Enhancing Interaction Design on the Semantic Web: A Case Study

José Antonio Macías Iglesias

***Abstract*— The use of RDF-based ontologies as knowledge repositories has become increasingly popular in the last few years. The semantic web has rapidly spread, appearing as a new challenge for knowledge sharing and automatic processing. However, the reality is that the power of the semantic web is still barely used. This is mostly due to the fact that the semantic web is a powerful but complex technology that most end-users cannot afford to use for their common problem-solving activities. This has probably made the semantic web to stay in the background of interactive technologies, unlike other new end-user-oriented paradigms (e.g., the so-called Web 2.0 and later approaches) that have very much increased along these years. Nevertheless, the semantic web can be considered as a highly valuable paradigm that has not been conveniently exploited yet. In this paper, we propose a semantic environment to exploit semantic interaction by end-users in order to help them access semantic information easily. We follow a Programming by Demonstration approach, where the user navigates and modifies HTML presentation of data and the system automatically infers changes to the underlying semantic models. Furthermore, we provide an evaluation of the interaction, including the most important results obtained for the proposed approach.**

***Index Terms*—Human Computer Interaction, User Centered Design, User Interfaces, Semantic Web.**

I. Introduction

WEB browsers have become a common platform for navigating through different data repositories along the distributed space of knowledge. Most of interaction today is carried out through web applications rather than desktop software. This fact has led to change the way users traditionally interact with software artifacts in many respects. In general, the web has brought a new paradigm of interaction, reducing the gap between the human's cognitive conception of task and its computational representation. This change is related to the considerable progress made over the last few years, where computer applications include much more sophisticated user interfaces that encourages user interaction to be one of the most important concerns in the design of today's software artifacts. Consequently, the new computing today is the shift from machine-centered automation to user-centered services, tools and information access, which is provoking traditional computing to change from what computers can do to what people can do with computers [1].

Related to web research, one of the most important breakthroughs over the last decade has been the Semantic Web, where the main idea is to have semantically-related representations of knowledge rather than static stand-alone information all over the web. This supports explicit expressiveness in representing and relating domain information by using semantic information comprising ontologies and RDF-data referring to them. However, Semantic Web technologies have principally focused on providing rigorous accuracy and mathematical coherence to the different standards, and the process of creating ontologies usually demands ontology-language specialists in order to create real semantic designs. Despite the efforts to build graphical tools (e.g., Protégé or Swoop) that assist experts in the process to create ontologies, this task cannot be yet realized in an intuitive way, and it remains as an interesting challenge to Human-Computer Interaction (HCI) research.

In particular, one of the main problems that avoid end-users to properly exploit the Semantic Web is the complexity that semantics implicitly have. Although end-users can be regarded as potential content creators, the Semantic Web is out of the scope for this kind of users due to the inherent difficulties in dealing with the underlying technology [2]. This fact turns collaborative end-user-oriented semantic applications into a real need today. Even web designers could benefit, when designing rich web user interfaces, from more robust domain models provided by the semantic web. However, most of the web designers are not expert on semantic-web languages, and usually they cannot afford to learn specific programming skills that are not directly related to their common problem-solving activities. It is worth mentioning that being a domain expert does not necessarily imply to be a programming expert [3].

Considering the above arguments, the implicit complexity of managing semantics can be considered as one of the main drawbacks of the semantic web that make it to decrease in popularity, also being commercially unsupported and overtook by new cutting-edge technologies and services comprising a real end-user-intended paradigm – e.g., the Web 2.0 (and later) based applications. This new end-user trend has made the

Manuscript received September 20, 2011; revised December 29, 2011; accepted February 1, 2012. This research has been supported by the UAM and the Madrid Research Council, project ID: CCG10-UAM/TIC-5772, and also by the Spanish Ministry of Science and Technology, project ID: TIN2011-24139

José A. Macías is with the Computer Science Department, Universidad Autónoma de Madrid. Cantoblanco, Madrid 28049, Spain (phone: +34 91 4976241; fax: +34 91 4972235; e-mail: j.macias@uam.es).

Semantic Web paradigm to slightly stay in the background, the industry and academy paying more attention to interactive proposals that provide feasible capabilities in knowledge sharing, collaboration and autonomy in managing services. Since Web 2.0 technology can be meant as an end-user oriented approach, this also enabled non-expert users to take part in this technology in a more effective way.

However, and compared to the Semantic Web, the Web 2.0 approach also have some notable drawbacks that make this paradigm unable to be meant as the interactive panacea. On the one hand, it generally lacks expressive capabilities to deal with meta-languages and set up structural relationships. On the other hand, it also lacks capabilities to easily represent semantic information and carry out more efficient inference processes on the information. This implies that, although the Web 2.0 paradigm has gathered programmatic functionality [4] and collaboration intended for the final user, the Semantic Web outweighs the specification of more expressive user interfaces for data managing, also enhancing the possibility to specify, model and design interfaces through a convenient model-based process [5].

An interesting HCI approach intended to bring together rich end-user interaction and semantic capabilities is the conceptual modeling of the user interface, which consists in splitting up responsibilities between domain expert and programmers to obtain a real trade-off between expressiveness and easy-of-use in interacting with semantic information [6]. The Model-Based User Interface Design (MBUID) paradigm [7][8] is an attempt to carry through such a trade-off. The implicit idea behind MBUID is to separate the conceptual level of a user interface, which leads consequently to the explicit specification of different aspects of the interface itself, such as domain knowledge, presentation, dialog and behavior. In this sense, the Semantic Web fits very well the commented interactive paradigm, since ontologies and RDF can be considered as domain information, as long as templates and other presentation elements can be seem as specific skins for domain data representation. This facility is somehow provided by wiki environments, although the interactive capabilities of most wikis – i.e., visualize and manipulate semantics, are still rather low [9].

II. Exploiting Semantics by End-Users

The use of RDF semantics provides an efficient way to model different aspects of web user interfaces. Conceptual models allow for complex relationships that can be formally defined. Such a conceptualization can be used to codify high-level semantic paths for automatic web-based interface generation, further characterization and reverse-engineering purposes [10].

We have previous experience in exploiting semantics to specify knowledge for building data models (domain models) used together with application or presentation models. Such a fact has informed our approach towards specifying complex knowledge focused on the interface's domain and presentation models, as well as working with XML-based languages that better fulfill our assumptions about knowledge distribution and sharing. More precisely, we work on combining RDF with Model-Based User Interface (MBUI) techniques, which emerged as a solution claiming to overcome several difficulties in automating the process of generating interfaces (e.g. redundancy, lack of encapsulation and reusability).

All in all, semantics are not enough to provide an efficient solution in order for end-users/domain expert to deal with daily problem-solving activities. We must thought of end-users as non-expert people, who do not have to have specific skills on programming or semantics. In this respect, a new approach arises in order to face the problem of easily accessing and managing complex information by non-expert users, namely End-User Development (EUD) [11][12]. EUD is focused on a user-centered approach, and can be thought of as a set of activities and techniques that allow people (including non-professional) to create or modify software artifacts or complex data. It has been demonstrated that End-User Development techniques reduce the gentle slope of complexity and make easy the way the user accomplishes tasks by means of computers. Programming by Example [13][14] is one of the more flourishing approaches concerning EUD, which aims at obtaining a satisfactory trade-off between ease-of-specification and expressiveness. Programming by Example has the potential to allow users to customize their applications and manage semantics. Rather than writing a program in a programming language or dealing with abstract specifications (e.g., ontologies or RDF-based code), users simply demonstrate how to perform actions and the system automates the whole process, inferring patterns that will be applied next time to similar behavior, generating code automatically when necessary.

EUD techniques can be applied to MBUID in order to relieve the user from having to deal with semantics and abstract languages. To carry out this challenge, it is necessary to provide with low-level abstract design environments such as WYSIWYG approaches that provide end-users with a real representation of data. The goal is to provide users with environments where they can easily manipulate the interface's objects rather than using complex visual or specification languages. This makes it possible to have a more accurately conceptualization of what the user is attempting to do at every step [6].

III. A Proposed Environment for End-Users to Easily Deal With RDF Semantics

Our research is mainly focused on applying MBUID and EUD techniques to the semantic web, empowering end-users to manage semantics without the necessity of using semantic-web languages (i.e., RDF) or abstract specifications (i.e., ontologies). Instead, there is an explicit separation of responsibilities: while ontology expert can add semantic knowledge using a specific tool (called PERSEUS), template developers can create presentation to render the knowledge

(using PEGASUS), allowing end-users to easily deal with the rendered semantic data using a WYSIWYG authoring tool (called DESK). The whole environment is principally intended to get a real trade-off between expressiveness and easy-of-use. This way, non-expert users can access the semantic information by manipulating the rendered data, and the system infers what data have been manipulated and automatically applies the changes to the RDF and the domain ontologies (i.e., following an automatic reserve-engineering mechanism).

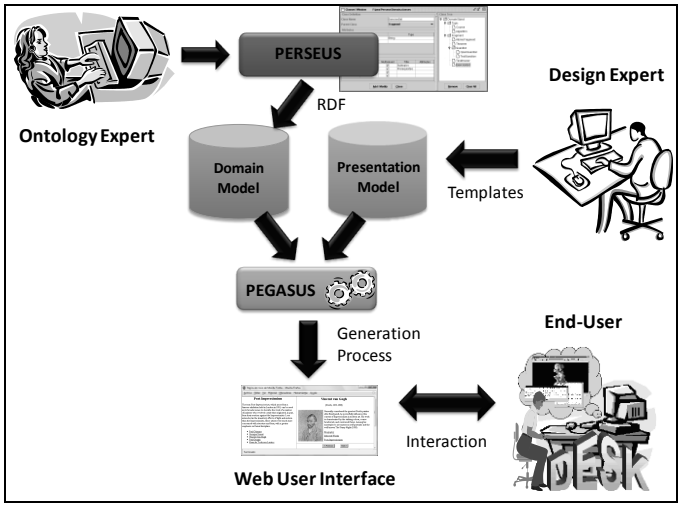

Fig. 1. Semantic environment made up of different tools: PERSEUS, PEGASUS and DESK.

In a nutshell, our EUD environment is made up of the following tools (see Figure 1):

• PEGASUS (Presentation modeling Environment for the Generation of ontology-Aware context-Sensitive web User interfaceS) [15][16] is a domain-independent system that helps to create a dynamic front-end for ontology-driven knowledge-based applications on the web. PEGASUS is based on a MBUI approach, and it supports the definition of made-to-measure ontologies for the description of domain knowledge. The MBUI mechanisms ensure domain independence by splitting up knowledge and presentation. This way, the system is capable of generating web pages on the fly by selecting domain objects and assembling them into HTML documents in response to the user's requests for concrete knowledge units.

• PERSEUS (Presentation ontology buildER for cuStom lEarning sUpport Systems) [10] is an interactive form-driven tool for the automatic generation of RDF-based files that contain domain information processed by PEGASUS. Using PERSEUS, the ontology designer can create custom domain-model designs by specifying the hierarchical structure of the ontology. This way, the authors are requested to create different classes and relate them one another by defining dependencies in terms of parent classes and semantic relations.

• DESK (Dynamic web documents by Example using Semantic Knowledge) [6][10] is an authoring tool supporting the customization of dynamically generated web pages in an environment that looks like an HTML editor. DESK is the client-side complement of the dynamic web page generation system, PEGASUS, which generates HTML pages. DESK helps end-user to modify the internal presentation model by editing the HTML pages generated by PEGASUS, avoiding non-expert users to deal with the PEGASUS modeling language. DESK is based on a Programming By Example (PBE) approach, where the system infers changes that affect whole classes of knowledge from user's actions on the presentation of a specific unit. DESK is able to identify domain values, fragments, and presentation constructs in the HTML code, from which it infers meaningful transformations. To carry out such a task, DESK observes the user's activity and, at the same time, it generates a monitoring model containing user actions together with its context for characterizing each action conveniently. Such information is sent to the DESK's server-side component, which processes the monitoring model, infers changes, generates suitable feedback and sends it back to the user. In a last step, DESK applies the inferred changes to the PEGASUS's underlying models.

Our semantic web based environment provides a natural way of interaction, mostly based on end-users with low or no skills on semantic web technologies but making use of semantics to enhancing the way they deal with daily problem-solving activities.

IV. Case Study

As a brief use case to demonstrate the functionality of our end-user semantic environment, let us suppose that we want to build knowledge to codify artworks for a museum web application. To carry out this task, first it is necessary to create the semantics (domain model) to code the domain knowledge information. Normally, this information is built up by domain (ontology) experts, as stated in previous sections.

In our system, the domain model (or domain ontology) is represented through an ad-hoc ontological RDF-like language, which is made up of domain classes and objects. Domain classes consist of a set of classes that suits the nature of a specific domain sketching the particular vision of a specific author on the domain. On the other hand, domain objects comprise object instances from the domain classes.

The domain model can be created by experts using the PERSEUS tool to better integrate the design cycle, or even by using other external ontology authoring tools that generate XML (such as Protégé). For instance, Figure 2 depicts new subclasses such as `Painter` or `Artwork` that inherit (dotted lines) from other existing abstract ones such as `KnowledgeItem`, `Topic` or `Fragment`

Domain classes can be defined with a high degree of freedom. Classes can be very generic or intended to reflect fine-grained concepts according to the designer's idea. Ontologies can include terms for subject information (e.g., a theorem has a statement and a proof), pedagogical information (e.g., lessons have levels of difficulty), and run-time state information (e.g., whether a concept is known by the user). This knowledge is captured by defining attributes for classes and relations between them. Two predefined root classes are

provided for ontology designers: `Topic` and `Fragment`. `Topics` and `Fragments` are different in such a way that the former are presented to the end-user in a separate page, while fragments can be inserted in a page together with other fragments and links to topics. A predefined subclass of `Fragment`, namely `AtomicFragment`, is also provided. It consists of HTML code, either in the form of a literal string, or as a URL from which HTML contents are to be retrieved. In addition to domain ontology, simpler data structures are defined by the designer to describe user profiles, platform characteristics, and other aspects considered relevant for adaptive presentation.

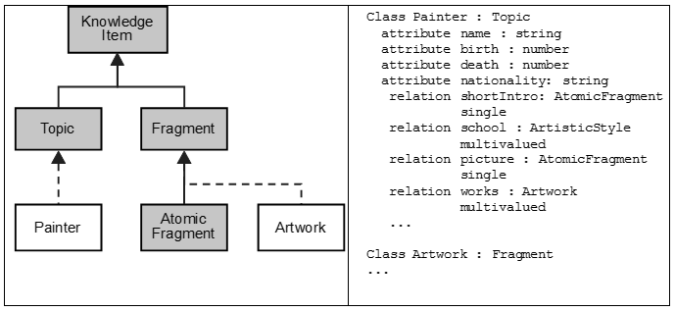

Fig. 2. Creation of semantic domain information.

Specific knowledge (domain objects) is constructed by creating instances of ontology classes and setting relations between instances, building semantic networks of topic and fragment subclasses, where multimedia contents are included as atomic fragments. For instance, the following example shows a simplified version of an object of class `Painter` that represents knowledge about Vicent van Gogh:

```
<Painter id="vanGogh" name="Vincent van Gogh"
         birth="1853" death="1890"
nationality="Dutch">
   <school>
      <ArtisticStyle ref="postimpressionism"/>
   </school>
   <picture>
      <AtomicFragment url="vangogh-pict.jpg"/>
   </picture>
   <shortIntro>
      <AtomicFragment> Generally considered
the greatest Dutch painter after Rembrandt, he
powerfully influenced the current of
Expressionism in modern art. His work is
characterized by the striking colour, coarse
brushwork, and contoured forms. Among his
masterpieces are numerous self-portraitsand
the well-known The Starry Night (1889).
      </AtomicFragment>
   </shortIntro>
   <biography>
      <AtomicFragment url="vangogh-bio.html"/>
   </biography>
   <works>
      <Artwork ref="starrynight"/>
      <Artwork ref="sunflowers1"/>
      <Artwork ref="irises"/>
   </works>
</Painter>
```

Attributes like `name` and `birth` identify domain object properties, whereas `school`, `picture`, `shortIntro`, `biography` and `works` are relationships with other domain objects (the `ref` attribute indicates referenced object IDs). Literal fragments can be inserted inline as XML elements (like `shortIntro`), or stored in external files (URLs) that are referenced in the code (like `picture` and `biography`).

In parallel with the creation of the domain model, web designers can create presentation templates (presentation model) to render the semantic information. Domain and presentation models are then processed by PEGASUS system in order to build HTML pages intended for end-users to navigate through. Presentations are defined by creating a template for each class of the ontology. Templates are defined using a textual language based on JavaServer PagesTM that allows the presentation designer to insert Java expressions (between <%= and %>) and control statements (between <% and %>) into HTML code. A template defines what parts (attributes and relations) of a topic must be included in its presentation, their visual appearance and layout. For instance, a simple template for the class `Painter` can be defined as follows:

```
<center>
<h2> <%= name %> </h2>
(<%= nationality %>, <%= birth %> - <%= death
%>) <br>
</center><br>
<center><table>
<tr><td valign="top" rowspan="5"> <%= picture
%> </td>
<td valign="top"> <%= shortIntro %> </td></tr>
<tr><td> <%= biography %> </td></tr>
<tr><td> <%= works %> </td></tr>
<tr><td> <%= school %> </td></tr>
</table></center>
```

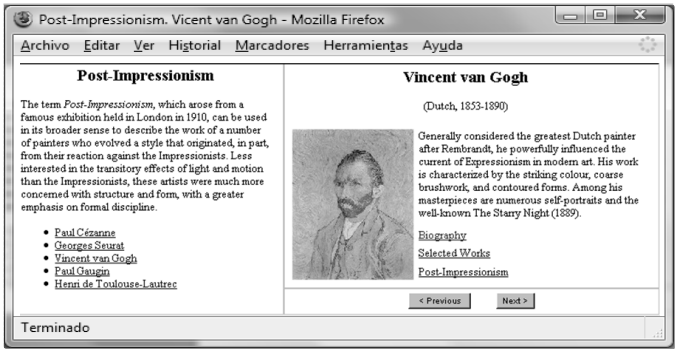

Fig. 3. Web page rendered by PEGASUS depicting the domain object vanGogh.

Dynamic presentation constructs are generated from simple descriptions at a high level of abstraction. In the example above, to include information about important works of art in a painter's page, the web designer only needs to refer to the works relation for the displayed object (shown in bold). The system automatically takes care of deciding whether to insert the corresponding works into the generated page, to generate a link for each one, or a single link for all of them, which style and/or visual effects are applied in the latter cases, and how all

the pieces are laid out. In doing so, the system analyzes whether the relation is simple or multivalued, the class of the involved topics or fragments, their state, and other conditions, if any, stated by the designer. The right panel in Figure 3 shows the web page generated by PEGASUS using this presentation template for the "vanGogh" domain unit described before. This knowledge unit can be visualized as part of a more complete ontology intended to make up an art repository for the museum application (left panel in Figure 3).

Once the presentation has been generated, end-users can modify the contents, style and structure by using DESK, the WYSIWYG authoring tool. The PEGASUS domain model is conveniently used by DESK in order to (a) identify pieces of domain contents in the web page, (b) establish relations between such pieces of knowledge, (c) select one (or more) of the involved knowledge items as the root domain object behind the web page, from which all other objects are referred, and (d) detect iteration patterns when the user lays out data over structured displays (e.g., records in a table).

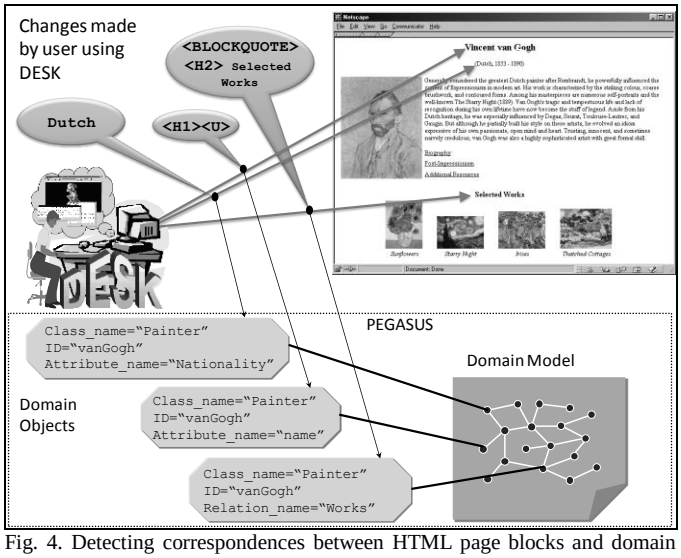

Fig. 4. Detecting correspondences between HTML page blocks and domain objects with DESK.

For instance, let us consider a web page like the one shown at the upper-right corner in Figure 4, where information about Vincent van Gogh is presented in a different layout. DESK is able to find that this page is displaying attributes (`name`, `birth`, `short biography`) and relations (`works`, `school`) of the instance `Painter`. If the user adds text, changes the style or the position of a piece in the document (e.g., the thumbnail image at the lower-right corner), DESK finds a description of this piece that relates it to the main object (`vanGogh`) in terms of the vocabulary defined by the application domain ontology (e.g. "the small-image attribute of the last element in the selected-works relation of the object with ID `vanGogh`"). This information is used by DESK to modify the presentation model for class `Painter` so that the change is permanent for all objects of that class. The `vanGogh` instance acts as an example for the user to see and change, in terms of how a painter presentation looks like.

The process of generation and interaction with knowledge is supported by a combination of forward and reverse engineering achieved by PEGASUS and DESK, respectively. On the one hand, PEGASUS performs the forward engineering process by generating web pages on the fly from a semantic network of ontology instances (the application data/knowledge). This process is carried out when the user implicitly requests viewing domain objects. These requests are internally generated from the navigational interaction of the user with an application supported by PEGASUS. To generate (and present) a concrete object, PEGASUS finds its class and applies the presentation model associated to the class to generate a web page where selected pieces of the object are displayed. On the other hand, DESK follows the inverse path: it processes the web page and locates the source of page fragments in the domain model, as well as the part of the presentation model that defines how the fragment was presented. This backward transition from syntactic blocks to semantic ones can be seen as a reverse engineering approach (see Figure 5), where the main concern is to support an automatic generation process from semantic and presentation models, as well as to provide end-users with an easy mechanisms to modify HTML content that will be transformed, later on, into semantic changes to the underlying models again.

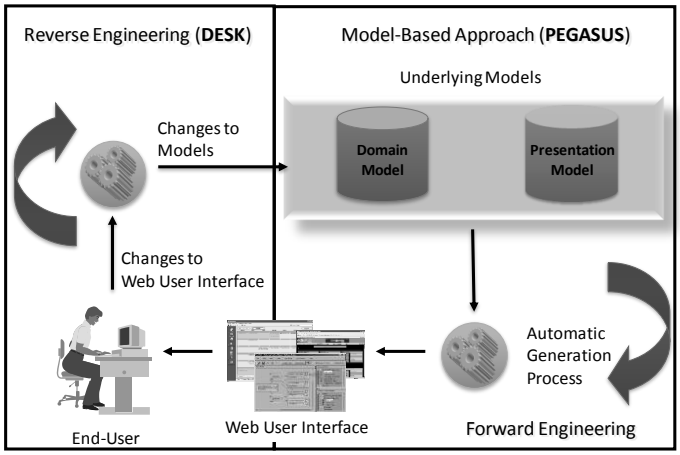

Fig. 5. Forward and reverse engineering processes followed by the semantic environment.

To carry through a successful reverse-engineering approach, efficient semantic information has to be provided in order to recover maximal information about the changes performed by the user [10]. To this end, it is important to find a specification that better fits these semantic requirements and allow for relationships between syntactic changes and semantic models. It is therefore necessary to set an explicit separation between contents and presentation, with the aim of avoiding missing information in the final generation step. This facility is automatically supported by our system throughout the forward and reverse engineering processes.

V. Evaluation and Discussion

To have a real perception about our system, we carried out a user experiment, mostly focused on the interactive part of our system (DESK tool) and intended to observe the degree of

satisfaction experienced by end-users while navigating and managing semantic contents by means of the rendered HTML documents. The main aim was to have some indicators about the user's satisfaction and clues to improve our system further. The experiment consisted in suggesting a task for end-users to achieve in our system and, in turn, studying the results and comparing them with the cognitive load in carrying out the task. Also, by using the Retrospective Testing [17], a user-centered testing technique that records the whole interaction between the end-user and the tool, all the interaction process was recorded on audio and video. This way, the execution of the tasks performed by users can be studied and analyzed in detail to extract further information about the interaction. The aforementioned technique included the Thinking Aloud protocol [18], which allows analyzing audio data of users during the interactive sessions, with the purpose of obtaining information about their thoughts, feelings and opinions while interacting with the system. This allows observing behavior patterns and/or phrases that may provide clues about the user's satisfaction while interacting with our tool.

To carry through the user experiment, we required 20 persons partaking in the activity, all having heterogeneous scientific backgrounds but no skills on ontologies and/or semantic-web languages. The participants' previous experience was mainly limited to creating and modifying simple HTML pages manually. However, all of them had some experience in WYSIWYG web authoring and navigation, which were the only skills required to carry out our experiment. The participants were given a 10-minute general introduction to the goal of the study, which principally consisted in carrying out modifications to the example presented in previous sections about the Painter class rendered through a presentation object about Vicent van Gogh (Figure 3). This way, the task consisted in modifying the given page by carrying out changes to page elements, independently of the order, to obtain a final version. We measure the accuracy of inference, the expressiveness and the freedom of design provided by DESK, placing neither restriction nor order on the way users carried out the customizations from the initial design. This implies that different users could accomplish the modifications by following different steps and thereby we expect the system to respond in different ways. In this case, the main objective of this was to get the maximum information about the user's opinion about the tool, as well as to corroborate whether the changes applied to the underlying models (throughout the reverse engineering process) can be considered coherent enough.

Once the participants finished navigating and modifying the presentation, users were asked to fill in a questionnaire based on User Interface Satisfaction [19] and another based on Perceived Usefulness and Ease of Use [20]. The questions in both questionnaires were selected and customized to mainly focus on DESK. Actually, in order to evaluate the user's satisfaction, we used a slightly modified version of the previously commented test. The standard version includes 27 questions, but it was reduced to 25 due to overlaps with the other test used. In general, both tests comprise an interesting approach to evaluating web interfaces in the context of end-user interaction. Since we initially had some experience in dealing with these tests, we have customized some part of them that can be considered useful for studying the user's satisfaction while interacting with web artefacts. Users were also asked to answer a set of open questions about the general perception of DESK, in order to obtain and identify additional comments about strengths and weaknesses to improve the tool.

The analysis of the questionnaires revealed that most users thought of DESK as a useful and easy-to-use authoring tool, very similar to other static authoring tools they may have used, but with an extra and powerful capability of authoring dynamically generated web pages. In particular, in a Likert scale including "Very High", "High", "Normal" and "Low" scores, 85% of users perceived the ease of use and usefulness of DESK between "Very High" and "High". Also, the satisfaction perceived by 90% of users was, considering the same Likert scale, between "Very High" and "High". In addition, the open questions brought to light that users considered DESK as a useful tool that can be applied to common daily tasks; to cite a few: authoring personal agendas and CVs, dealing with database-generated pages, managing dynamic on-line courses and teaching information and managing collaborative documents. According to the opinions requested, there exist conclusive evidences that motivate the increasing need to provide end-users with easy mechanisms for dealing with dynamically generated semantic web contents in real time.

In addition, the Thinking Aloud protocol used during the experiment helped us obtain other valuable information such as interaction styles including different types of customization carried out by users. For instance, syntactic customization greatly overcame the number of semantic ones (changing the structure and/or relationships of contents). This is due to the fact that syntactical aspects are easier to modify and have an immediate impact on the user's perception [3]. Concretely, this fact reflects that most changes made by users were related to syntactic modifications such as changing font style, size, colour, text justification, and so on.

On the other hand, and concerning the inference process, we internally evaluated that the hit rate shows 98% success in inferring the users' intentions through the reverse engineering process, which implies that DESK carried out most changes successfully when achieving the reverse-path analysis. The rest of 2% errors were due to some ambiguities appearing when the user's intentions were inferred. These ambiguities will be considered for future improvements on the system.

VI. Related Work

In general, there is a small number of existing systems that allow dealing with semantic information in form of syntactic HTML-rendered code and provide facilities for the final user to modify and navigate through. To introduce a few, FORTUNATA [21] is a system based on the generation of templates to represent visual elements of a domain ontology that are published and accessed through HTTP channels. Users can use this channel to access the semantics and the query templates, as well as to choose the most appropriate one for

their right visual representation. Related to the concept of transformations, RHIZOMER [22] is a system intended for the creation of Semantic Web applications. It manages RDF meta-data (resources, properties and literals) and semantically structures them in default contents (people, projects, publications, etc.). This way, the system generates navigational elements in a recursive way. Each meta-data fragment uses XSLT transformations to carry out its own visual representation. All in all, one of the main drawbacks concerning these systems is the implicit difficulty they undergo in order to face extensibility and re-modelling, since they lack an explicit separation among presentation and contents. Also, most of these systems are not explicitly focused on a EUD approach, since users have to finally end up dealing with ontologies in any case.

Other related systems are based on the idea of the Semantic Web Browser. Two main groups can be considered, consisting of browser and server centred approaches. Tabulator [23] belongs to the former group. DISCO [24] and WATSON [25] belong to the latter group. Browser-centred applications experiment some security problems, only solved by reducing the browser security level. Server-centred browsing does not have this limitation because browsing is indeed done by the server side. However, one common problem with semantic web browser is that still there is a small number of web sites that currently attach semantic annotations to their content since there are no real applications that use them. Our system does not include annotations in the presentation model, and it can render ontologies easily and independently from presentation. Furthermore, our system is intended for end-users and can be used in a secure way. On the other hand, some other systems like Piggy Bank [26] requires advanced technical skills, so that they are not designed for common users. Although Piggy Bank is designed for Web browsing, it renders semantic data in a fixed way (determined by Fresnel specifications). Other data browsers, such as Sindice [27], provide facilities to find other RDF documents on the Semantic Web that mention a particular thing. This kind of service might help ensure that the user experience is coherent — that is, that it includes all data the user expects it to. However, ensuring that a particular view of data is useful is another issue [9], since the presentation of data is also an important issue mostly uncovered by existing semantic systems.

Semantic Wiki-based applications, such as Shortipedia [28] and Semantic Wikipedia [29], provide recent trends in the creation and manipulation of knowledge, aiming to address problems related to unstructured accumulated information of conventional wikis. The main motivation is to make the inherent structure of a wiki – given by the strong linking between pages – accessible to machines beyond mere navigation. The idea is to enable structural organization of information resources with semantic association while providing diverse customized facilities, such as semantic search, multi-view filter, relevance-recommendation, etc. [30]. However, most of these applications have limited interaction capabilities that reduce the end-user's expressive abilities to visualize and interact with the information, since end-users have to deal with semantic query (sometimes using script-based languages) and the item-relation structure based on the domain model [31] rather than on a worked interactive presentation model, forcing users to somehow manipulate the ontologies or even the relations between sematic objects. Therefore, the separation of ambits between information (semantics) and presentation is not explicit in most cases, requiring advance users to successfully exploit the facilities of such systems. In addition, most of these systems do not implement simple mechanisms to modify the relationships or semantic contents from the user interface by using direct object manipulation.

Similarly, Semantic Portal systems [32] provide facilities to select, classify and access different information resources such as sites, documents and data for diverse target audiences. Some systems, such as OntoWeb [33], KAON [34] or ODESeW [35] to cite a few, have been proposed to automatically generate Semantic Portals from specific ontologies. However, these systems mostly exploit direct engineering processes (straight generation) rather than the reverse path that would enable end-user to create or modify semantic resources easily. Instead, most of these systems require information managers or advanced users to carry out authoring tasks. By contrast, our approach overcomes such drawback by carrying out automatic reverse mechanisms that allow reducing the cognitive load when creating or modifying information sources.

There are also a great variety of commercial web-development tools that provide some similar functionally to deal with XML languages and render some specific pieces of knowledge for web designs, including web-based languages such as HTML, CSS, XSL, XML, JSP, ASP and so forth. However, although these tools come with multiple tool bars and debugging facilities, they are not intended for end-users. That is to say, users have to act as skilled designers on web-based languages if they desire to modify procedural, content and presentation information, being subjected to the authoring formalisms. Some studies [36] revealed that, although much progress has been made by commercial web tools, most of the end-user tools that they reviewed lacked not functionality but ease of use. In general, the cognitive load in carrying out editing tasks using such environments is very high, because these commercial tools are mostly intended for professional designers rather than end-users. In general, end-users might just want to accomplish customization and easy changes to concrete parts of contents, expressing a preference for desktop-based tools that embrace drag-and-drop and copy-and-paste metaphors, and offers wizards, examples and template solutions. This implies reducing expressiveness in favour of increasing ease of use, something that is barely visible in existing commercial authoring tools today. In DESK, we provide easy mechanisms for authoring contents directly rendered from semantic information. This practically means that users do not have to deal with programmatic representations.

We followed a user-centred approach in order to provide DESK with less functionality than commercial tools, this way increasing its ease of use. On the other hand, DESK features reverse engineering processes intended to fulfil end-user

needs, modifying the underlying ontologies in PEGASUS and traversing the reverse path automatically, with no user intervention. Thus, end-users can easily customize and make partial changes to semantically-rendered contents. This helps users pay attention to syntactic changes in the WYSIWYG environment, and so reducing the cognitive load by avoiding specification languages and procedural information, which are automatically addressed by the system.

VII. Conclusions

Traditional computation has changed over the last years. Most computer-related technologies today are focused on end-users and their problem-solving activities rather than machine or process oriented concerns. This denotes an evolutionary trend even for recent technologies including the Semantic Web, which should be much closer to the final user, as well as specifically oriented to provide possible solutions for everyday computer users.

Our research is mainly focused on these concerns, bringing the gap between end-users and sophisticated computer technology in order to support practical solutions in both directions. In this respect, our contribution is supported by paradigms such as the Programming by Example and Model-Based User Interface design. In our experience, PBE and MBUI techniques can be combined together to relieve the user from having to deal with cumbersome languages, abstract specifications and complex development environments not intended for end-users. Certainly, this implies some reduction in the expressive power of the MBUI approach, since end-users do not need to manipulate declarative specifications, but rather to devote all their effort to dealing with rendered semantic information to fulfil their expectations in content customization. This motivated us to research on formal mechanisms in order to implement authoring tools that help users modify dynamic knowledge-based pages in order to deal with their daily, non-programming-oriented, creative problem-solving activities.

However, building dynamically-generated interfaces from examples requires elaborate data characterizations when the underlying domain knowledge has a complex structure, as it is the case in many semantic-based web applications and information systems. The usage of ontologies (i.e., explicit descriptions) to organize and share knowledge in such systems is becoming an increasingly popular approach. We propose to exploit these explicit models of domain knowledge as a highly valuable source of information for data characterization.

We have presented a semantic environment for data generation and manipulation that comprises three different tools. PEGASUS is a flexible system for the dynamic generation of semantic data in terms of custom domain knowledge representations. Our approach allows the specification of presentation independently from the elaboration of contents, enhancing presentation consistency and content reuse, and reducing the development cost. On the other hand, PERSEUS allows ontology RDF experts and web designers to create knowledge and presentation templates in form of both domain and presentation models that will be supplied to PEGASUS to render semantic presentations. Additionally, DESK enables end-users to deal with complex web content authoring. DESK includes Programming by Example facilities, which implies that users only have to provide the system with an example of what they wants to get and the system infers the changes to underlying semantic models automatically. DESK gets valuable information from user actions. This information is processed together with semantic domain knowledge in order to infer the knowledge necessary to provide the user with assistance during the authoring process. Changes are automatically performed in the server side by using both domain and presentation knowledge from PEGASUS. DESK attempts to infer maximal information from existing semantic knowledge that is independent from the application domain.

In order to corroborate our initial hypotheses, we have carried out a user test obtaining notable results on user satisfaction. This experiment demonstrated the hypothesis that is possible to reduce the gentle slope of complexity by supplying easy-to-use WYSIWYG user interfaces, but it has also revealed some limitations on expressive power, due to the fact that DESK is focused on concrete WYSIWYG representations rather than abstract ones.

We plan to test our system with other similar case studies to obtain findings about the applicability of the system in this specific domain. So far, we have tested our system with courses on Graph Algorithms, Object Oriented Programming, Art History and Geography. On the other hand, we are currently moving to W3C semantic web languages, instead of using an ad-hoc RDF version, to improve reuse and interoperability. We also plan to improve DESK to address much more sophisticated cases of inference. The idea is to obtain further findings that provide end-users with authoring assistance. Besides, the results here obtained will be also considered to carry out improvements.

References

[1] B. Shneiderman, “Leonardo’s Laptop”. The MIT Press, 2003.

[2] D. Djuric and V. Devedzic, “Incorporating the Ontology Paradigm into Software Engineering: Enhacing Domain-Driven Programing in Clojure/Java”; IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, vol. 42, no. 1, 2012, pp. 3-14.

[3] J. A. Macías and F. Paternò, “Customization of Web Applications through an Intelligent Environment Exploiting Logical Interface Descriptions”. Interacting with Computers. Elsevier, vol. 20, no. 1, 2008, pp. 29-47.

[4] E. Vlist, D. Ayers, E. Bruchez,, J. Fawcett and Vernet, “A. Professional Web 2.0 Programming”; Wrox Professional Guides. Wiley Publishing, Inc., 2007.

[5] E. Chavarriaga and J. A. Macías, ”A Model-Driven Approach to Building Modern Semantic-Web-Based User Interfaces”; Advances in Engineering Software, vol. 40, no.12, 2009, pp. 1329-1334.

[6] J. A. Macías and P. Castells, “Dynamic Web Page Authoring by Example Using Ontology-Based Domain Knowledge”; Proceedings of the International Conference on Intelligent User Interfaces (IUI). Miami, Florida, USA. January 2003, pp. 12-15.

[7] F. Paternò, “Model-Based Design and Evaluation of Interactive Applications”. Springer Verlag, 2001.

[8] A. R. Puerta and J. Eisenstein, “Towards a General Computational Framework for Model-Based Development Systems”; Proceedings of the International Conference on Intelligent User Interfaces (IUI). ACM Press, New York, 1999.

[9] T. Health, “How Will we Interact with the Web of Data?”; IEEE Internet Computing, vol 8, 2008, pp. 88-91.

[10] J. A. Macías, A. R. Puerta and P. Castells, “Model-Based User Interface Reengineering”; HCI Related Papers of Interacción 2004. Jesús Lorés y Raquel Navarro (eds.). Springer-Verlag, 2006, pp. 155-162.
[11] H. Lieberman, F. Paternò and V. Wulf, (eds), “End-User Development”; Human Computer Interaction Series. Springer Verlag, 2006.
[12] EUD-NET. Network of Excellence on End-User Development. See http://giove.cnuce.cnr.it/EUD-NET.
[13] A. Cypher, “Watch What I Do: Programming by Demonstration”; The MIT Press, 1993.
[14] H. Lieberman, H., “Your Wish is my Command. Programming By Example”; Morgan Kaufmann Publishers. Academic Press, USA, 2001.
[15] P. Castells and J. A. Macías, “An Adaptive Hypermedia Presentation Modeling System for Custom Knowledge Representations”; Proceedings of WebNet - World Conference on the WWW and Internet. Orlando, Florida; October 23-27. Published by AAC, 2001, pp. 148-153.
[16] P. Castells and J. A. Macías, “Context-Sensitive User Interface Support for Ontology-Based Web Applications”; Poster Session of the 1st. International Semantic Web Conference (ISWC’02), Sardinia, Italia; June 9-12th, 2002.
[17] J. Nielsen, J. “Usability Engineering”. Morgan Kaufmann Publishers, San Francisco, 1993.
[18] M. Nørgaard and K. Hornbæk. “What do usability evaluators do in practice?: an explorative study of think-aloud testing”; Proceedings of the 6th conference on Designing Interactive systems. ACM, University Park, PA, 2006, pp. 209-218.
[19] J. P. Chin, V. A. Diehl and K. L. Norman, “Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface”; Proceedings of ACM Conference on Human Factors in Computing Systems, 1998, pp. 213-218.
[20] F. D. Davis, “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology”; MIS Quarterly, vol 13 no. 3, 1989, pp. 319-340.
[21] M. Rico, O. Corcho, J. A. Macías and D. Camacho, “A Tool Suit to Enable Web Designers, Web Application Developers and End-Users to Handle Semantic Data”; International Journal on Semantic Web and Information Systems. IGI Global, vol. 6 no. 3, 2010, pp. 38-60.
[22] R. García, J. M. Gimeno, F. Perdrix, R. Gil, M. Oliva, J. M. López, A. Pascual, and M. Sendín, “Building a Usable and Accesible Semantic Web Interaction Platform”; World Wide Web, vol. 13, no. 1-2, 2010, pp. 143-167.
[23] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer and D. Sheets, "Tabulator: Exploring and Analyzing linked data on the Semantic Web"; Proceedings of The 3rd International Semantic Web User Interaction Workshop (SWUI). Athens, Georgia, USA, 2006.
[24] C. Bizer and T. Gau, “DISCO - Hyperdata Browser”; See http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/.
[25] M. d’Aquin, M. Sabou, E. Motta, S. Angeletou, L. Gridinoc, V. Lopez, and F. Zablith, “What can be done with the Semantic Web? An overview of Watson-based applications”. Proceedings of the Fifth Workshop on Semantic Web Applications and Perspectives, 15-17 Dec Rome, Italy, 2008.
[26] D. Huynh, S. Mazzocchi and D. Karger, “Piggy bank: Experience the semantic web inside your web browser”. LNCS. Proceedings of the International Semantic Web Conference (ISWC) vol. 3729, 2005, pp. 413–430.
[27] G. Tummarello, R. Delbru, and E. Oren, “Sindice.com: Weaving the Open Linked Data”; LNCS. Proceedings of the International Semantic Web Conference (ISWC) vol. 4825, 2007, pp. 552–565.
[28] D. Vrandecic, V. Ratnakar, M. Krötzsch and Y. Gil, “Shortipedia: Aggregating and Curating Semantic Web Data”; Journal of Web Semantics, vol. 9 no. 3, 2011, pp. 334-338.
[29] M. Krötzsch, D. Vrandecic, M. Völkel, H. Haller and R. Studer, “Semantic Wikipedia”; Journal of Web Semantics vol. 5, 2007, pp. 251-261.
[30] Y. Li, M. Dong, and R. Huang, “Designing Collaborative E-Learning Environments based upon Semantic Wiki: From Design Models to Application Scenarios”; Educational Technology & Society, vol. 14, no. 4, 2011, pp. 49–63.
[31] R. Valencia-García, F. García-Sánchez, D. Castellanos-Nieves, and J. T. Fernández-Breis, “OWLPath: An OWL Ontology-Guided Query Editor”; IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, vol. 41, no. 1, 2011, pp. 121-136.
[32] S. Staab J. Angele, “AI for the Web – Ontology-based Community Web Portals”; 17th National Conference on Artificial Intelligence and 12th Innovative Applications of Artificial Intelligence Conference (AAAI 2000/IAAI 2000), Menlo Park/CA, Cambridge/MA, AAAI Press/MIT Press, 2000.
[33] P. Spyns, D. Oberle, R. Volz, J. Zheng, M. Jarrar, Y. Sure, R. Studer and R. Meersman, “Ontoweb – a Semantic Web Community Portal”; Fourth International Conference on Practical Aspects of Knowledge Management (PAKM), 2–3 December, 2002, Vienna, Austria, 2002, pp. 189–200.
[34] G. Karvounarakis, V. Christophides, D. Plexousakis and S. Alexaki, “Querying community web portals”; Technical report, Institute of Computer Science, FORTH, Heraklion, Greece. 2000. See http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.pdf.
[35] O. Corcho. A. Gómez-Pérez, A. López-Cima, V. Lopez-Garcia and M. C. Suárez-Figueroa, “ODESeW. Automatic Generation of Knowledge Portals for Intranets and Extranets” 2nd International Semantic Web Conference (ISWC2003), 20-23 October 2003, Sanibel Island, Florida, USA.
[36] J. Rode, M. B. Rosson and M. A. Pérez, “End-User Development of Web Applications”. Lieberman, H., Paternò, F., and Wulf, V. (eds): End-User Development. Human Computer Interaction Series. Springer Verlag, 2006.

José A. Macías received the M.S. and Ph.D. degree in computer science from Madrid Technical University and Madrid Autónoma University, Spain, in 1999 and 2003, respectively.

He is a Permanent Professor in the Computer Science Department at the Madrid Autónoma University. His principal research area of interest is Human-Computer Interaction (HCI) and educational-related issues.

Dr. Macías is also part of different associations, such as AIPO (Spanish HCI Association), where he participates in the steering committee as Vice-President, and ACM-SIGCHI, where he participates as co-chair. He also obtained outstanding academic awards such as the "Outstanding Paper Award" conferred by IEEE Neural Networks Society.