AN EVALUATION SUPPORT FRAMEWORK FOR INTERNET TECHNOLOGIES AND TOOLS

Sotiris P. Christodoulou, Despoina G. Tzimou and Theodore S. Papatheodorou
HPCLab, Computer Engineering & Informatics Dept, University of Patras, 26500 Rion, Patras, Greece
spc@hpclab.ceid.upatras.gr, tzimou@ceid.upatras.gr, tsp@hpclab.ceid.upatras.gr

ABSTRACT
In several stages of Web development, Web Engineers have to study, evaluate, compare and finally select one, among many, similar software systems/tools and theoretical Web resources, i.e. processes, methodologies, technologies and patterns. In this work we address this need of Web engineers and introduce an interactive and flexible Evaluation Support Framework (ESF) to facilitate the fast and accurate comparative evaluation of similar “items” against a set of requirements. ESF consists of an Evaluation Support Environment (ESEnv) and several Modules (ESMod) – one per class of “items”. ESEnv provides an efficient scoring methodology and an XML/XSL-based middleware in order to integrate all available ESMod, that are described in XML. Moreover, it allows the specification of evaluation parameters / requirements in two levels of detail through easy-to-comprehend Web forms and presents the results in a meaningful and concentrated way. Finally, we describe the full-lifecycle process for the specification, development and maintenance of an ESMod and demonstrate its usefulness.

KEYWORDS: Evaluation Support, Internet Tools, Web Engineering, XML.

1. Introduction
Web Engineering [6] is defined as the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based Information Systems (WIS) [8]. Web engineering is essentially defined as “matching the problem domains properly to solution methods and the relevant mix of technologies” [6]. Several process models have emerged lately for ensuring successful WIS development. Some of them come from the area of software engineering and they are tailored to the WIS special needs, like Rational Unified Process [12], Adaptable Process Model [11] and OPEN Framework [10]. Moreover, there are some hypermedia research originated processes like OOADM [13], RMM [9], Flavio [7], WebML [1], Christodoulou [4] and Conallen’s UML adaptation to Web [5].

It’s obvious from the very start that a Web Engineer has to choose a process among many and similar ones. Additionally, in several stages of these processes, he has to choose among several “items” like, software systems / tools and Web theoretical resources, i.e. processes, methodologies, technologies and patterns. Especially when comes to the implementation phase, several issues concerning Web technologies coming up, and the Web Engineer has to choose quickly and accurately. Just to mention some: data structure models, storage formats, authoring issues (e.g. XML, RDF, Semantic Web, RDBMS, OODBMS, XHTML, etc.), application logic / system integration issues (e.g. CGI languages, SSI techniques like ASP, PHP, JSP, SOAP / Web services implementations) and interface / user interaction issues (e.g. CSS/XSL, Plug-ins, Accessibility). The problem is getting even bigger if we consider that tools and technologies (standards, protocols etc.) are shifting extremely fast in the Web-Engineering world and their volume is big. Moreover, WIS projects span to a variety of application areas, stakeholders and a wide range of technologies. Thus they have very different requirements for processes, tools, technologies, etc. even for different components of the same WIS Project.

Thus, Web Engineers have to continually be in a process of learning and evaluating emerging tools and technologies and exhaustively study the recent research results, in order to have the knowledge and the skills to evaluate fast and accurately. This is a very time consuming process, it’s impossible for most of the Web Engineers to follow and explain the clear practical interest in web sites and magazines that offer reviewing and comparing articles.

In this work, we address the emerging need of Web Engineers for a decision support system that will also act as a knowledge base and taxonomy system for Web Engineering resources, i.e. software tools, processes, methodologies, technologies and patterns. To this end, we introduce an Evaluation Support Framework (ESF) to facilitate the fast and accurate comparative evaluation of similar “items” against a set of requirements specified by the Web Engineer. The idea of the framework was conceived by our works and observations on evaluating Hypermedia Application Development and Management Systems (HADMS) [3] and Web-based Document Directory Systems (WDDS) [2]. In next section we outline ESF key concept, define the main terms used in it and describe ESF evaluation lifecycle process.
2. ESF Concepts, Terms and Evaluation Process

The main objective of ESF is to support its users quickly and also correctly selecting an Item from a set of Candidate Items according to specific user parameters / requirements. By the term item, we mean a well-defined entity that can be used by a set of users for one or more purposes. The triplet of Item Class, Users Classes and Purposes, defines one Evaluation Support Module (ESMod), the key element of ESF. The basic concept behind ESF is the continuous production, delivery and maintenance of several ESMod, supporting the fast and accurate evaluation of various “items” of interest. Under web engineering concepts, such items can be, for instance, hypermedia design methodologies, development processes, UML design tools, XML Editors, DBs, scripting techniques etc. ESF consists of:

- **Several Evaluation Support Modules (ESMod)** – one per triplet. Users’ needs and requirements derived by the purposes are mapped into evaluation criteria and categories. Moreover, candidate items’ are evaluated against the criteria and categories. At the implementation level, all these are specified in a single XML file based on an XML DTD (see Appendix) provided by the framework.
- **A full-lifecycle process for ESMod specification, development, usage and maintenance.** This process incorporates phases, actions, milestones, workflows, roles and modeling guidelines

- **An Evaluation Support Environment (ESEnv)** providing:
  - An XML/XSL-based middleware, to integrate all available ESMod
  - An easy-to-comprehend Web-based GUI, for users interaction
  - An efficient scoring methodology

For each ESMod we define two roles:
1. **ESMod Users:** The members of the User Classes defining the ESMod.
2. **ESMod Experts:** The gathering of users’ needs, their analysis and definition of the Criteria, the selection of the Candidate Items and their evaluation are very critical and important aspects for the usefulness of the ESMod. Thus, these actions should be carried out by a group of people that are experts and very experienced in the field. These are the ESMod Administrators and we refer to them as Experts.

The lifecycle process of an ESMod (see Fig.1) is separated to four phases:
- **Phase 1: ESMod Specification:** Definition and specification by Experts and Users.
- **Phase 2: ESMod Development:** Designing, development and integration to ESEnv.
- **Phase 3: ESMod Usage:** The ESMod is used by Users in order to select one of the candidate items according to specific requirements.

![Figure 1. ESF Overview, based on the full-lifecycle of an ESMod](image)
• **Phase4: ESMod Maintenance:** The ESMod should always be up-to-date, thus actions in phases 1 and 2 should be performed periodically in order to incorporate emerging users’ needs, new features of the Candidate Items or new Items.

Phase1 consists of two actions.

• **Action1 – Conceptual Definition:** The first action for the Experts is define what to evaluate and for whom needs it. The milestone of this action is the ESMod Conceptual Definition, i.e. Items Class, Users Classes and Purposes. For the same Item Class, different ESMod, i.e. evaluation criteria and categories, can be defined for different users and kind of use (e.g. for specialists many criteria and detail categories can be defined).

• **Action2 – User and Usage Requirements Specification:** For each class of users the Experts have to gather their requirements from simple to more advanced ones, deriving from the purposes. These needs, together with items’ characteristics, research results and best practices in the field, should be organized in a requirements specification report and modeled using well-known requirements management techniques, like use cases. This report is the milestone of the action.

In the following sections, Phase2 and Phase3 are described in detail. To better demonstrate ESF usage, we developed HADMS [3] ESMod and provide some ESEnv screenshots while using it (see fig.3 and fig.4).

3. **Phase2: ESMod Development**

The outcome of this phase is the construction of a ready-to-use ESMod, based on the requirements specifications. This phase consists of two actions.

• **Action3 - Evaluation Categories and Criteria Specification:** Requirements specifications are mapped into Evaluation Criteria, classified under Basic and Cross Evaluation Categories.

• **Action4 - Candidate Items Selection and Evaluation:** a set of Candidate Items is selected and evaluated against the Evaluation Criteria and Categories.

3.1 **Action3: Evaluation Categories and Criteria Specification**

Based on the requirements analysis in the specification report, Experts define a set of evaluation criteria. For each criterion a type must be assigned which specifies the followings:

1. The way Experts evaluates the Candidate Items against each Criterion.
2. The way ESMod Users specifies evaluation parameters through the Criteria.
3. The ranking algorithm of the Scoring Methodology.

We distinguish 4 criterion types, as outlined below:

• **Scale:** Criteria of this type are considered to be quantifiable; their degree of requirement support / satisfaction by a Candidate Item can be indicated by a float number in [0, 1]. Experts must define the distinguished levels of support, starting from “No Support” and ending to “Full Support”. The number of the intermediate supporting levels depends on the Experts and the criterion. Experts should have in mind that the supportability of a criterion by an Item becomes more difficult to estimate as the number of the supporting levels increases. Moreover, if the supporting levels are too few, it won’t be possible for distinguishing similar items (they will get the same support rank for a criterion, even if the one is slightly better than the other). Based on our experience, for many criteria of this type it’s enough to have a supporting scale of: “No Support”, “Poor Support”, Good Support”, “Full Support”. After defining the supporting levels, Experts must associate a mark for each one.

These marks are float numbers between 0 (“No Support” mark) and 1 (“Full Support” mark). By default the marks for the intermediate levels are equally distributed between 0 and 1, e.g. “Poor Support”: 0.33 and “Good Support”: 0.67. However, Experts are able to define their own marks for the intermediate levels (in some cases it is reasonable) e.g. “Poor Support”: 0.50 and “Good Support”: 0.75. For criteria that accept continuous values, e.g. CPU speed, memory size, price, etc., ranges of such values should be defined as levels of a scale criterion. E.g. for a price criterion, we could have the following definition in XML: (the vref value is used for referring to the corresponding mark, in the evaluation of the candidate items later in the process)

```
<Values>
  <v vref="NS" mark="0">Don’t care about price</v>
  <v vref="R1" mark="0.2">5000-10000</v>
  <v vref="R2" mark="0.5">1000-5000</v>
  <v vref="R3" mark="0.7">500-1000</v>
  <v vref="R4" mark="1">Less than 500</v>
</Values>
```

• **Boolean:** Boolean are these criteria which can either be supported or not (they don’t have intermediate supporting levels). For these criteria, Experts define only the values of the two levels, e.g. “No Supported” / “Supported” or “No” / “Yes”, etc. The marks for the two levels are always 0 and 1 correspondingly.
• OneOfASet & ManyOfASet: There are some criteria that are not quantifiable, e.g. the platforms of a software tool. These criteria provide a set of possible values (e.g. Windows, Linux, MacOS, Irix, other), and when Experts evaluate an item against such a criterion, they assign one or more of these values to it. If the type of the criterion is OneOfASet, then it allows only one of its values to get assigned during evaluation.

Figure 2. An Example of Basic Categories Tree, Criteria, Cross Categories and weights

The criteria are classified under a Basic Categories Tree (see Fig. 2) where the criteria are the leaves and the Basic Categories (BC) is the nodes. The root of the tree is called Root BC. A BC may incorporate other BCs and individual criteria. This hierarchical classification / grouping of criteria will facilitate the ESMod users, later during the ESMod usage phase, to specify their evaluation parameters at different “levels of detail” and focus on specific areas of functionality of the items.

The Experts may associate a weight (a number between 0 and 1) with each branch of the tree, which indicates the importance of the child (BC or criterion) inside the father BC. The aggregation of all weights of the criteria under a BC should be 1. These weights will facilitate the ESMod to provide more accurate results, when the users want absolute evaluation of the items against all the criteria.

Additionally, the framework allows Experts to introduce some Cross Categories (CC), in order to group the criteria of various BCs, under different horizontal contexts, like extensibility, usability, reusability, openness, etc. Experts may associate a new weight with each criterion, which indicates the importance of the criterion inside the Cross Category. For simplicity’s sake a CC cannot include other CCs. There is a standard CC always-defined called “Item Properties” which groups all the non-quantifiable criteria of the BC tree, and plays an important role in the decision-making stage. There are no constrains on how many criteria should be included in a category, however, we suggest having more than 3 and less than 12.

In summarizing, during this action, Experts should provide:
• The tree hierarchy of the Basic Categories, the classification of the Criteria under them and the weights of the branches.
• The Cross Categories, the links to their basic Criteria and the new weights.
• For each Category and Criterion: A unique reference name across the specific ESMod, a name and a brief description. From now on, C stands for a criterion’s reference name and BC, CC stands for a basic and cross category’s reference name respectively.
• For each Criterion, provide a type and its possible values.

All this information is described in an XML file based on the ESMOD.dtd (see Appendix). We refer to this file as: ESMOD.xml and it’s the main milestone of this phase. The upper part of the HADMS.xml is included in Appendix.

3.2 Action4: Candidate Items Selection and Evaluation

During this action Experts select and evaluate a set of Candidate Items against each criterion and category. Here are some generic suggestions on the efficient selection of the Candidate Items:
• Include all the items that cover more than 70% of the evaluation criteria.
• Include the 3-4 most popular items.
• You may consider including an item, just because is the only one that covers an evaluation criterion.
• Keep the number of the Candidate Items not too small and not too big (5 to 12).

For each Candidate Item, Experts provide a unique reference name (CI stands for it), a full name, a brief description and a list of relative references (Web sites, papers, etc.). Afterwards, they evaluate the CI against each criterion, by providing the values EVALCI - a float, a string or a set of strings depending on the criterion type. Additionally, Experts can optionally provide notes or comments on their evaluations.

The evaluation should be also performed at the categories level. Thus, for each Basic or Cross category, each CI is ranked with an integer from 0 to 4, indicating the overall category support. Notes on the evaluation may be provided at this level also. Thus, the user of the framework will be able to compare the
items at a higher level (not have to go through each single criterion). It will be useful for users that want fast evaluations or don’t have the expertise to go through the criteria. We refer to the evaluation values for the BCs as $EVAL_{BC,CI}$ and for CCs as $EVAL_{CC,CI}$. Milestones of this action: (1) List of Candidate Items, (2) Evaluation of them against criteria and categories, along with Experts’ notes. This information is included in $ESMod.xml$, as declared in the $ESMod.dtd$. In the Appendix you can see the evaluation of OOHDM, against HADMS criteria and categories, as described in $HADMS.xml$.

3.3 ESMod Integration with ESEnv

ESEnv based on the $ESMod.xml$ file, automatically produces two HTML forms for the evaluation parameters specification at criteria and categories level. Additionally, other HTML pages are produced, like $ESMod.dtd$, $HADMS.xml$. In the Appendix you can see the evaluation of OOHDM, against HADMS criteria and categories, along with Experts’ notes. This information is included in $ESMod.xml$, as declared in the $ESMod.dtd$. In the Appendix you can see the evaluation of OOHDM, against HADMS criteria and categories, as described in $HADMS.xml$.

4. Phase3: ESMod Usage

Every time the Users of the Framework want to select an Item according to their specific needs, they have to use an appropriate ESMod. If the Experts have already defined such ESMod, Users can use it through the ESEnv. This Phase includes 2 actions:

- **Action5 – Evaluation Parameters Specification:** the users define the Evaluation Parameters either at Criteria or Categories level.
- **Action6 – Results Interpretation – Decision Making:** The users interpret the Evaluation Results and decide which Candidate Item to choose.

In-between the two actions, ESEnv calculates the evaluation results based on a scoring methodology, the user’s parameters and $ESMod.xml$, and presents them in a meaningful and easy-to-read way. The Users may repeat these actions several times, before coming to a decision.

4.1 Action5: Evaluation Parameters Specification

During this action, User selects to perform the evaluation either at criteria level or categories level, and access the appropriate form provided by the ESEnv (see Fig.1).

At criteria level, User specifies to ESEnv:
1. The criteria on which he wishes to base the evaluation and specifies the requirement for each criterion ($REQ_C$), depending on the criterion type
2. Which criteria requirements are critical

3. Which of the Candidate Items to evaluate
4. Whether the system will show all criteria in the results, or the selected only.

At categories level form, User simply specifies to ESEnv the basic and/or cross categories on which he wishes to base the evaluation and specifies the requirements for each of them ($REQ_{BC,CC}$).

Fig. 4 (see Appendix) shows the lower part of the evaluation parameters form at criteria level for HADMS ESMod. We have to clarify here, that for criteria of $OneOfASet$ type, the user is able to specify requirements like, “I want (OO or ER) model approach”. For $ManyOfASet$ criterion type, the requirement could be like “HADMS running on (Windows AND Linux)”, or like “HADMS running on (Irix OR Linux)”.

4.2 Scoring Methodology of ESEnv

The ESEnv introduces a simple and efficient scoring methodology that can be used either at criteria or category level. We present the overall scoring algorithm at criteria level, which is more complicated than the one at category level.

For each specified Candidate Item $CI$ the methodology calculates a support rank for each selected criterion $C$, according to $EVAL_{CC,CI}$ and $REQ_C$. This rank is represented as $SRANK_{CC}$. $SRANK$ stands for Supportability Rank and it’s a float number between 0 and 1 that indicates the coverage fraction (percentage) of the specified parameter for $C$ by the $CI$. The values of $SRANK_{CC}$ can be:

1. $C$: the $CI$ is covering or overriding the specified parameter for $C$.
2. $0$: the $CI$ doesn’t support at all $C$.
3. IN (0,1): the $CI$ is covering a part (fraction) of the specified parameter for $C$.

We outline below the calculation algorithms of $SRANK$ for each of the three criterion types (for Boolean: $SRANK = EVAL$)

**Scale:**

$SRANK = 1 - (EVAL / REQ)$

$SRANK = 1$ else $SRANK = 0$

**OneOfASet:** $REQ = a \ OR b \ OR ...$

$SRANK = 1$ else $SRANK = 0$

**ManyOfASet:** $REQ = a \ AND b \ AND ...$

$SRANK = (num of REQ values in EVAL values)/(num of REQ values)$

After, calculating the $SRANK$ for each criterion, we calculate the $CSRANK$ and the $CRSRANK$ (Critical Support Rank) for each of the Basic Categories as shown below:
for each specified Candidate Item (CI)
for each Basic Category (BC)
CriticalC_{BC,CI} = \text{number of critical C in BC sub-tree}
CriticalCSupported_{BC,CI} = \text{number of supported critical C in BC sub-tree, i.e. those with SRANK == 1}
CSRANK_{BC,CI} = \text{sum of SRANKs of critical C in BC sub-tree}
CSRANKBC,CI = CSRANK_{BC,CI} / CriticalC_{BC,CI}
NormalC_{BC,CI} = \text{number of non-critical C in BC sub-tree}
NormalCSupported_{BC,CI} = \text{number of supported non-critical C in BC sub-tree, i.e. those with SRANK == 1}
NSRANK_{BC,CI} = \text{sum of SRANKs of non-critical C in BC sub-tree}
NSRANKBC,CI = NSRANK_{BC,CI} / NormalC_{BC,CI}

The calculations at the root of BC tree, gives the overall supportability ranks for user parameters for each CI. The same algorithm is used for Cross Categories.

Note that the weights of the BC branches are not used. They only used when the user wants to observe the absolute evaluation of the items, i.e. evaluation against all the criteria with the maximum requirements. In this case, the weights are used to estimate the comparative score of each CI more accurately.

We calculate first the SRANK of critical (CSRANK_{BC,CI}) and non-critical (NSRANK_{BC,CI}) criteria. Then we calculate the normalized weights of critical criteria (Wc) to be double the weight of non-critical (Wn). Then the total SRANK_{BC,CI} of the BC sub-tree is calculated as the weighted arithmetic mean of NSRANK_{BC,CI} and CSRANK_{BC,CI} as shown above.

<table>
<thead>
<tr>
<th>C</th>
<th>EVAL_{C,CI}</th>
<th>REQ_{C}</th>
<th>SRANK_{C,CI}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.67</td>
<td>0.33</td>
<td>0.5</td>
</tr>
<tr>
<td>C2</td>
<td>0.33</td>
<td>0.67pain</td>
<td>0.5</td>
</tr>
<tr>
<td>C3</td>
<td>0.33</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td>C4</td>
<td>a, b</td>
<td>a AND c</td>
<td>0.5</td>
</tr>
<tr>
<td>C5</td>
<td>0.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C6</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Let’s see an example. Consider a BC with 6 criteria shown in Table 1. You can see the evaluation values of a CI (EVAL_{C,CI}) and the parameters of the User (REQ_{C}), who has selected C1, C2, C3, C4 and C6 criteria to specify his requirements. Moreover he specifies requirement of C2 and C6 as critical. The results for this BC category will be:

CriticalC_{BC,CI} = 3, CriticalCSupported_{BC,CI} = 1,
CSRANK_{BC,CI} = 1.83
TotalC_{BC,CI} = 5, TotalCSupported_{BC,CI} = 2,
NSRANK_{BC,CI} = 1.5
CSRANK_{BC,CI} = 1.83 / 3 = 0.61 (61%),
NSRANK_{BC,CI} = 1.5 / 2 = 0.75 (75%)
Wn = 1 / (2*CriticalC_{BC,CI} + NormalC_{BC,CI}) = 1/(2*3+2)=0.125, Wc = 2*Wn = 0.25
SRANK_{BC,CI} = 1.5 * 0.125 + 1.83 * 0.25 = 0.645 (64.5%)

The values in bold are the ones presented to user.

4.3 Action6: Evaluation Results Interpretation – Decision Making

The ESEnv presents all these results to the user in a meaningful table (see Fig.3). A red cell indicates that a critical requirement (row) is not supported by the CI (column). An orange cell indicates that a non-critical requirement is not supported by a CI. For each category you can see their scores either for the critical or total parameters under this category. The CIs are presented sorted (from left to right) by (1) the number of Critical parameters supported, (2) the CSRANK of Root Basic Category, (3) the total number of parameters supported and (4) the SRANK of Root Basic Category. The User should examine in detail the CI evaluations starting from the one at the left, which covers more of his critical parameters. The user focus on the weak points of the CI, i.e. red and orange cells, and its “Item Properties” cross category, that shows all the non-quantifiable criteria and decides whether to select it or not. If he needs to further study the item, he can use the references provided by the framework.

5. Future Work

The next step is to identify which stages of Web Development life-cycle mostly need of an ESMod and specify what is to be evaluated in this stage, i.e. classes of methods, processes, tools, technologies or a combination of them e.g. Navigational Design Method and a Support Tool. The hard part is to provide high quality specifications for the ESMod, i.e. criteria, classification, selection of “items”, evaluation of them, and keep them up-to-date continually (incorporate new items, emerging features and requirements, etc.). For this part, we hope that Web research community will support such a global ESF by providing significant research efforts on standardization of criteria and categories in diverse areas and by constituting several groups of experts to maintain the ESM.
REFERENCES


APPENDIX

ESMod.dtd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT BasicCategories (Category+)>
<ELEMENT CrossCategories (CrossCategory+)>
<ELEMENT CandidateItem (Description, References, (EvalCateg | EvalCrit)+)>
<ATTLIST CandidateItem RefName CDATA #REQUIRED
Name CDATA #REQUIRED>
<ELEMENT Category (Description, Category*, Criterion*)>
<ATTLIST Category RefName CDATA #REQUIRED
Name CDATA #REQUIRED>
<ELEMENT Criterion (Description, Values)>
<ATTLIST Criterion RefName CDATA #REQUIRED
Name CDATA #REQUIRED
Type (BOOLEAN | ManyOfASet | OneOfASet | SCALE) #REQUIRED
Weight CDATA #IMPLIED>
<ELEMENT Description (#PCDATA)>
<ELEMENT EvalCateg (#PCDATA)>
<ATTLIST EvalCateg categ CDATA #REQUIRED
note CDATA #IMPLIED>
<ELEMENT EvalCrit (#PCDATA | mv)*>
<ATTLIST EvalCrit cr CDATA #REQUIRED
note CDATA #IMPLIED>
<ELEMENT EvaluatedCandidateItems (CandidateItem+)>
<ELEMENT EvaluationFramework (Description, BasicCategories, CrossCategories+, EvaluatedCandidateItems)>
<ATTLIST EvaluationFramework ESTRefName CDATA #REQUIRED
ESTName CDATA #REQUIRED>
<ELEMENT References (ref+)>
<ELEMENT Values (v+)>
<ATTLIST mv note CDATA #IMPLIED>
<ELEMENT ref (#PCDATA)>
<ELEMENT v (#PCDATA)>
<ATTLIST v vref CDATA #REQUIRED
mark CDATA #IMPLIED>
```

ESMod.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<EvaluationFramework ESModRefName="HADMS" ESModName="Hypermedia Application Development and Management Systems">
<Description>
</Description>
<BasicCategories>
<Category RefName="METH" Name="Methodology Evaluation Criteria">
<Description>
This category includes criteria met for the evaluation of the methodologies of HADMS. In order to present them in a well-structured and meaningful way, we examined several methodologies. Although different methodologies follow various kinds and numbers of steps, a more careful examination shows that most of them are based on four basic steps, which are explicitly observed in the OOHDM. These steps are Conceptual Data Model Design, Abstract Navigational Model Design, User-Interface and Run-time Behavior Design and Implementation.
</Description>
</Category>
</BasicCategories>
<Category RefName="METH_A" Name="Common Criteria among the Three First Steps">
<Description>
There are some criteria that are common to the three first steps and are classified into a separate category.
</Description>
</Category>
```
The methodology must support the definition and maintenance of structural constraints (e.g. composites that may not contain themselves recursively) and guarantee their enforcement among hypermedia objects participating in any of the three design steps.

In order to provide extensibility and tailorability it is required that design evolution is supported, in each of the three design steps, with automatic propagation of the modifications from the step (design model) where they occur, to the other steps. The design evolution should cover the fundamental requirements for a hypermedia development environment.

This criterion refers to the ability to allow reusing and tailoring of application-specific types of objects among different applications.

The Object-Oriented Hypermedia Design Method (OOHDM) uses abstraction and composition mechanisms in an OO framework in order to allow a concise description of complex information items, and the specification of complex navigation patterns and interface transformations. In OOHDM, a hypermedia application is built in a four-step process (conceptual design, navigational design, abstract interface design and implementation) supporting an incremental or prototype process model. Each step focuses on a particular design concern, and an OO model is built.
Figure 4. ESMod HADMS – Criteria Form

<table>
<thead>
<tr>
<th>System Properties</th>
<th>Any of the selected</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The model approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Storage System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Kinds of hypermedia applications supported</td>
<td>WWW or Other</td>
<td></td>
</tr>
<tr>
<td>4 Platforms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Systems to evaluate: Select all, Unselect all

Show all criteria in the results, Show Results, Reset Form