larity metric, the dynamic (motion-flow) content and other features. Moreover, use of personalized criteria is made possible by the fact that very little pre-computation is needed. Preliminary objective and subjective evaluations of basic principles show that storyboards produced on the fly can be of high quality.

Searching activity: the user may encounter a frame/scene during video-browsing that is of particular interest (eg a city skyline) and be curious as to whether other videos contain a similar scene/frame (eg a different take of the same city skyline). At this point the user could extract new and unexpected knowledge from a comparison of different videos having this key common scene/frame.

Personalization of browsing and searching: given a video that is of interest for a user, browsing involves being able to quickly assess the most interesting (relevant, typical or unusual) frames/scenes in the video in order to decide whether it is worth watching in its entirety (or which portions are worth viewing). Users should be allowed to select at view-time basic parameters such as storyboard length and waiting time. However more advanced summarization criteria need to be supported. For example users must be allowed to specify the dynamic (motion-flow) content they are interested in (eg a moving car is different from a parked one). Further, the underlying similarity metric used for selecting the representative frames should be biased using implicit or explicit user requirements.

Videos can be considered to be sequences of still images (with sound), and techniques for handling large collections of still images (pictures) might be the first line of attack. However, we do not consider this approach to be suitable. A single video corresponds to thousands/millions of frames (depending on its duration) that, with the exception of scene changes, are locally highly similar to one another. Thus the sheer volume of data poses scalability problems to approaches based on indexing single frames (or a dense blind sample of them) as single still images. Instead, in the VISTO project, we intend to develop video representations and indexing techniques that take dynamic components of the video data as prominent in the representation. The aim is to simultaneously exploit the spatial and temporal coherence of video data, not only to attain compression, but also to make searching fast and efficient. Attaining both goals simultaneously is challenging.

Future activities will involve setting up a complete network-aware software/hardware/conceptual architecture able to cope with high throughput demands in a P2P style of computation, taking advantage of IOF architectural designs. Awareness of network capabilities, which change over time, is particularly important for use on, for example, wireless hand-held devices.

The VISTO project started in 2007 at IIT-CNR as a collaboration between researchers of the Institute for Informatics and Telematics, CNR, Pisa, the University of Modena and Reggio Emilia, and the University of Piemonte Orientale.

Link:
http://visto.iit.cnr.it/

Please contact:
Marco Pellegrini
IIT – CNR, Italy
E-mail: marco.pellegrini@iit.cnr.it

The New Role of Humans in the Future Internet

by Daniel Schall and Schahram Dustdar

In most cases, service-oriented architecture is realized using Web services technology. At the Vienna University of Technology, we have implemented a platform enabling humans to provide services. We foresee important applications that will be based on Human-Provided Services and software services.

Service-oriented Architectures (SOA) are typically comprised of software services. Many collaboration and composition scenarios involve interaction between human actors as well as software services. Current tools and platforms offer limited support for human interactions in SOA: we therefore present Human-Provided Services (HPS) and the HPS framework. In particular, the aim of the HPS framework is to:

- offer a service registry maintaining information related to human and software services
- enhance service-related information by describing human characteristics and capabilities
- define interaction patterns using Web services technology so that human actors can efficiently deal with interactions.

HPSs are offered by human actors. Web services technology is used to describe HPSs and to enable interaction with real people. The advantage of HPS is that these services can be used in different compositions and Web-based collaborations.

A possible scenario of human and software services is shown in Figure 1a. A human can define an activity, eg 'review document activity', which is transformed into a Web service interface. Such interfaces are typically described using the Web Services Description Language (WSDL). The same standards can be used to describe HPSs and software (Web) services.

Via a graphical user interface - a Web 2.0 portal hosted by the HPS framework - the end user can create HPSs without having to understand XML or Web services technology.

The role of humans in the future Internet and SOA is thus not limited to consuming services (Figure 1b); services can also be provided by human actors. HPS unifies humans and services, because a service can be provided by a

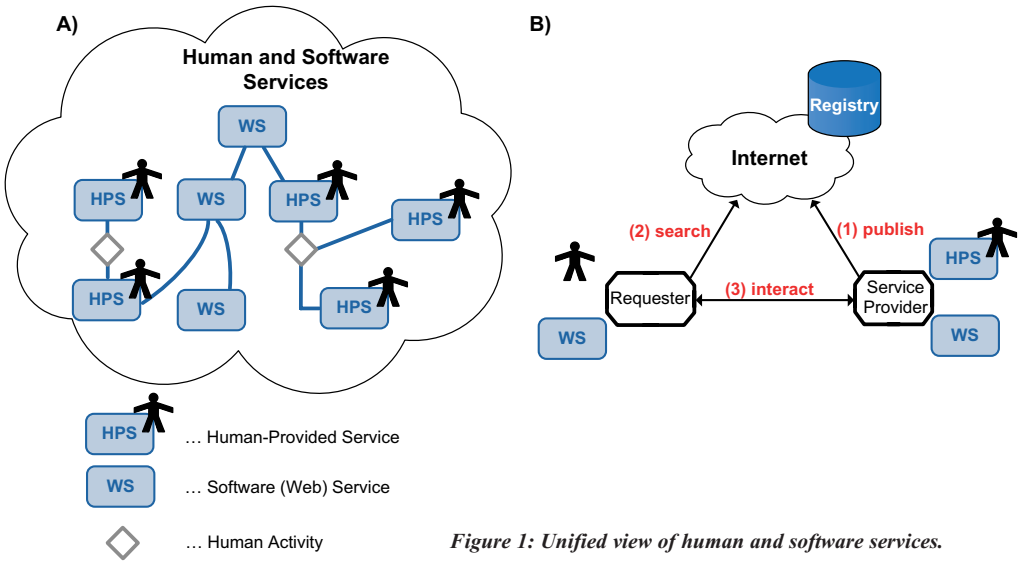

Figure 1: Unified view of human and software services.

human actor or implemented as a software service. Three steps are performed when using HPS:

1) Publish: the user can create an HPS and publish the service on the Web using a registry. Publishing a service is as simple as posting a blog entry on the Web. It is the association of the user's profile with an activity depicted as a service.

2) Search: the requester can perform a keyword-based search to find HPSs. Ranking is performed to find the most relevant HPS based on, for example, the expertise of the user providing the service.

3) Interact: the framework supports automatic user interface generation. HPS can be used in interactions between humans and in interactions between software services and HPSs.

The HPS framework is comprised of three layers (Figure 2). The first layer contains data collections to maintain user profile information, service descriptions (WSDL and related information) and interaction rules. These rules can be created by the user to define event-condition-action patterns such as pre-filtering of interactions, messages etc.

The second layer contains user tools including a graphical user interface for designing services (a simple tool to create HPSs without programming code), the activity management to organize and structure activities, interactions and related messages. The service management allows the user to modify his/her HPSs.

The third layer enables interaction between requesters and HPSs. The lookup is used to find suitable services, with matching HPSs being ranked and returned (comparable to using search engines on the Web). The access layer is a proxy service dispatching messages that are exchanged in HPS-based interactions.

The middleware services are implemented in Java/J2EE and data collections are managed with XML databases (to manage WSDL- and XML-type definitions) and MySQL to manage interactions and related messages.

The user tools are mainly implemented in C# and ASP.NET with the support of automatic user interface generation using XForm (XML forms) technology. These forms are generated based on XML descriptions (eg WSDL).

The framework offers APIs (lookup, registry), enabling integration with other platforms.

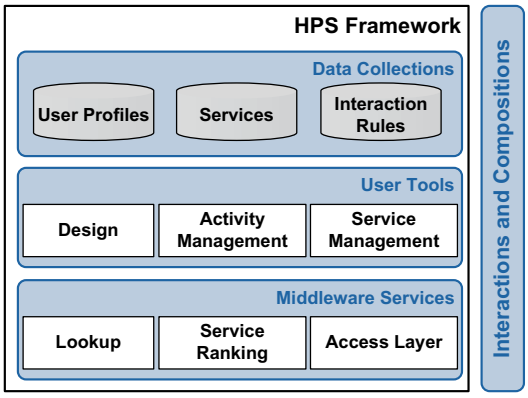

Figure 2: HPS framework.

Human-Provided Services

- Document Review/Translatio
- J2EE Consultant
- Java Expert
- Lawyer

Compositions Human and Software Services:

- Task Force
- Expert Team
- Service/Social Community
- Human Computation

Link:
http://vitalab.tuwien.ac.at/autocompwiki/index.php/Human-provided_Services

Please contact:
Daniel Schall
Vienna University of Technology / AARIT, Austria
Tel: +43 1 58801 18453
E-mail: schall@infosys.tuwien.ac.at
Schahram Dustdar
Vienna University of Technology / AARIT, Austria
Tel: +43 1 58801 18414
E-mail: dustdar@infosys.tuwien.ac.at