# Visual Support for Work Assignment in YAWL

Francesco Cardi[1], Massimiliano de Leoni[1], Michael Adams[2], Arthur H.M. ter Hofstede[2], and Wil M.P. van der Aalst[2,3]

[1] SAPIENZA - Università di Roma, Rome, Italy
fcardi82@gmail.com, deleoni@dis.uniroma1.it
[2] Queensland University of Technology, Brisbane, Australia
mj.adams@qut.edu.au, a.terhofstede@qut.edu.au
[3] Eindhoven University of Technology, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tue.nl

## 1 Introduction

A basic function of Process-aware Information Systems (PAISs) is to offer work to resources. The elementary pieces of work are called *work items*, e.g. "Approve travel request XYZ1234". Work items are offered to users via so-called *work-list handlers*, which take care of work distribution and authorisation issues.

Typically, PAISs use a so-called "pull mechanism", i.e. work is offered to all resources that qualify and one resource selects it for execution. Even though resources are free to select any work items they qualify, they should pick the right work items in the right order. The right order is thus balancing resources' personal needs and the overall benefit of their respective organisations. In order to drive to the right choice, basic information is provided, e.g. task name, due date, etc. To the best of our knowledge (see [1]), commercial and open source PAISs present work lists simply as a list of work items, each with a short textual description. Context-aware systems provide contextual information, but the different viewpoints are not merged together.[4]

This paper aims at describing the operationalisation of an innovative worklist handler that overcomes this limitation. This worklist handler has been implemented as a component of the YAWL Process Management System[5], although it has been conceived as being independent to any particular PAIS. This worklist handler provides participants with a more detailed insight of the context in which processes are carried out. In addition, it aims to provide support for work item selection by taking into account the circumstances and attributes of all of the users currently active.

In order to achieve these results, we have introduced the metaphor of *maps*. A map can be a geographical one (e.g. a map of a university campus), but may also be a process schema, an organisational diagram, a Gantt chart, etc. Work items are then visualised as dots on these maps. By allowing the choice of the map to be configurable, different types of relationships can be shown, thus providing a deeper insight into the context of the work to be performed.

Resources may also be shown on maps, e.g. by using their position. Besides the "map metaphor" we also use the "distance metaphor"— from the viewpoint of the user, some work items are close while others are far away. This distance may be geographic, e.g. a field service engineer may be far away from a malfunctioning printer

[4] E.g. Taskmind - http://www.taskmind.net/en/community/.
[5] The YAWL Web Site - http://www.yawlfoundation.org.

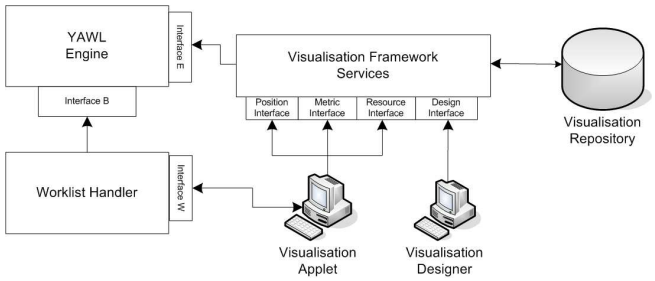



**Fig. 1.** The reference architecture

at the other side of the campus. However, many other distance metrics are possible. For example, one can support metrics capturing familiarity with certain types of work, levels of urgency, and organisational distance. It should be noted that *the choice of metric is orthogonal to the choice of map* thus providing a high degree of flexibility in context visualisation. Resources could, for example, opt to see a geographical map where work items display their level of urgency and their positions are calculated based on a function supplied at design time. The complete description of the general framework can be found in [1].

## 2 Implementation

This innovative worklist handler has been concretely implemented as a Java Applet. This choice is motivated by the fact that we aim to integrate it with YAWL's standard worklist handler, referred to as the *Resource Service*, which is a web application. By using the Resource Service, users can access their work queues to view their offered, allocated, started and suspended work items. The visual worklist handler provides an overlay to the Resource Service, so that users have the option to view a worklist as a traditional queue or in the form of a map of their choice.

Figure 1 shows the overall architecture of the implementation. The Visualisation Applet connects to *Visualisation Framework Services*, which is a collective name for a number of modules that provide information regarding the depiction of maps and the placement of work items (e.g. URLs to locate map images, work item positions on various maps, work item metrics). The framework services use information that is stored partly in a database, the *Visualisation Repository*, and partly obtained by querying the YAWL Engine and the Resource Service through *interfaces E* and *W*. Interface E is used to acquire logging information, which is needed to compute the values for certain metrics. Interface W allows the applet to obtain information about work items (e.g. their status) and users (e.g. work items assigned or offered to a given participant), as well as about running process instances (e.g. variable values).

Finally, the implementation provides a *Visualisation Designer*, which is used by administrators to define maps as well as to specify the policies for positioning work items. For instance, the Visualisation Designer allows dragging and dropping of tasks of process specifications onto maps so as to determine work item positions. The Designer then stores the proper data in the Visualisation Repository for later retrieval. The positioning

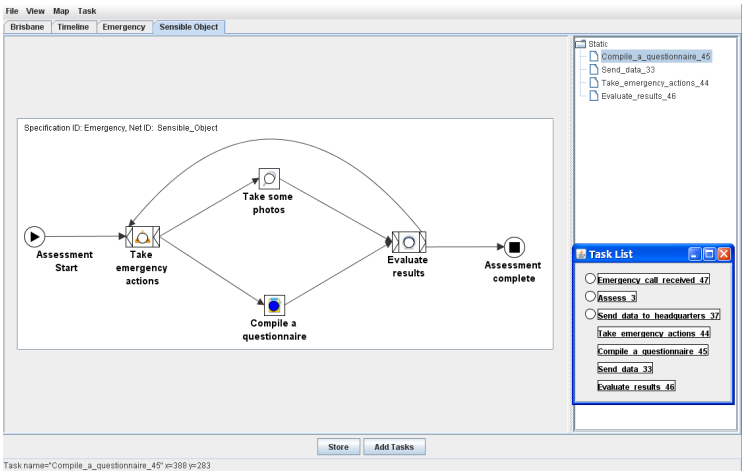
of work items on maps can be statically defined at design-time or determined at run-time according to certain policies. In the latter case the Visualisation Designer allows one to specify at design-time an XQuery expression over case variables for given pairs $< T, M >$ where $T$ and $M$ are respectively a task and a map. When the applet has to position a certain work item of task $T$ on map $M$, it performs the XQuery over the case variables involved, which returns the actual work item position on $M$.

While the implementation is targeted at YAWL, it remains generic, i.e. it is relatively easy to embed the visualisation framework in other PAISs that make available interfaces providing the information required.

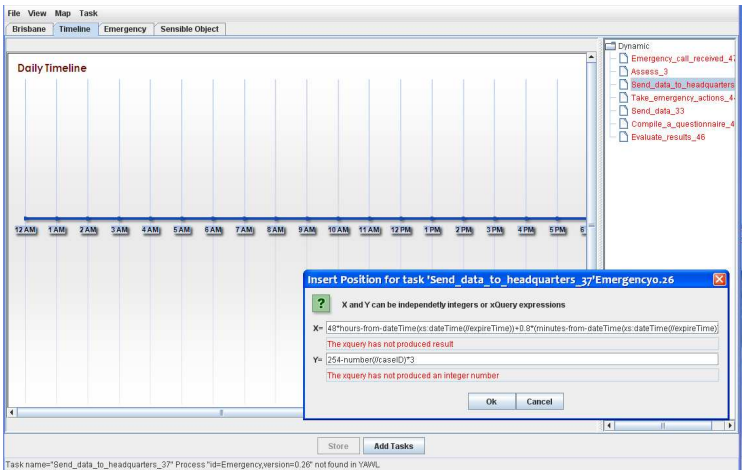
## 3 A Sample Use Case

In this section we briefly describe a sample use case of the new worklist handler which concerns emergency management. Let us assume an earthquake has occurred in the city of Brisbane (Australia) and several civil-protection teams are sent to the area to handle the emergency. For each location of interest a process needs to be followed, which can be generalised as the execution of four activities. The first is to take some emergency remedial actions to aid the population. Then, an emergency worker takes photos of the wounded, while another fills in a questionnaire about the situation. Finally, an emergency worker collects the photos and the questionnaire and contacts headquarters in order to obtain further instructions. According to the feedback, the activities for that area may need to be repeated.[6] Figure 2 shows two screenshots of the Visualisation Designer. In these screenshots, tabs are used to show the defined maps. Each tab allows users to specify the work item positioning for the corresponding map. In order to position some tasks of a certain process on a certain map, the designer first parses the YAWL process specification. Then, the *Task List* window appears, which contains a list of all tasks in the process. Figure 2(a) shows how to position tasks statically. The designer chooses the proper map and then drags and drops tasks from the *Task List* window onto the map. Figure 2(b) shows the dynamic positioning of work items. Instead of dragging and dropping, users can double click to open a new window where an XQuery expression can be specified to compute the $x$ and $y$ coordinates. If applicable, the window allows the designer to preview the values resulting from the evaluation of the XQuery applied to a randomly chosen case. Figure 3 shows two screenshots of the applet of the worklist handler. Specifically, Figure 3(a) illustrates a geographic map. The small dots correspond to work items positioned on the map. They are coloured according to the currently selected metric, in this example the Geographic Distance. This metric measures the proximity of a user to the locations where work items should be executed. A design decision was made that for offered work items (i.e. work items not yet assigned to a participant) a colour close to white represents a low metric value whereas a colour close to red represents a high metric value. In addition, according to the general framework described in [1], there are special colours denoting work item status (e.g. purple means 'allocated to specific participants'). When dots overlap, they are joined to form bigger dots, since otherwise some of them would be invisible. The diameters of such dots grow logarithmically with the number of work items amalgamated, and their colour is determined by the colours of the underlying dots. The applet also shows the locations

[6] Two videos that demonstrate the example can be found at www.yawlfoundation.org/videos.



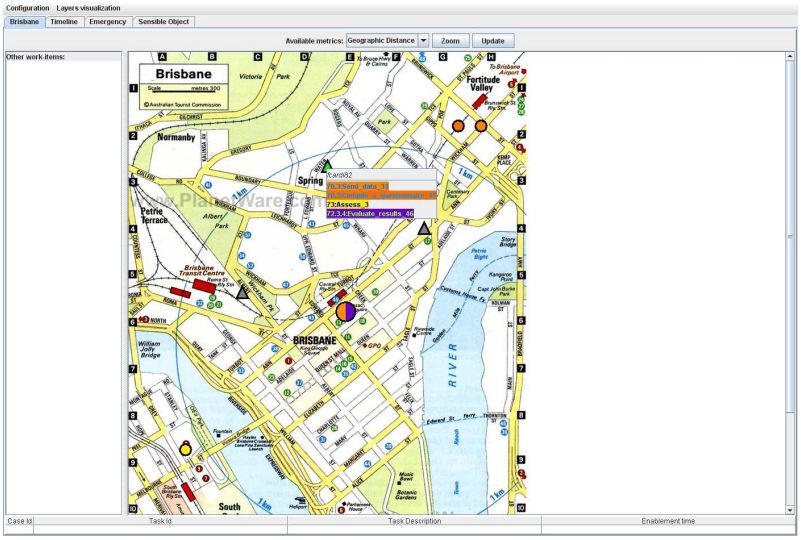

(a) Definition of a static positioning on a "process specification" map



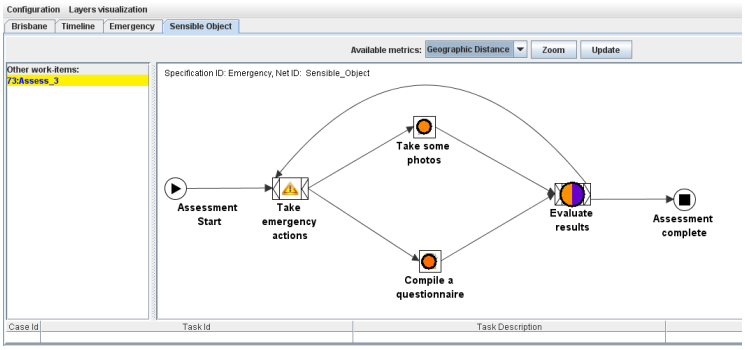

(b) Definition of a dynamic positioning on a time-line map

**Fig. 2.** Screenshots of the Visualisation Designer

of participants on maps in the form of triangles. One green triangle denotes the position of the participant who is logged on through that running instance of the applet. The other gray triangles show the positions of other participants. In line with the authorisations specified, clicking on a participant leads to a pop-up showing the list of work items that this participant would see in their applet. The applet is also organised using tabs, one for each map. Figure 3(b) depicts the "process definition" map where work items are located on top of the corresponding tasks. Work items on the left-hand side




(a) A “geographic” map




(b) A “process specification” map

**Fig. 3.** Screenshots of the Visualisation Applet

are those having no specific position on the map. Indeed, for a certain map, positioning of some work items may be meaningless.

## References

1. de Leoni, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Visual Support for Work Assignment in Process-Aware Information Systems. In: Proceedings of the 6th International Conference on Business Process Management (BPM’08), Milan, Italy, September 2-4. Volume 5240 of Lecture Notes in Computer Science., Springer (2008)