The On-TIME project

Tiziana Catarci[1], Alan Dix[2], Raffaele Giuliano, Marco Piva, Antonella Poggi[1], Fabio Terella, Emanuele Tracanna

[1]SAPIENZA Università di Roma, Italy
{catarci, poggi}@dis.uniroma1.it
[2]Lancaster University, United Kingdom
alan@hcibook.com

Abstract. The On-TIME project was born as follow-up of a DELOS task within the User Interface and Visualization workpackage. Its general aim is to provide the user with a system that allows her to focus on tasks rather than just on managing her personal information, as in traditional personal information management systems. To achieve this goal, user data and tasks are described in terms of explicit semantics that the user can share, i.e., by means of a Personal Ontology, reflecting the user's view of the world and her personal interests. In this paper, we describe the main current achievements of the on-going On-TIME project, namely the actual architecture and the effective user interface of a first working prototype.

Keywords: Personal Information Management, Task Management, User Interface.

1 Introduction

Today's personal desktops have become from far the most commonly/frequently used personal digital library. However, personal desktops mostly consist of a disconnected set of generic tools, that the user interacts with, often by manually repeating similar tasks and copying for this the same data from one application to the other several times.

This has motivated the arising of a specific research task within the DELOS Network Of Excellence, more precisely within the User Interface and Visualization workpackage. One of the outcomes of this task was the definition of a new paradigm of system, called Personal Interaction Management System (PIMS) [4], that allows the user to focus on the tasks they have to perform rather than just managing their personal information. The main feature of a PIMS is to rely on the use of a Personal Ontology reflecting the user's view of the world and her personal interests, to describe both user data and tasks in terms of explicit semantics that the user can share. This represents a challenge since tasks need to be explicitly represented both in their static aspects, i.e. the kind of information that they manipulate, the kind of programs involved in these manipulations, security and authentication issues that may arise, and in their dynamic ones, i.e. the sequences of actions that they require, the alternative

choices that are given to the user, pre-conditions, post-conditions, and invariants for the various actions that are involved in the task.

Other outcomes of the abovementioned DELOS task were (i) the definition of a "core" Personal Ontology to be further extended depending on the user stereotype and (ii) the design of a preliminary architecture based on the use of an existing web bookmarking tool, called Snip!t [5], whose distinguishing feature is to "intelligently" suggest appropriate actions to perform starting from a Web page content.

The On-TIME project was born as follow-up of the DELOS task. Specifically, On-TIME carried out an actual refinement of the PIMS architecture and led to a first working PIMS prototype [3], implementing and integrating all DELOS task outcomes. The ultimate goal of the On-TIME project is to provide a system that effectively supports the user while performing day-to-day tasks. This requires in particular to (i) design an effective user interface, (ii) be able to learn most common tasks, to infer what is the next task to be performed, and finally to execute (semi-automatically) tasks by linking them to user's data.

In this paper, we present the main current achievements of the On-TIME project. In particular, in Section 2, while describing the On-TIME architecture, we provide some insights on the main features of the system. In Section 3, we illustrate the On-TIME user interface. Then, we conclude the papers by discussing the main project open issues.

2 On-TIME architecture

In this section, we describe the On-TIME architecture. Specifically, we discuss each of the main modules of the system illustrated in Fig.1. Note that two traditional relational repositories of personal data and tasks definitions are used, named *onto* and *onto_task* respectively, that are uniquely accessed through dedicated modules. The main modules of the user interface, represented in the figure as application-like windows, will be discussed in the next section.

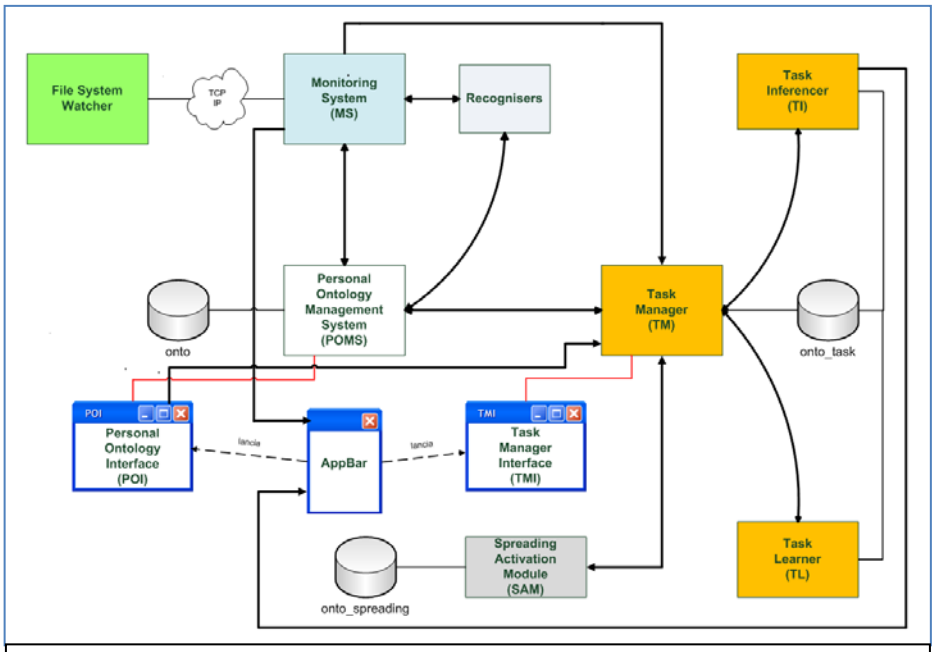

Fig. 1. The On-TIME architecture

File system watcher. This module is responsible for monitoring the file system. More precisely, it can be configured to make it detect events such as the creation, modification, renaming, or removal of files depending on their extension, and/or their location. Furthermore, it can monitor specific email clients, such as Windows Live Mail. For each event that is detected, the file system watcher sends an XML message via a TCP socket, containing both the description of the event and the element of the file system concerned by the event itself.

Monitoring system. As this module receives an XML message from the *File system watcher*, it processes the message and, possibly, uses the *Recognisers* to analyse its content. Then, depending on the event that was detected, it issues an update to the *Personal Ontology Management System*. For instance, if the event was the creation of a new file named On-TIME_Proposal.doc, then it issues the addition of an instance of the concept "File" of the Personal Ontology, with appropriate name, author and date of creation.

Recognisers. *Recognisers* are modules that detect the occurrence of complex types of data, e.g. person names, addresses, based on the use of specific regular expressions, as well as of dictionaries. Then, as a new value is detected, recognisers check whether such value already occurs in the Personal Ontology, and, if not, propose the user to add an instance of an appropriate concept, associated to the value.

Personal ontology management system. The *Personal Ontology Management System* is responsible to maintain the Personal Ontology, and to manage the underlying reasoning services over the Personal Ontology, namely query answering and semantic update/erasure of instances. To this aim, we use the Personal Ontology designed within the DELOS task. We also use the QuOnto system as underlying reasoning engine [2].

Spreading activation module. This module processes instances of the Personal Ontology, and associates to each instance a value that represents its "contextual relevance", also called *activation level*. This value is computed on the basis of studies of the mechanisms of the human memory [6]. For example, instances that have recently been part of the user's activities or are related to them are flagged as "hot". The spreading activation value associated to instances is crucial, e.g. to establish which data should be considered as input for next tasks.

Task manager. The *Task manager* maintains tasks definition, and executes tasks on the basis of the *Task inferencer* output. To this aim, a formal language with a well-defined semantics was devised, based on previous work on the topic[1], that, intuitively, allows to define tasks in terms of pre-conditions and post-conditions over the Personal Ontology.

Task inferencer. This module is responsible to infer what is the task that is more likely to be performed next, on the basis of tasks pre-conditions as well as current data activation levels. For instance, suppose that the task Confirm participation to a conference requires the existence of a forthcoming event related to an on-going project. Then, the *Task inferencer* will propose the user to perform all steps required to organize a mission in the occasion of a specific event, as soon as both the specific event is flagged as "hot", e.g. because the user recently updated the details.

Task Learner. The Task learner is the module in charge of learning tasks definition by monitoring the user interaction with her desktop, taking into account the semantics of personal data handled within the interaction.

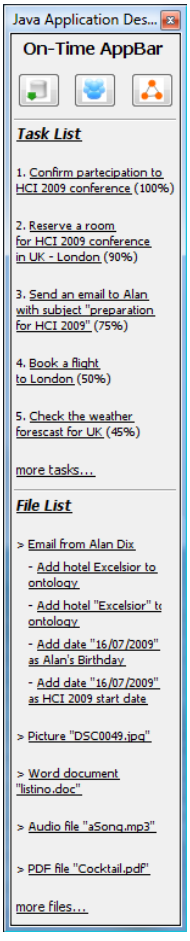

Fig. 2 - AppBar

3 The On-TIME user interface

In this section, we describe the On-TIME user interface. Specifically, we discuss the three main modules of the interface, namely the *AppBar* module, the *Personal Ontology interface* and the *Task manager interface* through the presentation of a typical user interaction with the On-TIME system interface.

Suppose that Antonella works in a research lab that has recently installed a new fax. She needs to update all the documents where she used to indicate the fax number. This can be easily achieved with On-TIME, by simply updating the Personal Ontology. Hence, the starting point of her interaction with On-TIME is the *AppBar* [Fig. 2] installed on her desktop.

The *AppBar* component is the core of the system interface and the first component that appears to the user. It is a side bar (like the widget bar in Windows Vista) that can easily be accessed by the user with the mouse. This bar is divided into three sections. The first section contains a link, represented as a button, for each possible view of the *Personal Ontology Interface* (see below). The second section contains a list of tasks that according to the *Task manager*, are more likely to be performed. The third section contains the list of updates suggested by the *Monitoring system*. More precisely, if candidates instances, or instance attributes, that are not currently in the ontology, were found by the *Monitoring system* within the newly saved documents, the user would be able to update the Personal Ontology associating to those instances the right semantics, e.g., a date could be her best friend's date of birth or an important event date.

The *Personal Ontology Interface* is a component providing three coordinated views[6] over the Personal Ontology, having a common section that is an indented list showing the concepts hierarchy tree. While the common section allows the user to select a specific concept quickly, for example to view its instances in details, the three views have each a different purpose, and are coordinated so that when the user switches from one view to the other, she keeps focusing on the same particular instance/concept. Specifically, the *Structure View* allows the user to investigate the details of the ontology structure, the *Instance View* to select an instance of a specific ontology concept and to visualize and edit its details, and finally the *Navigation View* to browse the ontology through both its instances and concepts. Let us describe the latter in more details. It shows always two graphs, representing respectively connections among instances and concepts. One of the two graphs is always foreground and the other is visible in a frame positioned in the upper right corner of the screen, with the facility of switching between them. In order to optimize instance browsing, we use the focus plus context technique which allows the user to focus her/his attention on a specific instance or concept, by moving it on the centre of the screen, highlighting neighboring nodes (at a certain distance from the focus) and displaying the names of relations connecting them. This approach allows for maintaining the graph visualization as thinner as possible, succeeding in managing huge graphs. Moreover, the user can navigate the graph arbitrarily, and then return to the focused instance, by clicking on an anchor that is always visible.

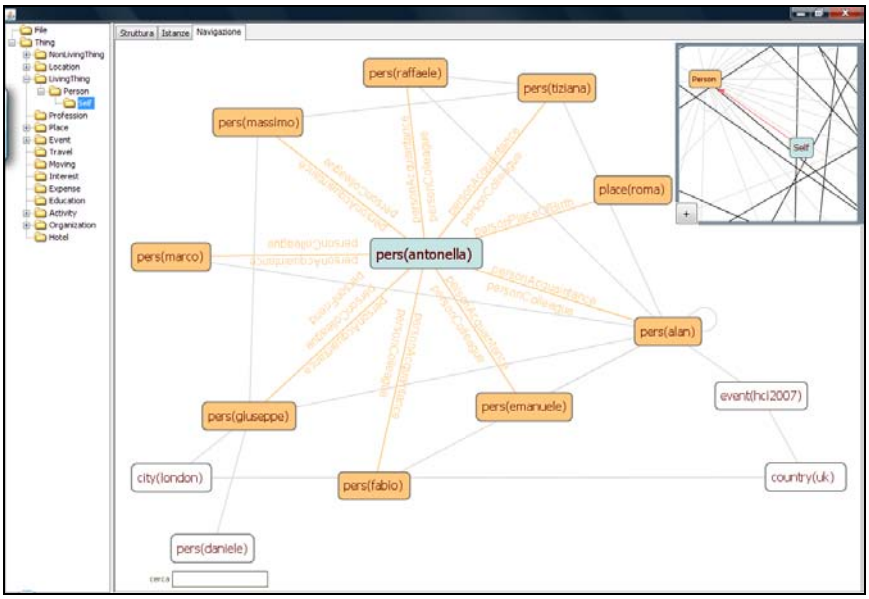

Fig. 3 - Graph view

Coming back to our scenario, by clicking on the *AppBar* appropriate button, Antonella accesses the Navigation View to browse the Personal Ontology. Doing so, the ontology graph appears foreground, centered on the instance pers(antonella) [Fig. 3] denoting herself, (i.e., the system owner). Then, she verifies all relations

connecting her with other ontology instances. In particular, she now switches to the Instance View, that according to the coordinated multiple views paradigm, is focused on pers(antonella). Thus, Antonella immediately accesses the attribute personFax denoting the fax number that she is using, and she updates it. Note that, while saving the changes, the *Personal Ontology management system* ensures that the information currently contained in the Personal Ontology is not contradictory.

Suppose now that Antonella moves her attention back to the *AppBar.* She would then notice that On-TIME suggests the task Confirm participation to HCI2009 conference. Actually, the *Task inferencer* proposes such a task, because Antonella just received an email request of confirmation by the HCI2009 organizers, which was detected by the *Monitoring system*, and made increase the activation level of the instance event(hci2009) denoting the event HCI2009. Antonella takes then advantage of this smart suggestion by the system, and executes the task by clicking on the relative link in the *AppBar* which opens a new window that is part of the *Task manager interface*, i.e. the component that, by means of appropriate wizards, allows the user to execute a task, by interacting with the *Task manager*. In our scenario, the selected task just requires Antonella to confirm the reservation details, automatically returned by the *Personal Ontology management system*. Then, the *Task manager* completes the task execution by updating the ontology with the information that Antonella is going to participate to the HCI2009 conference. Specifically, this will be achieved by adding the couple of instances (event(hci2009), pers(antonella)) to the relation personConference existing between the concepts Person and Events.

4 Conclusions

We presented the on-going On-TIME project. In particular, we described the main results achieved so far within the project that led to the first working prototype of a Personal Interaction Management System. Special emphasis was given to the On-TIME user interface, that, besides providing an effective access to the On-TIME system, also represents the first interface allowing the user both to browse and update an ontology, based on its semantics instead of its syntax.

Several issues need to be addressed to reach the ultimate goal of the On-TIME project. Concerning the interface, more tests need to be performed. Also, the interface should be personalized, and personalization should be as much as possible automatic making the Personal Ontology play the role of user profile. Finally, we want to investigate how to visualize more general ontologies, e.g., where an instance may belong to distinct concepts that need not to be in a hierarchy.

Another challenging feature is the design of task learning algorithms and the design of a bunch of personal ontologies, aiming at covering actual users stereotypes.

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable Description Logics for Ontologies. In: 20th Nat. Conf. on Artificial Intelligence AAAI'05 (2005)
2. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and Programs over Description Logic Ontologies. In: DL workshop (2007)
3. Catarci, T., Giuliano, R., Piva, M., Poggi, A., Terella, F: The On-TIME User Interface. In: CHItaly. Rome (2009)
4. Catarci, T., Dix, A., Katifori, A., Lepouras, G., Poggi, A.: Task-Centered Information Management. In: First International DELOS Conference, Pisa, Italy, February 13-14, 2007, Revised Selected Papers, C. Thanos, F. Borri and L. Candela (eds.). LNCS 4877, pp. 197-206. Springer (2007)
5. Dix A., Beale R. and Wood A. 2000. Architectures to make Simple Visualisations using Simple Systems. In Proceedings of Advanced Visual Interfaces (AVI2000), pp. 51--60. ACM Press (2000)
6. Dix, A., Katifori, A., Lepouras, G., Vassilakis, C.: Spreading Activation Over Ontologies: From Personal Context To Web Scale Reasoning. (Submitted for publication) (2009)
7. North, C. and Shneiderman, B.: A taxonomy of multiple window coordinations. Technical Report #CS-TR-3854, University of Maryland, College Park, Dept of Computer Science (1997)