Towards a Reusable Repository for Web Metrics

Luis Olsina¹, Guillermo Lafuente¹, Oscar Pastor²

¹GIDIS, Department of Informatics, Engineering School at National University of La Pampa
Calle 9 y 110, (6360) General Pico, La Pampa, Argentina
E-mail [olsinal, lafuente]@ing.unlpam.edu.ar

²Department of Informatics Systems and Computing at Polytechnic University of Valencia,
Camino de la Vera s/n. Aptd. 22.012. E-46071 Valencia. Spain
E-mail opastor@dsci.upv.es

Abstract. In this paper, we introduce a catalogue template as one of the building blocks for a repository of metrics. Particularly, starting from a conceptual quality framework, we thoroughly discuss a catalogue template for product metrics instantiating it with some Web metrics. A catalogue of metrics basically allows evaluators and other stakeholders to have a consultation and reuse mechanism, which starts from a sound specification of the entity type, the attribute definition and motivation, the metric formula, criteria, and application procedures, 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., both in the development and maintenance phases. Effective and full-fledged quality assurance processes require not only strategic but also technological support as well.

Keywords: Repository of Metrics, Quality framework, Web metrics, Cataloguing Tool.

1. Introduction

Due to the unceasing growth of Web sites and applications, 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 was and is one of the challenging processes in Software Engineering as well as for the Web Engineering, as a new discipline [5]. 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 Web sites and applications 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. In this direction, providing a repository of metrics and a cataloguing tool basically allow to have a consultation and reuse mechanism, starting from a sound specification of the entity type, the attribute definition, objective and motivation, the metric formula, criteria, application procedures, metric unit and scale type, among other elements.

As Web sites have grown both in interaction and functionality, they have changed from just 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 ([6, 7] among others), and metrics for the evaluation of Web sites and applications [4, 10, 13], most of them lack a well-defined specification framework and, even worse, a strategy for consultation 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 [3], in the 9126-2 and 3 draft documents. However, most of the product metrics informed in drafts are either enough generic or not customized for Web products; furthermore, the paper-based template of information used to describe those metrics is not enough complete as we will analyze later on.

The contribution of this paper resides in the definition of the template items or attributes which can be useful for cataloguing metrics to different entity types (though we focus the discussion mainly on the product template). Firstly, we propose a conceptual quality framework for entity types grounded in the ISO 9126-1 quality model. When a metric is defined and specified, it is necessary to previously know what attribute of what entity type it will measure. 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 metrics catalogued up to now for pages and sites 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
[10] 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 the one hand, and a Web-based robust
query functionality for consultation and reuse, on the other. Unfortunately, in recent Web research
initiatives, Web metric specifications and cataloguing environments as technological support for
quality assurance processes have often been neglected.

The rest of the paper proceeds as follows. In Section 2, we propose a conceptual quality framework
for entity types, quality models and metrics. In Section 3, starting from this conceptual quality
framework, we thoroughly discuss a catalogue template for product metrics, and some design aspects
of the cataloguing tool. In addition, we comment diverse catalogued Web attributes and metrics,
 focusing the attention on those automated ones. Finally, concluding remarks and future works are
drawn.


In the Web Engineering as in the Software Engineering field, the integral evaluation of attributes of
different entities is not an easy endeavour. In its broadest sense, it is difficult to consider all the
characteristics and mandatory and desirable attributes of a process, resource, product (like a Web site
or application), or product in use, if there are no sound quality framework, models and methods in
order to allow evaluators specify systematically those characteristics and attributes. As a first step, the
definition of a conceptual framework of quality that serves as a guide in the classification process of
entities, models and associated metrics should be considered important. 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 the
quality assurance process. Next, we outline some aspects of the conceptual framework without
discussing the details in depth, for space reasons.

![Figure 1. Conceptual quality framework regarding different entity types and potential quality
models of interest to Software and Web Engineering fields (adapted from ISO/IEC 9126-1, p.3).](image_url)
In this schema, adapted from the ISO/IEC 9126-1 quality model framework [3], 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 the figure 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. Next, we give a short description of each one of these factors involved in the conceptual framework.

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 [1, 3], models the same one in consideration of 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 behaviour and environment. The external quality is measured through external metrics where the product is generally in operation state; the important thing here 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 is tried to be measured, considering the 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 sub-characteristics for each one. The hierarchical model can be specified as a tree compounded of characteristics, sub-characteristics, 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 be obviously measured as a direct metric in the absolute scale. A possible indirect metric (regarding internal or external quality depending on the case) is: \( X = \frac{\text{OrphanPageCount}}{\text{PageCount}} \). 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 Web sites and applications [8, 9, 11]. In the next section we analyze, among other aspects, the different template items that should be considered for cataloguing purposes.

3. Towards a Repository of Metrics for the Cataloguing Tool.

For the above conceptual framework, it is necessary to have a catalogue that allow documenting and accessing metrics involved in the quality factor for each entity. A cataloguing environment can offer users consultation mechanisms from an on-line repository, among other functionalities. Users could consult and filtrate information with the purpose of obtaining the desired metrics in a quick way. For instance, they could query for the product entity and, specifically, for the Web site sub-entity, which automated metrics exist for broken links (let it be internal or external to the site), and for a potential beneficiary such as a maintainer.

3.1 Analyzing the Template Items.

One of the aims of this work is to analyze the template items in order to build a catalogue of metrics for quality factors. Although primarily thought for product metrics, it can be easily adaptable for other entity types. Table 1, shows the template items instantiated with an automated Web metric, namely, the Orphan Page Count metric.

The Title item serves to identify in natural language the attribute in the catalogue (and the Code as a unique key in the repository).
### Table 1. Template for cataloguing product metrics exemplified by the Orphan Page Count metric.

<table>
<thead>
<tr>
<th>Code</th>
<th>A catalogue code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Title (Name)</td>
<td>Orphan Page Count</td>
</tr>
<tr>
<td>Keywords / Alias</td>
<td>Dead-end Page Count, Orphan Page, Isolated Page, Web Metric</td>
</tr>
<tr>
<td>Attribute Definition</td>
<td>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.</td>
</tr>
<tr>
<td>Objective / Motivation</td>
<td>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 Obs. item).</td>
</tr>
<tr>
<td>Entity Type</td>
<td>Product</td>
</tr>
<tr>
<td>Sub-entity</td>
<td>Web site or page</td>
</tr>
<tr>
<td>Potential Quality Model</td>
<td>ISO/IEC 9126-1</td>
</tr>
<tr>
<td>Characteristic</td>
<td>Reliability</td>
</tr>
<tr>
<td>Sub-characteristic</td>
<td>Maturity</td>
</tr>
<tr>
<td>Potential Categories</td>
<td>Failures and Faults</td>
</tr>
<tr>
<td>Sub-categories</td>
<td>Page</td>
</tr>
<tr>
<td>Formula</td>
<td>X = #OP (Number of Orphan Pages)</td>
</tr>
<tr>
<td>Associated Metrics</td>
<td>None, because it is a direct metric</td>
</tr>
<tr>
<td>Attribute Type</td>
<td>External or internal, depending on the process lifecycle.</td>
</tr>
<tr>
<td>Application Procedure, Protocol</td>
<td>From a starting URL (of a given site page), recursively analyze all the pages, considering exclusively those that have at least an internal and not broken link, following this generic algorithm:</td>
</tr>
</tbody>
</table>

#### Preconditions
Starting from the initial URL of the Web site to analyze = URL\(_j\), Orphan\(_\text{page} = 0; \ j: 1.. \ \text{Page Count} \)

**Orphan\(_\text{pages} (\text{URL}\(_j\)): \#\text{Orphan\(_\text{pages}\)\(_{j}\}) = \#\text{Orphan\(_\text{pages}\)\(_{j+1}\)}**

**For each** page (URL\(_j\)) not previously analysed

**If** ¬∃ an internal (URL\(_{ji}\)) not broken **then**

\[ \#\text{Orphan\(_\text{pages}\)\(_{j}\)} = 1 + \#\text{Orphan\(_\text{pages}\)\(_{j+1}\)} \]

**else**

\[ \#\text{Orphan\(_\text{pages}\)\(_{j}\)} = \#\text{Orphan\(_\text{pages}\) (URL\(_j\))} \]

**end if**

**end for each**

```plaintext
end
```

**Alternative Procedures**
None

**Notes**
In order to perform the calculation the email address link (or other resources) will not be computed.

<table>
<thead>
<tr>
<th>Interpretation of Measured Value</th>
<th>X ≥ 0, the closer to zero the better</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Measure</td>
<td>Number of orphan pages</td>
</tr>
<tr>
<td>Scale Type</td>
<td>Absolute (it is a counting)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>To be determined from a site sample</td>
</tr>
<tr>
<td>Input to the Process</td>
<td>A Web site or page URL</td>
</tr>
<tr>
<td>Type of Data Gathering and Calculation</td>
<td>Measurement Tool</td>
</tr>
<tr>
<td>Potential Processes of Use</td>
<td>Development, Testing, Integration, Maintenance</td>
</tr>
<tr>
<td>Potential Beneficiaries</td>
<td>Developer, Tester, Maintainer</td>
</tr>
<tr>
<td>Observations/Comments</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 to identify clearly which the purpose of the attribute is, and which its utility is, that is to say, for what it serves.

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

The **Potential Quality Model** item indicates the stated quality model to which the attribute is specified. 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 **Sub-characteristic** belonging to the characteristic can be included as well.

The **Potential Categories** item serves as a way to classify metrics (such as metrics for Length and Size, Complexity, Faults, etc.), in order to help users searching and browsing the catalogue. For instance, the Failure and Fault category can be split it up into the Link **Sub-category**, among others.

The **Formula** item specifies how the X 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 an association of attributes and/or parameters). The **Associated Metrics** item is provided for indirect attributes, 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: \( X = \frac{(\text{OrphanPageCount} / \text{PageCount}) * 100}{\text{Percentage of Orphan Pages}} \); where **Orphan Page Count** and **Page Count** are as associated and catalogued metrics. On the other hand, the **Related Metrics** item is considered, mainly for browsing purposes.

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

The **Application Procedure and Protocol** element explains and specifies the procedural mechanism and possible protocol 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 the data collection and computation process. In addition, in this item or in the **Notes** one, we can add information about the protocol with the purpose of guaranteeing repeatability and reproducibility in the evaluation process [2]). Also other **Alternative Procedures** can be specified accordingly.

The purpose of the **Interpretation of Measured Value** item 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 of Measure** and the metric **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 [1, 15]. Also the overall **Accuracy** of the metric can be determined, e.g., considering the average hit and miss accuracies.

The **Input to the Process** item has the purpose to indicate which product such as document, page, site or application component serve as input to the metric and ultimately to the evaluation process.

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

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 if it can be reusable in different application domains. A possible ordinal categorization is: **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 Web domains.

The **Potential Processes of Use and Potential Beneficiaries** items indicate in what processes the metric could be used and to what participant roles would be targeted. The **References** element could...
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.

Up to now, 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 [8, 9] 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, Average of Links per Page*, among others. We are going to discuss some few automated metrics by tools implicitly highlighting some catalogue template items, for space reasons.

#### 3.1.1 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 2. 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.

**Preconditions**

Starting at the initial URL of the Website to analyse URL= URL\(_i\)

\[
#\text{Broken\_links} = 0; #\text{Internal\_broken\_links} = 0; #\text{External\_broken\_links} = 0;
\]

\( j: 1..\text{PageCount}. \)

**Broken\_links (URL\(_j\)):#Broken\_links**

**For each** link (URL\(_{ji}\)) of page with URL= URL\(_j\) not previously analysed

**If** (URL\(_{ji}\)) is broken **then**

**If** (URL\(_{ji}\)) is internal **then**

\[ #\text{Internal\_broken\_links} = #\text{Internal\_broken\_links} + 1 \]

**Else**

\[ #\text{External\_broken\_links} = #\text{External\_broken\_links} + 1 \]

**End if**

**Else**

**If** (URL\(_{ji}\)) is internal **then**

\[ #\text{Broken\_links} = #\text{Internal\_broken\_links} + #\text{External\_broken\_links} + \text{Broken\_links}(\) \]

**End if**

**End if**

**End for each**

**Return** (#Broken\_links)

**End**

**Figure 2. Specification of the algorithm that automates the Broken Link Count metric for static pages. This algorithm changes considerably for dynamic pages.**

Figure 2, shows the application procedure that automates the *Broken Link Count* metric. The Website MA tool [10] 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.
3.1.2 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. 2, 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 [4].

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

\[
\text{Percentage Broken Links} = 100 \times \frac{\# \text{Internal Broken Links} + \# \text{External Broken Links}}{\text{Link Count}}
\]

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 attribute).

3.1.4 Number of Different Broken Links. This metric is obtained analyzing 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. 2, has a slight change, i.e., just checking if the considered URL was visited before or not.

3.1.5 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:

\[
\text{Percentage Different Broken Links} = 100 \times \frac{\# \text{Different Broken Links}}{\text{Different Count}}
\]

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 3.1.10 attribute.

3.1.6 Number of Images with Alternative Text. Images give visual information that sometimes should be deactivated by the user for accessibility or performance reasons, for example, by disabling the browser’s image feature. A Web site should provide suitable mechanisms to visualize images optionally, without loosing all the conveyed information.

The ALT property (in the HTML code), links an alternative text with an image (or other objects in Web pages such as applets, sounds, etc.). This contributes to the readability of the page (even more the text could be read before the image is unloaded). However, the measure of this attribute does not guarantee the quality of the alternative text. Some text can be generated automatically when editing with tools like FrontPage, among others. On the other hand, this attribute is included in the WAI guidelines of the W3C [14].

3.1.7 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.

3.1.8 Percentage of Presence of Alternative Text. This indirect metric is calculated as follows:

\[
\text{Percentage Presence of Alternative Text} = 100 \times \frac{\# \text{Images with Alternative Text}}{\text{Image Count}}
\]

A more careful study on the impact of this metric it would be to observe the percentage of presence of alternative text in the images posted in the first levels of the tree (assuming a hierarchical structure), and/or the more visited pages of a site.

3.1.9 Different Image Count. This direct measure counts the non-repeated images in the site.

3.1.10 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.
3.1.11 **Page Count.** It is obtained counting the total number of static pages of a site both HTML and SHTML. It shows the initial size of the site according to the number of documents or pages; however, more elaborated metrics to measure size and length can be provided (e.g., considering the page size attribute as well).

3.1.12 **Average Links per Page.** It is calculated as the quotient between the **Links Count** and **Page Count**. This metric gives information about the interconnection density. In other words, it indicates how an average page is interconnected toward destination nodes.

3.1.13 **Page Size.** The size of a (static) page is measured considering all its images, sounds, videos and textual components. For each page, the size in bytes can be obtained. The size of pages is an important issue in order to appreciate the site efficiency as we posed in the following metric.

3.1.14 **Quick Access Pages.** The download time \( T \) is related with the size of a page \( \tau \) and the speed in the established connection line \( c \).

\[
T_{\text{Download}} = f(\tau, c)
\]

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 [6]).

\[
g(T_{\text{Download}}) = \begin{cases} 
\text{QuickAccess} & T_{\text{Download}} < T_{\text{max}} \\
\text{SlowAccess} & T_{\text{Download}} \geq T_{\text{max}} 
\end{cases}
\]

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.

| Table 2. Summary of attributes for different entity types that can be part of a repository of metrics. |
|---|---|---|---|
| **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 [1]. 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 WebApps * [4]. |
| 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). |
| WebApps | Program Types Count | Defined as the number of different programming technologies used to build programs in a WebApps. 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) | Defined as the quotient between the completeness level and the average completion time. |

* WebApps stands for Web sites and applications.
Finally, in Table 2, some few other attributes for the four 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 Tool.

Regarding the repository of metrics, we are designing and building a cataloguing tool which basically will provide a Web-based collaborative mechanism for discussing, agreeing, and adding approved metrics to the repository on the one hand, and a Web-based robust query functionality for consultation and reuse, on the other. Next, some general, rather not architectural design issues are commented.

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.

The Administrator user 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 responsible for the discussion forums of candidate metrics and, in definitive, in charge of updating the repository with the approved metric between the reviewers and the moderator, or ultimately to veto it and put it again into consideration.

The Moderator user 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-centered collaboration mechanisms shall be used both synchronous and asynchronous as well.

The Reviewer user is responsible for contributing and discussing in the definition of the different template items of the metric, 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 of this event to the administrator.

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). This will favour the rendering and the capture of formulas and specifications.

4. Conclusions and Future Work.

As Pfleeger says, “metrics are welcome when they are clearly needed and easy to collect and understand” [12]. 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 methodological 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 Website MA tool [10] 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 the cataloguing application with a Web-based browse and query functionalities for consultation and reuse. This catalogue of metrics will be beneficial for evaluators and other stakeholders in different milestones of the Web lifecycle as above commented.

Finally, even if we have discussed in this paper a catalogue template and the respective items for the product entity type, an adaptable and integral design that embraces all the items to the four entity types is easily doable. Furthermore, even if we have aimed and exemplified the template throughout this work with Web metrics (given our interest in this research line), readers can fairly see that the product catalogue template and the collaboration mechanisms discussed are applicable to different software domains. In a future work, architectural design aspects of the cataloguing environment as well as its usefulness for Web-based enterprise engineering processes will be thoroughly dealt with.

Acknowledgments.

This research is supported by the UNLPam-09/F022 research project. Also by the CYTED Program, in the VII.18 WEST (Web-based Software Technology) Ibero-american project. We kindly thank the thoughtful feedback given by Emilia Mendes, Department of Computer Science at University of Auckland.

References.