Using ACUITy to Personalize Content in Semantic Web Applications*

Amy Aragones[1], Jeanette Bruno[1], Andrew Crapo[1], Marc Garbiras[1],

[1] General Electric Global Research, 1 Research Circle, Niskayuna, NY 12065 USA
aaragones@research.ge.com, bruno@research.ge.com,
crapo@research.ge.com, garbiras@research.ge.com

Abstract. Work domains such as maintenance and logistics planning are characterized by open-ended problem solving and large quantities of heterogeneous and distributed information. Problem-solvers in these domains can benefit from semantic web applications for work-centric decision support. In this paper we describe the need for such technology and the missing links in the current state of the art. We also present Adaptive Work-Centered User Interface Technology (ACUITy), still in its formative stage, as a way to meet this need.

1 Introduction

One can argue that the Semantic Web will deliver the biggest benefits in dynamic work domains that are subject to variability in decision-making behavior and information needs. One example of such a domain is logistics and maintenance planning for complex systems, which is characterized by large amounts of distributed and heterogeneous information that must be accessed and comprehended in a timely manner. It is also constrained by many business rules, geographically distributed collaboration and problem solving and other factors that influence decision processes. There is often a need to analyze what-if scenarios to evaluate potential courses of action and to prioritize work at the plan or project level as well as at the task level.

Increasingly, decision-makers in maintenance planning are faced with an automated, proactive information flow that challenges them to cognitively process, analyze and take action on a large amount of information. In order to prevent information overload, it is critical to find ways to provide users with the right information at the right time and in the right format. What constitutes the right information, the right format and the right time will almost always depend on the user's objective and will often depend upon the preferences of the user and his or her unique experiences.

In domains like this, semantic technology can provide a more truly collaborative relationship between humans and computer systems to help manage the type, amount and content of information accessed and applied to solve planning problems. Personalization is a cornerstone of this collaboration. The development of user models has

* This work was partially funded by the Air Force Research Lab under contract F33615-03-2-6300.

been a key focus for many working toward personalized or adaptive information delivery. Other work (e.g. [13], [10]) explores how to infer user models from web access data and how to adapt content display and navigation in hypermedia (e.g. [4], [7]). In addition to these research efforts, it is useful to take a work-centric approach to semantic modeling for useful personalization, beyond information modeling and user modeling based on personal characteristics, navigation and information retrieval. In particular, we are exploring three additional dimensions.

First, we agree with Schwarzkopf [11] that there is a great deal to learn about how users make sense of information in semantic applications beyond information retrieval. Others are working in the areas of semantically-enabled searching, information management and annotation [11]. We can extend these sense-making activities to how users visually reorganize, reformat, or change other properties of information. For example, users who manually fuse information using pencil and paper or using work aids like spreadsheets find implicit meaning in data by laying out chunks of content in different ways, changing the format of the information and highlighting information to find new relationships. We would like to capture those layout and formatting choices in a semantic model to help us understand how information becomes useful.

Second, one would expect that user preferences revealed through information selection and layout, and feedback from users about usefulness and meaning are largely second-order effects. We also need to represent the first order effects associated with the intrinsic nature of the work that needs to be done and the context in which users find themselves faced with the tasks they will attempt to execute. What is the state of the domain in which the users are working and what needs to be done to change that state? Such factors are critical in predicting the relevance of information and the effectiveness of a particular format. Thus, we require not just models of the users themselves and their perspective on the world, but also higher-level models of the work domain and the problems being solved.

Third, we also need to develop upper-level models of the interaction mechanisms necessary to achieve mixed interaction between human users and decision support systems. The Semantic Web represents an important step forward in the very difficult problem of representing information so that it can be interpreted by both people and artificial agents. This is a pre-requisite for any meaningful collaboration between people and machines. Another pre-requisite, we believe, is that our human-computer interfaces (HCI) be mixed initiative. It is not acceptable for the program to be in control with the human simply providing the inputs and receiving the "right" answer generated. It is also not adequate for the computing platform to simply provide low-level tools like word processors, spreadsheets or search engines to assist the human in performing cognitive tasks without any understanding of what the person is trying to accomplish. Collaboration requires a shared semantics all the way up to the goals and objectives of the work being performed and all the way down to the way information is passed between user and system.

Thus, models of the user, the work and the interaction mechanisms between the human and the system must be represented and related. This representation is consistent with previous work in adaptive systems and adaptive hypermedia [2], [3].

The Adaptive Work-Centered User Interface Technology (ACUITy) research program has provided us with an opportunity to investigate the modeling of users, work domains and HCI constructs to achieve a collaborative environment in which personalized content can evolve in a meaningful and responsive way. In this paper, we give an overview of our emerging modeling approach and the architecture we have implemented to tailor content according to the problem being solved and the preferences of the human problem solver. Our goal is to invite further discussion by documenting the current state of our technology and sharing ideas for future directions, mindful that there are still many open questions to be answered.

2 Adaptive Work-Centered Support

First introduced by Eggleston and Whitaker [5], the goal of a Work-Centered Support System (WCSS) is to "provide an integrated and tailored support system that…offers support to work in a flexible and adaptable manner" by customizing the user experience and interaction according to the situated context in which work is accomplished. A WCSS also emphasizes the importance of the user interface itself as a work aid, regardless of automated decision support applications that may also deliver helpful information or take action on behalf of the user. That is, having a workspace where information can be visually represented and manipulated can be of significant benefit to users, whether or not collaborative agents are helping out in the background.

We extend Eggleston and Whitaker's approach by taking advantage of semantic web technology to achieve personalized decision support. We have derived core upper level concepts from the three work-centered design principles introduced by Eggleston and Whitaker [6], including:

1. *The Problem-Vantage-Frame Principle:* Effective decision support interfaces should display information that represents the perspective that the user requires on the situated work domain to solve particular types of problems.
2. *The Focus-Periphery Organization Principle:* Information that is the most critical to the user in the current work context should be displayed in the focal area of a decision support display to engage the user's attention; referential information should be offered in the periphery of the display to preserve context and support work management.
3. *The First Person Perspective Principle:* The user's own work ontology (terms and meaning) should be used to characterize information elements in the interface display.

Our vision of an "Adaptive WCSS" is much like a supportive collaborator who watches the user and provides whatever is needed while warning the user of pitfalls to be avoided. To interact successfully, human collaborators must establish several conditions. First, clear communication is only possible if there is a common vocabulary (shared semantics). Second, given that shared language, collaborators must agree on a model of the problem to be solved. It is this model of the problem that allows relevant

information to be identified and communicated, alternative solutions to be identified or synthesized and evaluated, and the success of the process to be evaluated. In the next section we present the upper-level concepts we use as a framework to facilitate this kind of collaboration.

3. The ACUITy Architecture

ACUITy has three major components: the ACUITy Problem-Vantage-Frame (APVF) ontology, the ACUITy Controller and the User Interface (UI) Engine. The high-level architecture is shown in Figure 1.

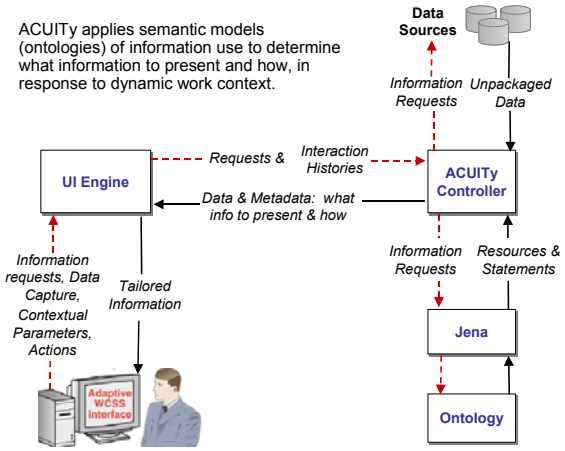

Figure 1. The ACUITy Architecture

3.1 An Introduction to the APVF Ontology

Work can be characterized as a series of unfolding problem-solving events [6]. Understanding the "intrinsic" nature of the work in terms of the type of problems being solved is critical to designing useful work aids. In the APVF ontology, we currently define a *problem* as something perceived not to be as desired; i.e., a state of affairs that requires the user to take action to change some aspect of the work domain. Although we regard the modeling of problems as a critical success factor, our modeling of problems is just in the beginning stages. There are many, many different ways that problems could be represented to varying degrees of detail. Thus, we decided to start from the other end of the issue and work backwards: to model how to communicate information between the application and the user and use that to discover what it is about the nature of problems that will be important to model. In other words, what should the system to know in order to effectively personalize information content and style?

Thus, at present, we have one-to-one mappings between problems and the sets of information needed to solve them, which we refer to as vantages. A *vantage* is a "window" into work domain information that provides the user with the particular perspective needed to solve a problem or problem set. More formally, a vantage is a collection of presentations of information that are relevant for a particular problem, along with their properties, which may include level of relevance, preferred position, size and other formatting considerations.

A WCSS frame "instantiates a vantage with specific display and control elements" [6]. We interpret this as the properties of a session with one or more vantages, each of which will have an associated problem set and a set of relevant information objects. In other words, a *frame* brings a set of vantages into relation via a set of common properties to mediate a particular work session. For each domain, the frame is specified as 'containing' vantage objects. The ACUITy Controller manages user sessions in terms of frames. With each user session, the user is given the option of creating a new frame instance or picking up their work from an existing session frame instance.

We enable a single frame to be populated by multiple vantages, one of which occupies the focus of the display while other secondary vantages are easily accessible in the periphery. We do this because a user may be working on a problem set that requires somewhat different and discretionary insight into the problem domain without losing the overall frame of reference. Figure 2 summarizes the relationships between the Problem-Vantage-Frame model constructs and work, user and interaction models.

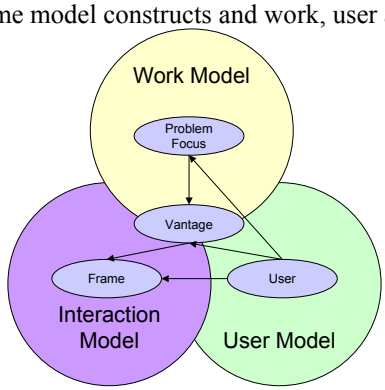

Figure 2. Problem-Vantage-Frame Concepts

A vantage is made up of a collection of presentation objects that are relevant for solving a particular problem set. A *presentation object* captures the what and how of displaying a certain set of information to the user. Presentation objects have the refining property "presentation nature", which specifies the general type of display. At present, ACUITy presentation natures include: 2d graphs, scrolling tables, text (including html and plain text documents as well as hyperlinks), display groups, external web applications, and user interaction objects such as forms, text entry fields, buttons, and various selection mechanisms (lists, check boxes, radio buttons, etc.). Although the type of the display object sets the base presentation style, the user may modify its look and feel unless precluded from doing so by ontology restrictions.

A *display object* is a type of presentation object that "encodes" work domain information for presentation to the user. A *script* captures the exact instructions for

retrieving information, whether from the ontology or from an external data repository. Work domain information objects may also be refined by filters, which mask some of the information in the data source.

An *interaction object,* a type of display object, allows the user to take some action that will have consequence or side effect, either in the client user interface or on the server. In the Web interface, client-side effects are implemented as client-side JavaScript. Server-side effects may be specified using a variety of *script* types. Note that the principles of WCSS encourage a mixed-initiative interface in which the user may take many actions to access information useful to the work at hand. Thus a particular state of the user interface might have many interaction objects.

Data maps define how to transform data returned by a script or a work domain information object to the format required by a display object. A set of standard maps is provided for the most common transformations. A map can also specify use of a custom plug-in transformation.

A property of a presentation object that affects its look and feel is referred to as a presentation *parameter*. Unless restricted by the model, presentation parameters are accessible to the user at run time for customization.

The APVF ontology contains many other concepts and properties, which may be used and extended to meet the needs of domain-specific applications. It is represented in the Web Ontology Language (OWL) and is described in detail in [1].

ACUITy and domain-specific work models also utilize upper-level and publicly available ontologies. These ontologies define concepts of time, physical versus abstract, problems, scripts, processes, and remote data sources. Scripting capability includes support of custom Java code that can implement data access, data transformation, or side effects. This facilitates integration of ACUITy applications with existing information repositories and computational models.

3.2 The ACUITy Controller

The ACUITy Controller is a Java class (with supporting classes) that provides an API to the APVF ontology. It provides special-purpose reasoning over this knowledge base to determine the set of information relevant to the problem at hand or the context of work performed. The ACUITy controller queries the ontology to understand where to find data, how to obtain it, and how to bundle it. The controller also accepts inputs from the client UI Engine and updates the ontology accordingly.

The ACUITy Controller selectively persists instance data, enabling the learning described below. It also integrates procedural knowledge (scripts). Some examples of the type of procedural actions performed by the ACUITy Controller include:

- User actions: e.g. create a new session (“frame”).
- Auto-populating a new frame with vantages and vantages with presentation objects.
- Mapping functions for graph presentation. For example, transforming the results of a query into a data series (e.g. create a candle chart), or transforming raw data into presentation attributes (e.g. numeric values to graph labels).

One can extend the procedural actions provided by the ACUITy Controller by creating user-defined scripts as new instances of action classes in the ontology.

The ACUITy Controller is designed for mixed interaction; perhaps the most fundamental interaction with the ACUITy Controller is to ask questions and tell new information. For example, a missing property is the name we give to an inconsistency between the necessary conditions imposed by restrictions on an OWL class and an actual individual of that class. For example, our ontology might define the class Mother as a Woman with a someValuesFrom class Person restriction on the property hasChild. If it is known that Jane is a Mother but Jane has no hasChild property with value an instance of Person, we may conclude that Jane has a missing property.

The ACUITy controller identifies three basic types of actions that can be taken in the event of a missing property:

1. Automatically create a new individual of the class identified as the range of the missing property
2. Ask the user to identify an individual to be the object of a new statement with the individual missing the property as subject and the missing property as predicate.
3. Execute a script as specified by the application developer (i.e. do something else that may or may not resolve the missing property)

This gives the ACUITy Controller the ability to allow, without requiring, the user to provide missing information.

3.3 The UI Engine

The User Interface Engine accepts metadata from the ACUITy Controller and creates the application's user interface. It sits between the end client and the ACUITy Controller and is meant to be the single interface point between the two entities. As such it interacts with the controller to request information and instruct the controller to update information and/or process scripts in response to the user's actions.

The UI Engine is made up of controller interface components, client platform agnostic UI component models, and UI renderer components. At this point we have implemented a web client renderer to produce a well-formed HTML document from the UI Engine.

4. Personalizing Content in ACUITy

4.1 Users Finish the Design

In an ACUITy application the users themselves finish the design by deciding what information they require to solve a particular problem – defining the vantage they need on the problem domain – and changing the characteristics of the information display in order to interact with the data more effectively. This type of direct adapta-

tion is regarded as a critical need in open-ended, information intensive work [4]. ACUITy captures in a centralized way the experience of users in open-ended problem-solving domains as they gather information from many disjoint sources not precisely identified at design time.

The user can reconfigure the display by adding and removing presentation objects. The approach can be extended to permit ad-hoc additions of information sources. Customization of information display includes, but is not limited to, the hiding, ordering, and sorting of data table columns, the selection of graph series types, e.g., line versus bar, color, and labels, and the type of enumerated selection lists, e.g., drop-down list versus checkbox versus tabs. The user can duplicate and then modify visualizations as desired. Information in disjoint tables and graphs can also be brought into relation by the creation of shared highlight regions, similar to data brushing in statistical graphics.

4.2 Learning Defaults and Patterns

Learning from accumulated instance data is an implicit benefit of the semantic modeling approach taken by ACUITy. As users customize the content and visual characteristics of the information that they view in particular problem-solving settings, these changes are stored in the ontology with their context. This past history creates the opportunity for a reasoner to infer what information content is most appropriate based on new information that was unavailable during the initial design of the web application. This special purpose reasoner then uses this instance data to personalize both default content and appearance for new sessions with similar contexts.

Learning can occur at different levels. A single user's preferences can be learned from that user's past behavior. When user models include grouping of users according to role or other shared attributes, learning can occur across peer groups. Sample size and predominance thresholds can be used to control the conditions under which learned values are used. This level of learned defaults will allow new users to benefit from the experience of more experienced users.

Learning a default from instance data is the recognition of a simple pattern. While not yet implemented in ACUITy, peer group learning will provide the basis of establishing "best practice" information displays. Recognizing beneficial patterns of usage across groups of users can lead to new classes of display objects explicitly available to developers and users, whereas learned defaults are only implicitly available. Abstraction of useful patterns might even extend across application domains.

5. Applications Created Using ACUITy

We have used ACUITy to prototype several web applications of interest to General Electric, Lockheed Martin and the US Air Force. From a developer's perspective, our preliminary experience is that ACUITy is a powerful and very flexible environment for exploiting semantic technology to create real-world decision support sys-

tems. The APVF ontology provides developers of new web applications a starting point from which they can create information-rich displays by relatively simple model extensions – essentially one models the UI rather than programs it. With respect to the user-interface, the developer is also "finishing the design." For example, a new data table can be added to a display through a few simple steps. The behavior and attributes necessary for the table to be constructed and displayed, as well as those that allow the user to customize the table display according to their preferences, are inherited.

We have also prepared a Hello World application that will accompany the release of ACUITy to open-source in 2006. The Professor/Student Course Management (PSCM) application illustrates the use of semantic technology to implement work-centered decision support and the benefits of ACUITy to both application users and developers. We will demonstrate this application at the European Semantic Web Conference 2006 Poster and Demo Session.

Our prototypes are currently in various stages of user testing and we expect that the results of those evaluations will guide future development on the reasoning algorithms by which user preferences are learned. These initial systems will also accumulate semantically-tagged instance data that we can use to better understand the relationship between problems and vantages and the ways in which decision-makers make sense of and use information.

6. Discussion and Future Directions

As mentioned earlier, our modeling of problems is only in the early stages. A significant challenge in this regard is one of observation: how can we detect the type and nature of the problem on which users are working and the domain context? In our current work domain of interest, maintenance and logistics planning, there are some very interesting possibilities. New complex systems, such as aircraft, locomotives, medical devices and power systems are being engineered with sensors that detect properties of a deployed fleet of equipment. Information from these sensors can be used in prognostics and health management, which can potentially tell us a great deal about the problems users face day-to-day and the state of the domain in which they are working. Ambiguity is unavoidable in inferring what the user is currently working on, but a well-designed mixed interaction approach should serve to close the gap (i.e. perhaps the system can ask the user to confirm what he or she is working on).

We would also like to leverage existing efforts to expand our user models (for example, UserML and GUMO [9], FOAF [8] and work in semantic portals) and our representation of reasoning and rules for adaptation.

One of our most significant accomplishments to date has been the ACUITy architecture, in which new semantic constructs and reasoning algorithms can be rapidly prototyped in the context of work-centric web applications. This architecture is not only agile with respect to the logic that can be applied, but also with respect to the application domain. We also believe that the upper level concepts we have formalized in the APVF ontology are valuable contributions that can be expanded upon in

open source and linked to other efforts in user modeling, adaptation and domain modeling. One of our next steps is to develop an end user-friendly ontology editor for domain-specific extensions of our APVF ontology. We plan to model problems in more depth and implement more sophisticated reasoning and learning capabilities; we would also like to expand our ability to allow users to perform ad hoc information searches and queries and integrate with web services. Now that the ACUITy framework is in place, we believe that we are well positioned to make longer, quicker strides in those directions and demonstrate incremental value to end users and enterprises from semantically-enabled systems.

References

1. Aragones, Amy, Bruno, Jeanette, Crapo, Andrew and Garbiras, Marc. "An Ontology-Based Architecture for Adaptive, Work-Centered User Interface Technology," IP.com: http://ip.com/pubView/IPCOM000134526D.
2. Benyon, D.R.: Adaptive Systems; a solution to usability problems. User Modeling and User Adapted Interaction, (1993) http://www.dcs.napier.ac.uk/~dbenyon/umuai.pdf.
3. Brusilovski, Peter, Cooper, David W. Domain, Task and User Models for an Adaptive Hypermedia Performance Support System. Intelligent User Interfaces IUI'02, (2002).
4. Carmagnola, Francesca, Cena, Frederica, Gena, Cristina, Torre, Haria: A Multidimensional Approach for the Semantic Representation of Taxonomies and Rules in Adaptive Hypermedia Systems. PerSWeb05 (2005) http://www.win.tue.nl/persweb/index.html.
5. Eggleston, R.G., Young, M.J., and Whitaker, R.D.: Work-Centered Support System Technology: A New Interface Client Technology for the Battlespace Infosphere. Proceedings of NAECON 2000, Dayton, OH, 10-12 (2000) 499-506.
6. Eggleston, R., and Whitaker, R.: Work-Centered Support Systems Design: Using organizing frames to reduce work complexity, Proceedings of HFES 46th Annual Meeting, Human Factors and Ergonomics Society (2002) 265 - 269.
7. Fransincar, F., Houben, G.: Hypermedia Presentation Adaptation on the Semantic Web. Proceedings of the AH 2002, LNCS, Springer Verlag (2002) 85-94.
8. The Friend of a Friend (FOAF) project: http://www.foaf-project.org/.
9. Heckmann, Dominik, Schwartz, Tim, Brandherm, Boris, Kroner, Alexander: Decentralized User Modeling with UserML and GUMO. 10th International Conference on User Modeling (2005) www.l3s.de/~dolog/dasum/DASUM05.Heckmann.pdf.
10. Kay, Judy, Lum, Andrew: Ontology-based User Modeling for the Semantic Web. PerSWeb05 (2005) http://www.win.tue.nl/persweb/index.html.
11. Schwartzkopf, Eric: Personalized Interaction with Semantic Information Portals. http://km.aifb.uni-karlsruhe.de/ws/LLWA/abis/schwarzkopf.pdf.
12. Vicente, Kim J.: HCI in the Global Knowledge-Based Economy: Designing to Support Worker Adaptation. ACM Transactions on Computer-Human Interaction, 2 (2000).
13. Zhou, Baoyao, Hui, Siu Cheung, Fong, Alvis C.M., Web Usage Mining for Semantic Web Personalization. PerSWeb05 (2005) http://www.win.tue.nl/persweb/index.html.