The Web Services Component Model and HumanML

Building Personalization into the Emerging Web Component Architecture

Rex Brooks (OASIS HumanMarkup TC Vice Chair, Secretary, Webmaster),

Kurt Cagle (OASIS HumanMarkup Editor),

1. Relating Human Markup Language to the Web Service Component Model

1.0. The Human Markup Language

The **Human Markup Language** is an attempt to codify the characteristics that define human physical description, emotion, action, and culture though the mechanisms of XML, RDF and other appropriate schemas. Human Markup Language (or **HumanML**, its primary abbreviation) is intended to provide a basic framework for a number of endeavors, including (but, as with human existence itself, hardly limited to) the creation of standardized profiling systems for various applications, building a framework for describing emotional state and response of both people and humanoid animations (i.e., avatars), laying the foundation for the interpretation of gestures for both person-to-person and person-to-computer interpretations, the encoding of gestures and expressions to facilitate the better understanding of modes of communication, etc.

The **OASIS Human Markup Language Technical Committee** is currently working closely with a number of standards bodies and organizations to build the foundation and framework for subsequent development of a set of HumanML standards. The principle purpose of this paper is to provide an explanation showing how Human Markup Language and its associated technologies can work effectively with the Web Services model and the Web Services Component Architecture.

1.1. HumanML Personalization and Web Services

Perhaps one of the most immediate benefits that Human Markup Language provides for the development of web services is a consistent format (schema) for both basic and targeted personalization. **Personalization** in this context is defined to cover a couple of fairly distinct concepts:

* **Preferential Personalization.** , in which specific emotional states, aesthetic preferences and gestural actions are used to shape the interfaces between people and computer systems,
* **Habitual Personalization.** , in which habits (such as buying habits), interests, and activities of people define both the interfaces used for computer interaction, and more significantly, the content presented to them.

Preferential Personalization impacts the web service arena in a number of different ways, especially in the area of web component user interfaces: graphical, vocal, aural, haptic (touch related) and so forth. All of us, whether in a home or business context, shape our user interfaces in some manner when we can. We change the font size or family of text to suit failing eyes, put up backgrounds of sunsets or waves or cartoon characters, create folders and shortcuts to better order our information spaces. As more of the graphical (and non-graphical) interfaces that people interact with become dynamically generated in some XML grammar or other, the potential for personalization increases dramatically. Web Components will ultimately generate these interfaces, and by using the personalization capabilities that can be exposed from HumanML, you will have a consistent profile schema that can be used to shape the interfaces to the user, rather than the user having to adapt herself to the interfaces.

This has implications in both business and consumer interactions. For instance, people in business need to be able to access the information that describes the current state of a company (their own or their competitors'), and they need to be able to configure that information in meaningful, ways. In general, this configuration does not necessarily flow along job roles, though it can; the CEO may not normally wish to see the detailed tabular view of information that the accountant sees, preferring instead graphs showing comparative information, but if his company is being audited he almost certainly will want those in-depth tables.

In the context of single desktop environments, this is generally not a problem - most components can readily retain their display state. However, as people increasingly migrate between platforms - from multiple desktop environments to PDAs to web kiosk, to cell phones, to embedded intelligent toasters - the difficulty of maintaining personalization profiles increases dramatically. HumanML has a definite role here, along with architectures that can work effectively with such XML profiles.

This ability of personalization also makes it possible to create web components dynamically tied to both identity and preference in response to system considerations. As businesses become increasingly automated, ever more of that business's state becomes exposed as automation objects. This business state becomes increasingly complex as a consequence, while at the same time the ability of programmers to create specific "one-off" customizations also becomes increasingly problematic.

The ultimate solution is for the business state to provide its own view that can be tied into personalization profiles. By providing a consistent, standard interface for describing personal preferences, applications can be built that can configure themselves to the appropriate views dynamically. This becomes even more relevent as those views move into the declarative languages realms (XHTML, XForms, SVG, etc.), as these languages can be used to generate very rich customizable applications by building up the views on the fly.

Businesses can employ such systems in a number of areas - creating "starter" interfaces for beginning users to the specific business data environment which can grow as the user becomes more proficient, portal systems that can configure themselves to multiple devices (including phone sets) and users, secured systems that lock out sensitive information transparently, the ability to migrate a person's profile between divisions (or possibly between businesses, such as may exist in a strategic partnership) and so forth.

Habitual Personalization, which tracks behavior and history, is both as useful in this environment as it is contentious. Habitual personalization (or delta profiling) maintains a history of a user's actions and from this builds up a picture about such things as the level of proficiency of that user in a given area and then begins to change the interfaces to reflect perceived changes. Such systems already exist in a number of proprietary applications (such as smart menus that remember those actions that are most often repeated and makes them easier to reach) but currently there is no open standard for developing such profiling systems that could be used between disparate applications. This is another problematic area HumanML is being designed to address.

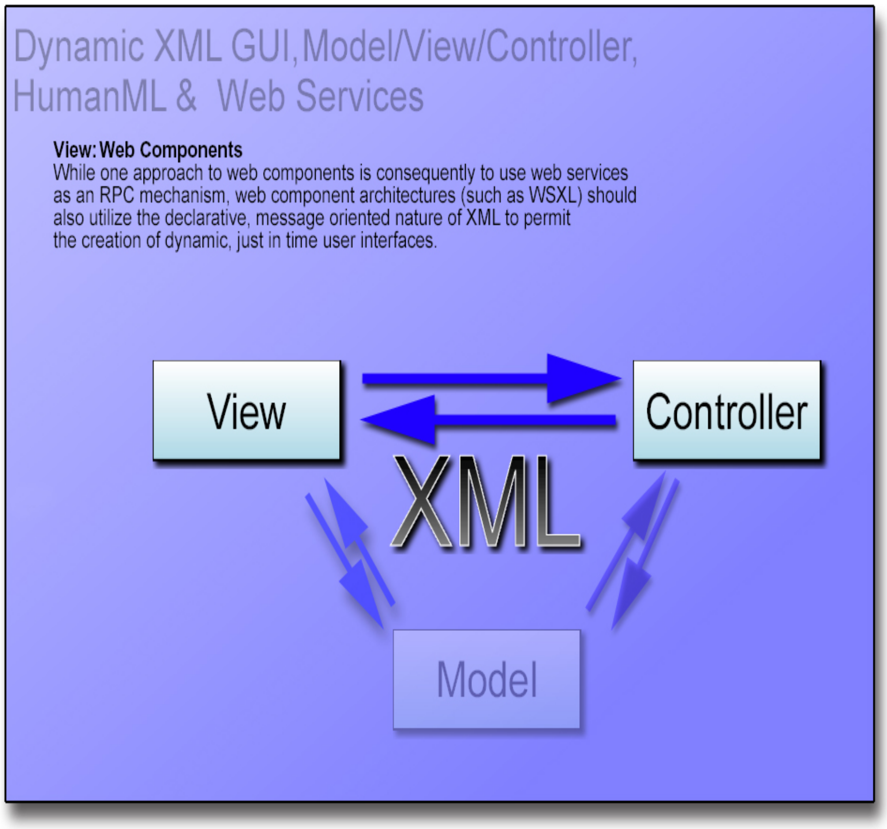
Of course, behavioral tracking also has distinct (and controversial) marketing implications, as does Preference Personalization. This raises the area of commercial possibility of HumanML, especially in conjunction with the rise of consumer web services. The era of mass marketing is over. For better or worse, marketing has become targeted at the individual level, and in order for consumer web services to become viable, a single, consistant standard for creating personalization is necessary. HumanML is intended to be that standard.

The flip side of such profiling, however, is a loss of privacy. Currently there is a battle going on between marketer and consumer that in the end is destructive to both - the marketer has increasingly fine tools at his beck and call to try to interest people to buy his product, but the consumer is also increasingly savvy in rejecting the often heavy-handed efforts that the marketer applies. An open standard of profiling, can change that equation, letting a person customize his or her profile to better fit her needs and requirements. By owning the profile (and with standard sets of tools opening up all kinds of applications for both reading and modifying that profile) it lets the consumer shape her presence in the increasingly wired marketplace.

Again, the benefit to Web Services and Web Component architectures of HumanML are evident here. The more common the representation of a person through a single common schema, the wider the potential audience is for marketers working with that schema. At the same time, common profiles make it possible for consumers to act collectively as well, bargaining for lot purchases that can then be distributed between members. This means that web services providers (which will likely in the end be the distributors of general goods and services, rather than software vendors) can concentrate on building services for selling their products rather than on attempting to establish a proprietary "marketshare" standard to lock people into.

Similarly, the Preference Personalization in the consumer sphere makes it possible to provide "skin" interfaces that not only provide additional services to consumers but also permit a basic level of branding. Currently web advertising is held in some disrepute because most web advertising is intrusive rather than constructive. If, instead, the application *was* the advertisement, then advertising can move into a more symbiotic role in web applications, with the application essentially being *sponsored* by the marketing agency. People would vie to get the IBM Astronaut interfaces or the Intel Alien interfaces or the Lord of the Ring system shells, configured to their specific needs transparently because of the existence of the HumanML profile language.

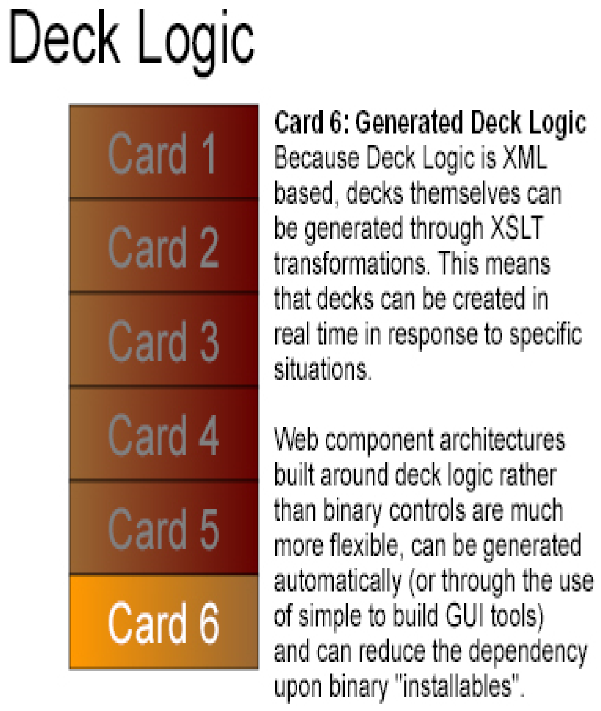
1.2. Integrating HumanML with Web Component Architectures


Dynamic XML GUI, Model/View/Controller, HumanML & Web Services

View: Web Components
While one approach to web components is consequently to use web services as an RPC mechanism, web component architectures (such as WSXL) should also utilize the declarative, message oriented nature of XML to permit the creation of dynamic, just in time user interfaces.

The proposals set out by the OASIS Web Services Component Model Technical Committee describe a standardized web based application architecture that employs primary W3C (and to a secondary extent OASIS) technologies for designing and implementing universal web components. The architecture, as laid in the Web Services Experience Language (WSXL) proposal and the Web Services User Interface (WSUI) proposal provide a means of creating systems based primarily upon distributed applications communicating via web services (presumably using SOAP based messaging systems).

One of the central characteristics of the WSCM proposals is the strong role that Deck Logic plays (this author's term, not the WSCMs). **Deck Logic** describes the use of a declarative document that contains the various possible states that a given node of a system can be in. State changes cause movement along various cards of this deck, though additional state information can (potentially) be maintained in an external variable cache (most likely as another XML entity). The state transformations can be handled via a direct procedural mechanism (such as the XML DOM), but may also be induced by parametrically driven XSLT (as outlined in the WSUI proposal). XForms and WML provide two examples of such Deck Logic entities - each **card** in a WML **deck** or form (called an **instance**) in an XForms **model** provides one specific state that a given node can be in. By creating such Hypercard-like structures, application "wizards" can be constructed declaratively, and can then use transformation mechanisms (such as XSLT) to perform the actual state transitions.


Deck Logic
Card 1
Card 2
Card 3
Card 4
Card 5
Card 6

Card 6: Generated Deck Logic
Because Deck Logic is XML based, decks themselves can be generated through XSLT transformations. This means that decks can be created in real time in response to specific situations.

Web component architectures built around deck logic rather than binary controls are much more flexible, can be generated automatically (or through the use of simple to build GUI tools) and can reduce the dependency upon binary "installables".

HumanML can work effectively in a number of ways within both WSUI and WSXL structures. First of all, HumanML can act as a specific descriptive schema for describing human-centric information within form structures. In this respect HumanML is much like any other schema, to be used primarily by the person creating the application, but the consistent use of a HumanML schema within the WSCM frameworks makes cross adoption of both WSCM and HumanML standards more likely.

However, another compelling use of HumanML is as a means of encoding user interactions (Habitual Personalization) that could in turn be used to generate both WSXL and WSUI declarative structures. In essence, encoding the various human interactions with the system provides a way of creating **intentional programming**; this in turn can make such systems far more responsive to users who are not themselves programmers.

In addition to deliberate actions, such systems could also be designed to work with real time analysis and response to crisis or security situations, as well as the generation of simulations. By encoding specific human actions or interactions, such can be used as events that would cause state changes, regardless of whether such actions come from live data or are artificially generated.

This in turn leads to the use of HumanML as a means for defining actions and intents of avatars, either conscious or programmed. An **Avatar** in HumanML usage is a representation of a (conceivably) intelligent entity. An avatar may be a representation of a human being to the internal system, or it could be a programmatic entity that works via a state engine mechanism. Note that an avatar here does not necessarily need to depict a human being, and certainly does not have to be a realtime 3D rendering of a human being as the more common usage of the word would connote.

The significant aspect of Avatars in the context of this paper is that they actually work quite well using the deck logic formalisms that WSCM promotes. Moreover, such Avatars serve the role of interfaces either into the computer (an avatar being essentially an interface that appears to have "human-like" thought processes) or from the computer to a human being. The integration of HumanML with WSCM makes it possible to provide a consistent set of programmatic interfaces (probably via web service messaging or RPC calls) that people can use to implement either full-blown avatars or at least minimally compliant subsets.

The role of HumanML in preferential personalization was discussed in the previous section, but a word should be made about specific integration issues. User interfaces as they exist currently still employ a model or metaphor that was developed thirty years ago - namely, the desktop metaphor, with its accompanying set of windows, scroll bars, rounded rectangle buttons, icons and so forth. Given the fairly limited computational and memory requirements that existed for much of that period, the creation of a proprietary, essentially static framework and architecture made a great deal of sense. However, it has also conditioned people into thinking that working with computers involved "learning the interface".

Declarative Web Component architectures are poised to radically reshape this notion. The web until recently has been declarative but (principally) static; you were publishing pages of content, not applications. As more computing moves into the browser (or as more of the browser spills into the "desktop"), these traditional GUI interfaces break down. We will borrow some metaphors from what we know, but increasingly the interface will involve a negotiation between the content provider and the content consumer. Previously the negotiation was between the content developer and the content purchaser (which is not quite the same relationship); the purchaser would specify his requirements and the developer would create the interfaces to best facilitate these requirements.

This relationship is changing. A given content provider is as likely to mediate streams of information that may come from other providers as to serve raw content. Web services enforce anonymity of both information and implementation, as well as the decoupling of interface from the data model of that information. If company X and company Y both provide a consistent set of web services, then the primary differentiating factor between such products becomes price (a situation that no vendor wants to be in).

Unless ... unless graphical (sensory) user interfaces comes into play. This is the realm of the content consumer. The strength of your user interfaces will largely dictate the success of your data streams. This in turn means that the negotiation moves down the pipe to the connection between the data provider and the data consumer. That negotiation will in turn be predicated by the specific needs and requirements of that consumer -- if I better negotiate those needs than you, then my data streams (and hence my products or services) will be utilized more than yours will. The purpose of HumanML in this regard is to act as the intermediary for negotiating these needs, and providing realistic clues or indications to the negotiating software as to what those needs are.

These can be translated another way, as well. As indicated previously, one of the principle purposes of dynamic graphical user interfaces is as a branding mechanism. Game designers will tell you that a game may be visually stunning, have profound music, and still be an absolute dog in sales ... because the game play itself suffers. The user experience suffers. Such disappointment can be interpreted as the fact that the expectations of the game player were not met. HumanML provides a tool for adapting user interfaces to better reflect the expectations of the users.

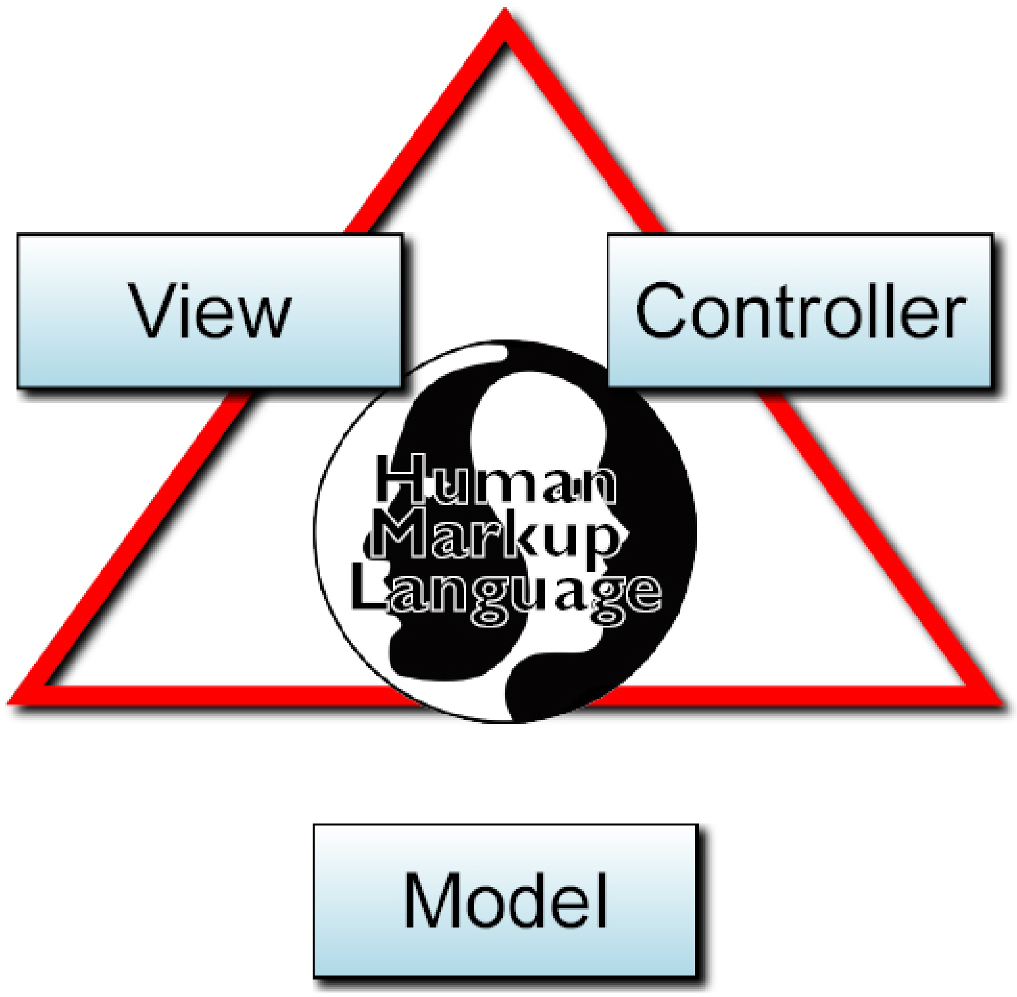
References:

Web Services Experience Language (WSXL) proposal

http://www-106.ibm.com/developerworks/library/ws-wsxl
Web Services User Interface (WSUI)

http://www.wsui.org/doc/20011031/WD-wsui-20011031.html

1.3. The Role of Human Markup in the Semantic Web



The Semantic Web has shifted back and forth in prominance over time, yet as the focus of applications continue to expand outward (from program to computer system to network to client/server Internet to peer-based distributed systems) there are increasing problems with semantic dissonance that has forced web application developers to re-examine many of the issues involved in dealing with complex namespaces (as well as to attempt to better understand the role of computational linguistics in the application development process).

Defining "human" is perhaps one of the more difficult tasks that can be undertaken. Indeed, the process of defining what it means to be human can very easily encompass significant swathes of philosophy, religion, ethics, and morality. It is not the intent of the HumanML team to attempt to encode the works of Aristotle, Aquinas, Kirkegarde, or

Hume in XML - we have neither the patience nor the desire to reduce the human condition to so many less than or greater than signs.

However, one purpose that HumanML does have is to provide a framework upon which to build relevant taxonomies pertaining to human characteristics. As a consequence, HumanML intersects with other XML-oriented technologies such as XML Topic Maps and the Resource Description Framework (RDF). The intent of the HumanML TC is to integrate their efforts with that of other W3C and OASIS taxonomic organizations to insure that there is as little semantic dissonance as possible in the definitions of relevant namespaces. HumanML will be described not only in the XML Schema Description language (XSD) but also RDF Schema to provide associated modelling at the semantic as well as syntactic levels. Additionally, HumanML will work with OASIS Topic Maps Published Subjects Technical Committee (TMPS).

HumanML is being designed to work in a modular extensible fashion. This will not only insure that the language can grow in response to shifting needs and requirements, but that the language can be easily integrated with other namespaces. This resolves (or at least ameliorates) the problem of having to create one schema for doctors, another for use by police, others for use by programmers, etc.

HumanMarkup can provide for applications which will allow text or collections of text that have not been previously annotated to be marked up in ways that can make anecdotal information yield up data not previously available. For example, Court Reports in Child Protective Services can be marked up, even years later, in ways that make the number of times that a certain markup tag is used in a collection of related documents a statistically relevant item of information as numerical data.

Within Objects corresponding to taxonomical categories, the relationally sorted information for individuals in various kinds of contexts can be retrieved. These information bases will also be accessible through Object searches that start from divergent points because HumanMarkup allows its ontologies and taxonomies to overlap as needed to make specific kinds of information available in a variety of contexts.

For more comprehensive information, see

References:

Human Markup Language Frameworks.txt

http://www.oasis-open.org/committees/humanmarkup/documents/HM.frameworks.txt
Human Markup Language Applications.txt

http://www.oasis-open.org/committees/humanmarkup/documents/HM.applications.txt

1.4. Goals and Intentions

HumanML is a long term project. Given the organizational difficulties of producing something like HumanML, a language for which everyone has their own (invariably unique) needs, HumanML will likely be an evolving standard for some time to come.

However, HumanML is ultimately concerned with the process of communication. In essence it defines a framework for establishing communication protocols between people, or between people and systems. As such, it is the intent of the Human Markup Language Technical Committee to insure that the language can both be cleanly integrated into the Web Services Component Model and that the WSCM provides an adequate level of support for handling the mission and goals of HumanML.

Some requirements that HumanML does have (a wish list, admittedly, but one that may benefit the WSCM TC as well) include the need for adequate authentication within transactions, and an error handling mechanism sufficiently robust to handle the encoding of semantic as well as syntactic invalidation, in both synchronous RPC and asynchronous messaging environments. Additionally, some mechanism should be possible (whether at the layer of WSCM on the protocol stack or via an application layer above WSCM) that would permit semantic validation (through an RDF Schema validator, for instance), as many of the applications for HumanML work at the semantic rather than the syntactic level.

1.5. Conclusion and Summary

The Human Markup Language addresses a number of issues that are involved with the programmatic development process, regardless of whether the process is carried out within a stand-alone application or over hundreds of thousands of systems. HumanML provides a mechanism for personalization of both content and visual interface. It offers a common standard for encoding human-centric information. The language can be utilized as a spring board for the creation of specialized schemas, and it can also be used to encode complex descriptive state for avatars and simulations, an increasingly frequent requirement as computers begin to exceed the threshhold of real-time rendering in both two and three dimensions.

As web applications become more diffuse, the ability to persist a cohesive standard for description of avatars and avatar services makes Human Markup Language a good fit with the Web Services Component Model. It is the hope of the HumanML group that this work can be done in conjunction with WSCM, and that the fruits of such collaboration will be beneficial to both groups and to humanity at large.

Appendix A

Examples and Illustrations

During Phase 0 of the HumanMarkup Initiative, a simple diagram was developed to show what HumanMarkup could do and where in the computing process it would work. This is the Interpretative Matrix, called such because what HumanMarkup does is to add a layer or level of interpretation to the process. How that interpretation is carried out will depend on the middleware application which uses it.

To view the entire series of three pages which show how the interpretation process was envisioned to work and which remains the model for now, visit:

http://groups.yahoo.com/group/ humanmarkup/files/Conceptual/ Interpretative.Matrix/ humanMLIntrprtMatrix-1.jpg
http://groups.yahoo.com/group/ humanmarkup/files/Conceptual/ Interpretative.Matrix/ humanMLIntrprtMatrix-2.jpg
http://groups.yahoo.com/group/ humanmarkup/files/Conceptual/ Interpretative.Matrix/ humanMLIntrprtMatrix-3.jpg
In terms of Web Services Component Model work, HumanMarkup will be concerned largely with Data Service, adding new kinds of data, as well as adding related data resources to enhance the range of information that can be utilized in personalization. HumanMarkup will also have a great deal to do with Presentation Service due to the enhanced personalization for which it will provide.

(Insert Model-View-Controller Diagram with Interpretative Matrix contained within it.)

To be more specific, the following diagram illustrates how HumanMarkup would work within one of the examples offered in the WSXL paper from IBM.

Lastly, to show a more current state of the art example of significance for HumanMarkup and for Web Services, I offer Voice Activated Humanoid Animation. It should be noted that this is a prototype, executed by a student at the Naval Postgraduate School in Monterey, California recently.

http://www.web3D.org/TaskGroups/x3d/translation/examples/HumanoidAnimation/VoiceActivated/NancyVoiceActivated.wrl
It is recommended that the Cortona Plug-in be installed to view Web3D samples:

http://www.parallelgraphics.com/products/cortona/
Ozan Opaydin has created a voice-activated interface for the sample Humanoid Animation 1.1 Specification for his work at the Naval Postgraduate School lab. Voice user interface (VUI) demo movie (4 MB, Windows media only for now) at

http://www.web3D.org/TaskGroups/x3d/translation/examples/HumanoidAnimation/VoiceActivated/NancyVUI.wmv
Voice Activated File:

http://www.web3D.org/TaskGroups/x3d/translation/examples/HumanoidAnimation/VoiceActivated/NancyVoiceActivated.wrl
This is Java code that uses the javax.speech API. Ozan and Don Brutzman, who provided this sample to the H-Anim Working Group Mailing List of the Web 3D

ConsortiumDecember 21, 2001.have tested it using the IBM ViaVoice product (~ $99) and also the free runtime bindings.

Setup instructions for IBM ViaVoice

athttp://www.web3D.org/TaskGroups/x3d/translation/examples/HumanoidAnimation/VoiceActivated/README.html
Batch files compile.bat and run.bat for easy execution under windows.