namely: *Administrator*, *Moderator*, *Reviewer* and *Final User*.

Table 2. Summary of attributes for different entity types that can be part of a repository of metrics. Although not shown in the table, the Project entity can have attributes such as *Project Duration*, or *Used Development Process*, among others.

Entity	Sub-entity	Attribute	Definition
Resource	Personnel	**Productivity**	Defined as the quotient between the size of the produced output and the required input as effort[8]. For instance, the LOC produced per person days.
	Method/Tool	**Method/Tool Usage Level**	Defined as the level of use of a given method (or tool) in a Web or software project.
Process	Authoring	**Interlinking Effort**	Defined as the estimated elapsed time taken to interlink Web site pages.
	Testing	**Link-testing Effort**	Defined as the estimated elapsed time taken to test all links in a WebApp[3].
		Coding Faults Count	Defined as the number of faults found in code testing.
Product	Program	**Code Length**	Defined as the number of lines of code in a program (here, a distinction whether commented lines of code or not can be made).
	WebApp	**Program Types Count**	Defined as the number of different programming technologies used to build programs in a WebApp. For instance, JavaScript, CGI scripts, Java applets, ActiveX, etc.
	Page	**Page Media Count**	Defined as the number of different types of media used in a page.
Product / System in Use	WebApps in use	**Task Completion Time**	Defined as the elapsed time a user takes in completing a previously established task. We can obtain the average elapsed time for a user's type and compare it with the one an expert user had taken.
		User Success Rate (or Task Completeness Level)	Defining this rate as the percentage of tasks that users complete correctly. It measures users' ability to complete tasks, http://www.useit.com/ alertbox/20010218.html
		Task Completeness Efficiency	Defined as the quotient between the completeness level and the average completion time.

The *Administrator* is the final responsible that has total access to the repository of metrics being able to add metrics to it once approved and, if were necessary, to eliminate instances of it. On the other hand, it is in charge of managing and coordinating to moderators (which are responsible for the discussion forums of candidate metrics), in addition to updating the repository with approved metrics between the reviewers and the moderator, or ultimately to veto them and put them again into consideration.

The *Moderator* is responsible for selecting and coordinating the reviewers group and for putting into discussion the candidate metric and the work calendar. As much the moderator as the reviewers will work in a private shared area that is not that of the repository, both with the respective visibility and permission accesses. Web-centred collaboration mechanisms shall be used both synchronous and asynchronous as well.

The *Reviewer* is responsible for contributing and discussing in the metric definition for the different template items, as seen in Section 3.1. Each reviewer will have reading access to the other reviewers' area and, in definitive, it will be the moderator who passes to approved state the agreed metric notifying in turn to the administrator of this event.

With regard to the *Final User*, it can be a human being or a software application using the repository services. People will be able to access the catalogue of metrics by means of searching and browsing functionalities with read-only access permissions. The applications will be able to access the repository in the same way, for example by means of a SOAP (*Simple Object Access Protocol*) interface and Web services.

Finally, it is important to remark that some template items will be specified in MathML (*Mathematical Mark-up Language*), favouring this the rendering and capturing of formulas and specifications.

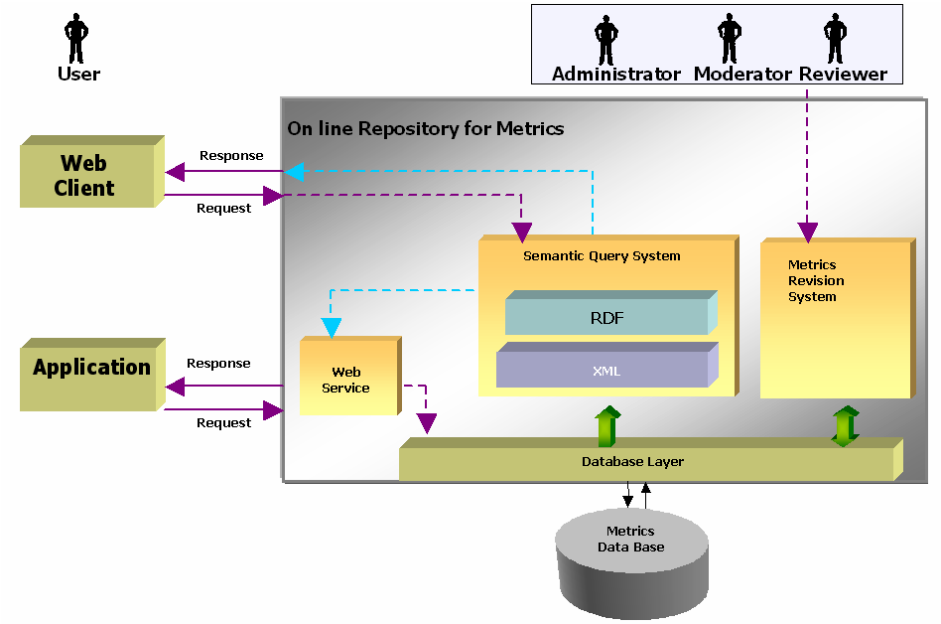

Figure 4. An Architectural view of the Cataloguing Environment.

4 Conclusions and Future Work

As Pfleeger says, "*metrics are welcome when they are clearly needed and easy to collect and understand*" [15]. In order to contribute to the comprehension and selection process whether metrics can be useful, easy to collect and understand, a sound and flexible metric documentation and consultation mechanism is needed. For this end, we thought that a reusable repository of metrics and a cataloguing tool could be efficiently used to support different quality assurance processes such as non-functional requirement definition and specification, metrics understanding and selection, quality testing definition, either in the inception, development or maintenance phases. It is recognized that effective and full-fledged quality assurance processes require not only conceptual frameworks but also technological support as well.

Unfortunately, in the recent initiatives of the Web research community, Web metric specifications and cataloguing environments as technological support for quality assurance processes have often been neglected. As a way to contribute to fill this gap, our current research concern is twofold. Firstly, we are exploring, specifying, and documenting Web metrics mainly those where data collection and calculation can be automated. In addition to the metric documentation work, a follow-up version of the WebMA_Tool[4] that automates many of those metrics is being developed. Website MA and WebQEM_Tool[11] are application tools for Web data collection, elementary and global evaluation, analyses and recommendation that will use the metric repository services at different levels.

Secondly, with regard to the cataloguing environment we are, on the one side, designing and building a virtual community mechanism that basically will provide a Web-based synchronous and asynchronous collaborative facilities for discussing, agreeing, and adding approved metrics to the repository. On the other side, we are designing and implementing the cataloguing application with a Web-based browse and query functionalities for consultation and reuse based on Semantic Web principles[12]. This catalogue of metrics will be beneficial for evaluators and other stakeholders in different milestones of the Web lifecycle as above commented.

In a future work, conceptual, navigational, and architectural design aspects to the cataloguing environment as well as its usefulness for Web-based enterprise engineering processes will be thoroughly dealt with.

Acknowledgments

This research is supported partially by the UNLPam-09/F022 research project. Also by the CYTED Program, in the VII.18 WEST (Web-based Software Technology) Iberoamerican project.

References

1. S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige, Web Engineering: A New Discipline for Development of Web-based Systems, LNCS 2016 of Springer-Verlag, Web Engineering: Managing University and Complexity of Web Application Development., San Murugesan, Yogesh Deshpande Eds., pp. 3-13 (2001)
2. J. Nielsen, The Alertbox, Available online at: http://www.useit.com/alertbox/ (1995-2002)
3. E. Mendes, N. Mosley and S. Counsell, Web Metric –Estimating Design and Authoring Effort, IEEE Multimedia, V. 8, Nº 1, pp. 50-57 (2001)
4. L. Olsina, J. González Rodríguez, G.J. Lafuente, O. Pastor, Towards Automated Web Metrics, Proc. of VIII Quality Brazilian Workshop, RJ-Br, pp. 74-86 (2001)
5. A. Scharl, Evolutionary Web Development, (Applied Computing) Springer (2000)
6. ISO/IEC 9126-1: 2001 International Standard, Software Engineering - Product Quality - Part 1: Quality Model (2001)
7. B.A. Kitchenham, R.T. Hughes and S.G. Linkman, Modeling Software Measurement Data, IEEE Transactions on Software Engineering, 27(9), pp. 788-804 (2001)
8. N.E. Fenton and S.L. Pfleeger, Software Metrics: a Rigorous and Practical Approach, 2nd Ed., PWS Publishing Company (1997)
9. L. Olsina, G.J. Lafuente, D. Godoy and G. Rossi, Assessing the Quality of Academic Websites: a Case Study, New Review of Hypermedia and Multimedia (NRHM) Journal, Taylor Graham Publishers, UK, Vol. 5, pp. 81-103 (1999)
10. L. Olsina, G.J. Lafuente and G. Rossi, E-commerce Site Evaluation: a Case Study, LNCS 1875 of Springer, 1st International Conference on Electronic Commerce and Web Technologies, EC-Web 2000, London, UK, pp. 239-252 (2000)
11. L. Olsina and G. Rossi, A Quantitative Method for Quality Evaluation of Web Sites and Applications, IEEE Multimedia, Vol. 9, Nº 4, pp. 20-29 (2002)
12. M. de los A. Martín, M. F. Bertoa, A. Vallecillo and L. Olsina, Towards a Semantic Web Approach for Metrics Cataloguing, Submitted paper (In Spanish) (2002)
13. ISO/IEC 14598-5:1998 International Standard, Information technology -- Software product evaluation -- Part 5: Process for evaluators (1998)
14. H. Zuse, A Framework of Software Measurement, Walter de Gruyter, Berlín-NY (1998)
15. S. L. Pfleeger, Lessons Learned in Building a Corporate Metric Program, IEEE Software, Vol. 10, No. 3, pp. 67-74 (1993)