Visual Support for Work Assignment in Process-aware Information Systems: Framework, Formalisation, Operationalisation

Massimiliano de Leoni[1], W.M.P. van der Aalst[2,3], and A.H.M. ter Hofstede[3]

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

Abstract. Process-aware information systems ranging from generic workflow systems to dedicated enterprise information systems use *work lists* to offer so-called *work items* to users. The work list handlers typically show a sorted list of work items comparable to the way that e-mails are presented in most e-mail programs. Since the work list handler is the dominant interface between the system and its users, it is worthwhile to provide a more advanced graphical interface that uses context information about work items and users. This paper uses the "map metaphor" to visualise work items and resources (e.g., users) in a sophisticated manner. Moreover, based on "distance notions" work items are visualised differently. For example, urgent work items of a type that suits the user are highlighted. The underlying map and distance notions may be of a geographical nature (e.g., a map of a city of office building), but may also be based on the process design, organisational structures, social networks, due dates, calenders, etc. The approach presented in this paper is supported by a visualisation framework implemented in the context of YAWL. The framework is set up in such a way that it can easily be combined with other workflow systems.
Keywords: Process-aware Information Systems, work list visualisation, YAWL

1 Introduction

Originally, *Process-Aware Information Systems* (PAISs) [1] were mainly applied in the context of administrative processes. Later their application was extended to cross-organisational processes. Currently, PAISs are starting to be used for more flexible and/or pervasive processes, e.g., disaster management scenarios [2].

Independent of the application domain and underlying technology, a PAIS is driven by some process model. The model may be implicit or hidden, but the system supports the handling of cases in some (semi-)structured form. PAISs have also in common that they offer work to resources (typically people). The elementary pieces of work are called *work items*, e.g., “Approve travel request XYZ1234”. These work items are offered to the users by the so-called *work list handler*. This component takes 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 the first resource to select the work item will be the only one executing it. To allow users to “pull the right work items in the right order”, basic information is provided, e.g., task name, due date, etc. However, given the fact that the work list is the main interface of the PAIS with its users it seems important to provide support that goes beyond a sorted list of items. If work items are selected by less qualified users than necessary or if users select items in a non-optimal order, then the performance of the overall process is hampered.

Assume the situation where multiple resources have overlapping roles and authorisations and that there are times where work is piling up (i.e., any normal business). In such a situation the questions listed below are relevant.

– “What is the most urgent work item I can perform?”
– “What work item is, geographically speaking, closest to me?”
– “Is there another resource that can perform this work item that is closer to it than me?”
– “Is it critical that I handle this work item or are there others that can also do this?”
– “How are the work items divided over the different departments?”

To our knowledge, commercial as well as open source PAISs present work lists simply as a list of work items each with a short textual description. Some products sort the work items in a work list using a certain priority scheme specified at design time and not updated at run time. To support the user in a better way and assist her in answering the above questions, we use *maps*. A map can be a geographical map (e.g., the map of a university’s campus). But other maps can be used, e.g., process schema’s, organisational diagrams, Gantt charts, etc. Work items can be visualised by dots on the map. By not fixing the type of map, but allowing this choice to be configurable, different types of relationships can be shown thus providing a deeper insight into the context of the work to be performed.

Work items are shown on maps. Moreover, for some maps also resources can be shown, e.g., the geographical position of a user. Besides the “map metaphor” we also use the “distance metaphor”. Seen 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 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, whose position is calculated based on a function supplied at design time, display their level of urgency.

This paper proposes different types of maps and distance metrics. Moreover, the framework has been implemented and integrated in YAWL.[4] YAWL is an open source workflow system based on the so-called workflow patterns. However, the framework and its implementation are set-up in such a way that it can easily be combined with other PAISs.

The paper is structured as follows. Section 2 discusses the state of the art in work list visualisation in PAISs, whereas Section 3 provides a detailed overview of the general framework. Section 4 focusses on the implementation of the framework and highlights some design choices in relation to user and system interfaces. In Section 5 the framework is illustrated through a case study. Section 6 summarises the contributions of the paper and outlines avenues of future work aimed at improving the operationalisation of the framework.

2 Related Work

Little work has been conducted in the field of work list visualisation. Visualisation techniques in the area of PAIS have predominantly been used to aid in the understanding of process schemas and their run time behaviour, e.g. through simulation [3] or process mining [4]. Although the value of business process visualisation is acknowledged, both in the literature [5–8] and in the industry, little work has been done in the context of visualising work items.

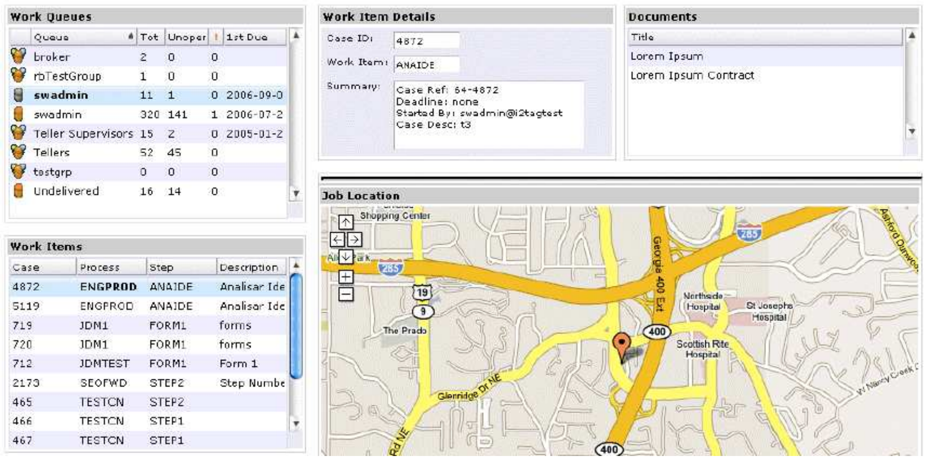
The aforementioned body of work does not provide specific support for context-dependent work item selection. This is addressed though in the work by Brown and Paik [9], whose basic idea is close to the proposal of this paper. Images can be defined as maps and mappings can be specified between work items and these maps. Work items are visualized through the use of intuitive icons and the colour of work items changes according to their state. However, the approach chosen does not work so well in real-life scenarios where many work items may have the same position (especially in course-grained maps) as icons with the same position are placed alongside each other. This may lead to a situation where a map is completely obscured by its work items. In our approach, these items are coalesced in a single dot of which the size is proportionate to their number. By gradually zooming in on such a dot, the individual work items cam become visible again. In addition, in [9] there is no concept similar to our distance notion, which is an ingredient that can provide significant assistance with work item selection to resources. Finally, the work of Brown and Paik does not take the visualisation of the positions of resources into account.

Also related is the work presented in [10], where proximity of work items is considered without discussing their visualization.

Most PAISs present work lists as a simple enumeration of their work items, their textual descriptions, and possibly information about their priority and/or their deadlines. This holds both for open source products, as e.g. jBPM[5] and

[4] www.yawlfoundation.org

[5] jBPM web site - `http://www.jboss.com/products/jbpm`



Work Queues

	Queue	Tot	Unoper	!	1st Due
	broker	2	0	0	
	rbTestGroup	1	0	0	
	swadmin	11	1	0	2006-09-0
	swadmin	320	141	1	2006-07-2
	Teller Supervisors	15	2	0	2005-01-2
	Tellers	52	45	0	
	testgrp	0	0	0	
	Undelivered	16	14	0	

Work Items

Case	Process	Step	Description
4872	**ENGPROD**	ANAIDE	Analisar Ide
5119	ENGPROD	ANAIDE	Analisar Ide
719	JDM1	FORM1	forms
720	JDM1	FORM1	forms
712	JDMTEST	FORM1	Form 1
2173	SEOFWD	STEP2	Step Numbe
465	TESTCN	STEP2	
466	TESTCN	STEP1	
467	TESTCN	STEP1	

Work Item Details

Case ID: 4872

Work Item: ANAIDE

Summary: Case Ref: 64-4872
Deadline: none
Started By: swadmin@i2tagtest
Case Desc: t3

Documents

Title
Lorem Ipsum
Lorem Ipsum Contract

Fig. 1. TIBCO's iProcess Client

Together Workflow[6], as for commercial systems, such as SAP Netweaver[7] and Flower[8]. An exception is TIBCO's iProcess Suite[9] which provides a richer type of work list handler that partially addresses the problem of supporting resources with work item selection. Figure 1 depicts a screen shot of the work list handler. In the bottom left corner a resource's work list is shown, and above this the lengths of the work lists of other resources is shown. By clicking on a work item, a resource can see it on a *Google Map* positioned where it should be executed. The iProcess Suite also supports a kind of look-head in the form of a list of "predicted" work items and their start times. One can also learn about projected deadline expirations and exception flows. This is achieved through the use of expected durations specified at design time for the various tasks. Our visualisation framework is more accurate as it can take actual execution times of work items of a task into account through the use of log files when considering predictions for new work items of that task. Basically, the iProcess Suite provides support for some specific views (geographical position, deadline expiration) but these are isolated from each other. Our approach allows these views (and others) to be combined (e.g. a geographical view where deadlines are also visualised) thus enabling the use of views that may prove useful in certain contexts. Our approach also generalises over the type of map and goes beyond support for a single map as in the iProcess Suite (a geographical map).

[6] Together Workflow web site - `http://www.together.at/together/prod/tws/`
[7] Netweaver web site - `http://www.sap.com/usa/platform/netweaver`
[8] Flower web site - `http://global.pallas-athena.com/products/bpmflower_product/`
[9] iProcess Suite web site - `http://www.tibco.com/software/business_process_management/`

Table 1. Examples of maps and mappings.

Process context view	**Possible map and mapping**
The physical environment where tasks are going to be performed.	A real geographical map (e.g., Google maps). Work items are placed where they should be performed and resource are placed where they are located.
The process schema of the case that work items belong to.	The process schema is the map and work items are placed on top of tasks that they are an instance of.
Deadline expiration of work items.	The map is a time-line where the origin is the current time. Work items are placed on the time-line at the latest moment when they can start without their deadline expiring.
The organisation that is in charge of carrying out the process.	The map is an organizational chart. Work items are associated with the role required for their execution. Resources are also shown based on their organizational position.
The materials that are needed for carrying out work items.	The map is a multidimensional graph where the axes are the materials that are needed for work item execution. Let us assume that materials A and B are associated with axes x and y respectively. In this case, a work item is placed on coordinates (x, y) if it needs a quantity of x of material A and a quantity y of material B.
Costs versus benefits in executing work items.	In this case, the axes represent "Revenue" (the amount of money received for the performance of work items) and "Cost" (the expense of their execution). A work item is placed on coordinates (x, y) if the revenue of its execution is x and its cost is y. In this case one is best off executing work items close to the x axis and far from the origin.

3 The General Framework

The proposed visualisation framework is based on a two-layer approach: (1) maps and (2) the visualisation of work items based on a distance notion. A work item is represented as a dot positioned along certain coordinates on a background map. A map is meant to capture a particular perspective of the context of the process. Since a work item can be associated with several perspectives, it can be visualised in several maps (at different positions). Maps can be designed as needed. When the use of a certain map is envisaged, the relationship between work items and their position on the map should be specified through a function determined at design time. Table 1 gives some examples of context views and the corresponding work item mapping.

Several active "views" can be supported whereby users can switch from one view to another. Resources can (optionally) see their own position on the map and work items are coloured according to the value of the applicable distance metric. Additionally, it may be helpful to show executing work items as well as the position of other resources. Naturally, these visualisations are governed by the authorisations that are in place.

Our framework assumes a generic lifecycle model as described in [11]. First, a work item is *created* indicating that it is ready for distribution. The item is then *offered* to appropriate resources. A resource can commit to the execution of the item, after which it moves to the *allocated* state. The start of its execution leads it to the next state, *started*, after which it can successfully *complete*, it can be *suspended* (and subsequently *resumed*) or it can *fail* altogether. During run-time a workflow engine (in our case the YAWL engine) informs the framework about the lifecyle states of work items.

3.1 Fundamentals

In this section the various notions used in our framework, e.g. work item and resource, are defined formally.

Definition 1 (Work item). *A work item w is a tuple (c, t, i, y, e, l), where:*

- c *is the identifier of the case that w belongs to.*
- t *is the identifier of the task of which w is an instance.*
- i *is a unique instance number.*
- y *is the timestamp capturing when w moved to the "offered" state.*
- e *is the (optional) deadline of w.*
- l *represents the (optional) GPS coordinates where w should be executed.*

Dimensions y and l may be *undefined* in case the work item w is not yet offered or no specific execution location exists respectively. The e value concerns timers which may be defined in YAWL processes. A process region may be associated with a timer. When the timer expires, the work items part of the region are cancelled. Note that a work item can potentially be a part of more than one cancellation region and that this has implications for the definition of y. In such a case the latest possible completion time with respect to these cancellation regions is assumed.

Definition 2 (Resource). *A resource r is a pair (j, l), where:*

- j *is the identifier of the resource.*
- l *represents the (optional) GPS coordinates where the resource is currently located.*

The notation w_x is used to denote the projection on dimension x of work item w, while the notation r_y is used to denote the projection on dimension y of resource r. For example, w_t yields the task of which work item w is an instance.

Work items w' and w'' are considered to be *siblings* iff $w'_t = w''_t$. The set *Coordinates* consists of all possible coordinates. Elements of this set will be used to identify various positions on a given map.

Definition 3 (Position function). *Let W and R be the set of work items and resources. Let M be the set of available maps. For each available map $m \in M$, there exists a function $position_m : W \cup R \nrightarrow Coordinates$ which returns the current coordinates for work items and available resources on map m.*

For a map $m \in M$, the function $position_m$ may be partial, since some elements of W and/or R may not have an associated position. Consider for example the case where a work item can be performed at any geographical location or where it does not really make sense to associate a resource with a position on a certain map. As the various attributes of work items and resources may vary over time it is important to see the class of functions $position_m$ as time dependent.

To formalise the notion of distance metric, a distance function is defined for each metric that yields the distance between a work item and a resource according to that metric.

Table 2. Distance Metrics currently provided by the implementation

Metric	**Returned Value**
$distance_{Familiarity}(w, r)$	How familiar is resource r with performing work item w. This can be measured through the number of sibling work items the resource has already performed.
$distance_{Geo\_Distance}(w, r)$	How close is resource r to work item w compared to the closest resource that was offered w. For the closest resource this distance is 1. In case w does not have a specific GPS location where it should be executed, this metric returns 1 for all resources.
$distance_{Popularity}(w, r)$	The ratio of logged-in resources having been offered w to all logged-in resources. This metric is independent from resource r making the request.
$distance_{Urgency}(w, r)$	The ratio between the current timestamp and the latest timestamp when work item w can start but is not likely to expire. The latter timestamp is obtained as the difference between w_e, the latest timestamp when w has to be finished without expiring, and w's estimated duration. This estimation is based on past execution of sibling work items of w by r.
$distance_{Past\_Execution(w,r)}$	How familiar is resource r with work item w compared to the familiarity of all other resources that w has been offered to. More information about this metric is provided in the text.

Definition 4 (**Distance function**). *Let W and R be the set of work items and resources. Let D be the set of available distance metrics. For each distance metric $d \in D$, there exists a function $distance_d : W \times R \rightarrow [0, 1]$ that returns a number in the range [0,1] capturing the distance between work-item $w \in W$ and resource $r \in R$ with respect to metric d.*[10]

Given a certain metric d and a resource r, the next work item r should perform is a work item w for which the value $distance_d(w, r)$ is the closest to 1 among all offered work items.

3.2 Available Metrics

In Table 2 a number of general-purpose distance metrics are informally explained. These are all provided with the current implementation. Later in this section, we formalise the notion of metrics. Let us denote R as the set of resources currently logged in. In order to make explanations easier, some auxiliary functions are introduced.

past_execution(w,r) yields the weighted mean of the past execution times of the last h work items performed by r among all work item siblings of w. In this context, the past execution time of work item w' is defined as the duration that elapsed between its assignment to r and its successful completion. Let $time_i(w, r)$ be the execution time of the i-th last work item among w's siblings performed by r, then:

$$past\_execution(w, r) = \frac{\sum_{i=1}^{h} \alpha^{i-1} \cdot time_i(w, r)}{\sum_{i=1}^{h} \alpha^{i-1}} \quad (1)$$

[10] Please note the value 1 represents the minimum distance while 0 is the maximum.

where $\alpha \in [0,1]$. Both h and α have to be tuned through testing. The intuition behind this definition stems from the fact that more recent executions should be given more consideration and hence weighted more as they better reflect resources gaining experience in the execution of instances of a certain task.

Res(w) returns all currently logged-in resources that have been offered w:

$$Res(w) = \{r \in R \mid w \ is \ offered \ to \ r\}.$$

best_past_execution(w) denotes the smallest value for $past\_execution(w,r)$ computed among all logged-in resources r qualified for w. Specifically:

$$best\_past\_execution(w) = \min_{r' \in Res(w)} past\_execution(w, r')$$

bestDistance(w) returns the minimum geographic distance between a given work-item w and all qualified resources:

$$best\_Distance(w) = \min_{r' \in Res(w)} \|w_l - r'_l\|$$

where $\|w_l - r'_l\|$ stands for the Euclidian distance between the GPS coordinates where w should be executed and the GPS location of resource r. Function $best\_Distance(w)$ is not total since w_l may be undefined for certain work items w.

Using these auxiliary functions the following metrics can be defined:

1. Familiarity. How familiar is resource r with performing work item w. This can be measured through the number of sibling work items the resource has already performed:

$$distance_{Familiarity}(w,r) = \begin{cases} 0 & best\_past\_execution(w) \rightarrow \infty \\ \frac{best\_past\_execution(w)}{past\_execution(w,r)} & otherwise \end{cases}$$

The $best\_past\_execution(w)$ value can tend to infinite, if nobody has ever executed work items for task w_t. Otherwise, if someone executed work item siblings of w_t but r did not, then $past\_execution(w,r) \rightarrow \infty$ and, hence, $distance_{Familiarity}(w,r) \rightarrow 0$.

2. Popularity. The ratio of logged-in resources having been offered w to all logged-in resources. This metric is independent from resource r making the request. The intuition is that if many resources can perform w then it is quite distant from every resource. Indeed, even if a resource doesn't pick w for performance, it is likely someone else may execute w. Therefore:

$$distance_{Popularity}(w,r) = 1 - \frac{|Res(w)|}{|R|}$$

If every resource can perform w, then the distance is 0. If many resources can perform w, then the value is near to 1.

3. Urgency. The ratio between the current timestamp and the latest timestamp when work item w can start but is not likely to expire. This second timestamp is obtained from w_e, the latest timestamp when w has to be finished without expiring, and w's estimated duration. This estimation relies on the past execution by r of w's sibling work items. Specifically:

$$distance_{Urgency}(w,r) = \begin{cases} 1 - \frac{t_{now}}{w_e - pastExecution(w,r)} & w_e \ is \ defined \\ 0 & w_e \ is \ undefined \end{cases}$$

where t_{now} stands for the current timestamp. If r has never performed work-items for the same task w_t, $pastExecution(w,r) \rightarrow \infty$ and, hence, $distance_{Urgency}(w,r) \rightarrow 0$.

4. Relative Geographic Distance. How close is resource r to work item w compared to the closest resource that was offered w. For the closest resource this distance is 1. In case w does not have a specific GPS location where it should be executed, this metric returns 1 for all resources. Its definition is:

$$distance_{Relative\_Geo}(w,r) = \begin{cases} 1 - \frac{\|w_l - r_l\|}{bestDistance(w)} & bestDistance_t(w) > 0 \\ 0 & bestDistance_t(w) = 0 \\ 1 & bestDistance_t(w) \ is \ undef \end{cases}$$

5. Relative Past Execution. The metric chosen combines the familiarity of a resource with a certain work item and the familiarity of other resources that are able to execute that work item:

$$distance_{Relative\_Past\_Execution}(w,r) = \frac{1/past\_execution(w,r)}{\sum\limits_{r' \in Res(w)} 1/past\_execution(w,r)} \quad (2)$$

Let us give an informal explanation. First observe that if exactly one resource r exists capable of performing work item w, then the equation yields one. If n resources are available and they roughly have the same familiarity with performing work item w, then for each of them the distance will be about $1/n$. It is clear then that as n increases in value, the value of the distance metric approaches zero. If on the other hand many resources exist that are significantly more effective in performing w than a certain resource r, then the value of the denominator increases even more and the value of the metric for w and r will be closer to zero.

For instance, let us suppose that at time $\hat{t}$ there are n resources capable of performing w. Let us assume that, on average, one of them, namely r_1 is such that $past\_execution(w, r_1) = \tilde{d}$. Moreover, let us also assume that the other resources required twice this amount of time on average in the past, i.e. for each resource r_i (with $i > 1$) $past\_execution(w, r_i) = 2\tilde{d}$.

In such a situation, the distance metric value for r_1 is as follows:

$$\begin{aligned}
&distance(w, r_1, \mathsf{Relative\_Past\_Execution}) = \\
&= \frac{\frac{1}{past\_execution(w,r_1)}}{\frac{1}{past\_execution(w,r_1)} + \sum_{i=2}^{n} \frac{1}{past\_execution(w,r_i)}} \\
&= \frac{\frac{1}{\tilde{d}}}{\frac{1}{\tilde{d}} + \sum_{i=2}^{n} \frac{1}{2\tilde{d}}} = \frac{1}{1 + \frac{n-1}{2}} = \frac{2}{1+n}
\end{aligned}$$

This value is greater than $1/n$, if $n > 1$ (i.e., there are al least two resources that may perform w). If $n = 1$, then it is easy to see that the obtained value is 1 for both.

Conversely, the value for any other resource r_i (with $i > 1$) is as follows:

$$\begin{aligned}
&distance(w, r_i, \mathsf{Relative\_Past\_Execution}) = \\
&= \frac{\frac{1}{past\_execution(w,r_i)}}{\frac{1}{past\_execution(w,r_1)} + \sum_{i=2}^{n} \frac{1}{past\_execution(w,r_i)}} \\
&= \frac{\frac{1}{2\tilde{d}}}{\frac{1}{\tilde{d}} + \sum_{i=2}^{n} \frac{1}{2\tilde{d}}} = \frac{1/2}{1 + \frac{n-1}{2}} = \frac{1}{1+n}
\end{aligned}$$

For all $n > 0$, this value is smaller than $\frac{2}{n+1}$, that is the metric value for r_1.

Work-item ageing. Some of the metrics above suffer from the fact that their values do not change over time. Therefore, if some work-items have a small value with respect to those metrics, it is likely that there are always other work items that have a greater value for those metrics. If resources behave "fairly", picking always work items that provide more benefit for the organizations, some work-items could remain on a work list for a very long time or even indefinitely.

Therefore, we devised a technique of ageing work-items that occur on work lists in such a way that they eventually become the least distant work item. Let d be any metric and $\chi_{t_{en}} = distance_d(w, r)$ be the distance value when w becomes enabled, where w, r are, respectively, a metric and resource. The distance value with respect to metric d at time $t_{en} + t$ ages as follows:

$$\chi_{t_{en}+t} = 1 - (1 - \chi_{t_{en}}) \cdot \exp^{-\alpha \cdot t} \qquad (3)$$

If $t = 0$, then $\chi_{t_{en}+t} = \chi_{t_{en}}$ and if $t \to \infty$ (i.e., time t increases indefinitely), then $\chi_{t_{en}+t} \to 1$. Please note that if $\alpha = 0$, then work-items do not age. The greater value α, the more quickly Equation 3 approaches 1 when t increases. Vice versa, smaller values of α make the growth of Equation 3 with t slower.

4 Implementation

The general framework described in the previous section has been operationalised through the development of a component that can be plugged into the YAWL system. The YAWL environment is an open source PAIS, based on the workflow patterns[11], using a service-oriented architecture. The YAWL engine and all other services (work list handler, web-service broker, exception handler, etc.) communicate through XML messages. The YAWL work list handler was developed as a web application. In its graphical interface different tabs are used to show the various queues (e.g. offered work items). The visualisation framework can be accessed through a newly introduced tab and is implemented as a Java Applet.

Section 4.1 illustrates some of the visualisation features provided by the implementation, whereas Section 4.2 focusses on how the component fits within the YAWL architecture.

4.1 The User Interface

The position and distance functions represent orthogonal concepts that require joint visualisation for every map. The position function for a map determines where work items and resources will be placed as dots, while the distance function will determine the colour of work items. Conceptually, work item information and resource information is split and represented in different layers. Users can choose which layers they wish to see and in case they choose both layers which of them should overlay the other.

Work-item Layer. Distances can be mapped to colours for work items through a function $colour : [0, 1] \rightarrow C$ which associates every metric value with a different colour in the set C. In our implementation colours range from white to red, with intermediate shades of yellow and orange. When a resource sees a red work item this could for example indicate that the item is very urgent, that it is one of those most familiar to this resource, or that it is the closest work item in terms of its geographical position. While the colour of a work item can depend on the resource viewing it, it can also depend on which state of the lifecycle it is in. Special colours are used to represent the various states of the work item lifecycle and Table 3 provides an overview. The various rows correspond to the various states and their visualisation. Resources can filter work items depending on the state of items. This is achieved through the provision of a checkbox for each of the states of Table 3. Several checkboxes can be ticked. There is an additional checkbox which allows resources to see work items that they cannot execute, but they are authorised to see.

Resources may be offered work items whose positions are the same or very close. In such cases their visualisations may overlap and they are grouped into a so-called "joint dot". The diameter of a joint dot is proportional to the number of work items involved. More precisely, the diameter D of a joint dot is determined

[11] www.workflowpatterns.com

Table 3. Visualisation of a work item depending on its state in the life cycle.

Work item state	**Colour scheme used in the work-list handler**
Created	Work item is not shown.
Offered to single/multiple resource(s)	The colour is determined by the distance to the resource with respect to the chosen metric. The colour ranges from white through various shades of yellow and orange to red.
Allocated to a single resource	Purple.
Started	Black.
Suspended	The same as for offered.
Failed	Grey.
Completed	Work item is not shown.

by $D = d(1 + \lg n)$, where d is the standard diameter of a normal dot and n is the number of work items involved. Note that we use a logarithmic (lg) scaling for the relative size of a composite dot.

Combining several work items int a single dot raises the question of how the distance of this dot is determined. Four options are offered for defining the distance of a joint dot, one can take a) the maximum of all the distances of the work items involved, b) their minimum, c) their median, or d) their mean. When a resource clicks on a joint dot, all work items involved are enumerated in a list and they are coloured according to their value in terms of the distance metric chosen.

Resource Layer. When a resource clicks on a work item the positions of the other resources to whom this work item is offered are shown. Naturally this is governed by authorisation privileges and by the availability of location information for resources for the map involved.

Resource visualisation can be customised so that a resource can choose to see a) only herself, b) all resources, or c) all resources that can perform a certain work item. The latter option supports the case where a resource clicks on a work item and wishes to see the locations of the other resources that can do this work item.

4.2 Architectural Considerations

Figure 2 shows the overall architecture of the visualisation framework and the connections with other YAWL components. Specifically, the visualisation framework comprises:

The Visualisation Applet is the client-side applet that allows resources to access the visualisation framework and it resides as a separate tab in the work-list handler.

The Visualisation Designer is used by special administrators in order to define and update maps as well as to specify the position of work items on defined maps. Designers can define positions as fixed or as variable through the use of XQuery. In the latter case, an XQuery expression is defined that

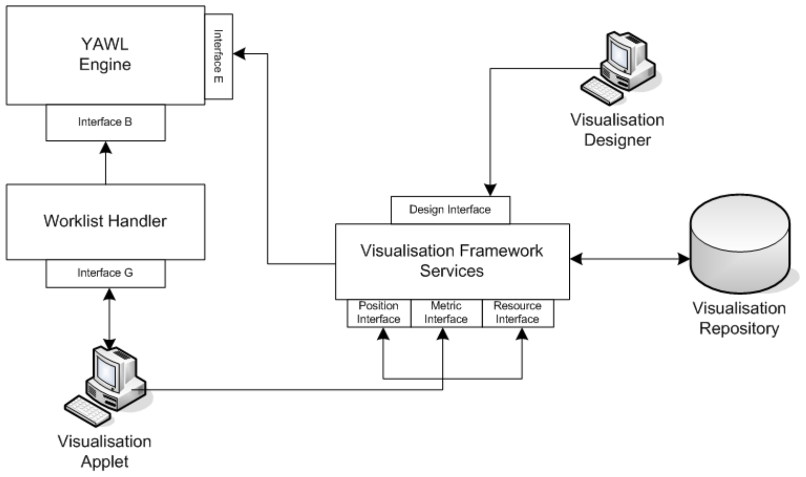


Fig. 2. Position of the visualisation components in the YAWL architecture.

refers to case variables. This expression is evaluated at run time when required.

Services is the collective name for modules providing information used to depict maps and to place work items (e.g. URLs to locate map images, work item positions on various maps).

The *YAWL engine* is at the heart of the YAWL environment. It determines which work items are enabled and can thus be offered for execution and it handles the data that is involved. While the YAWL engine offers a number of external interfaces, for the visualisation component interfaces B and E are relevant. Interface B is used, for example, by the work list handler to learn about work items that need to be offered for execution. This interface can also be used for starting new cases. Interface E provides an abstraction mechanism to access log information, and can thus e.g. be used to learn about past executions of siblings of a work item. In particular one can learn how long a certain work item remained in a certain state.

The *work list handler* is used by resources to access their “to-do” list. The standard version of the work list handler provides queues containing work items in a specific state. This component provides interface G which allows other components to access information about the relationships between work items and resources. For example, which resources have been offered a certain work item or which work items are in a certain state. Naturally this component is vital to the Visualisation Applet.

In addition to interface G, the Visualisation Applet also connects to the *Services* modules through the following interfaces:

The Position Interface provides information about maps and the positioning of work items on these maps. Specifically, it returns an XQuery over the

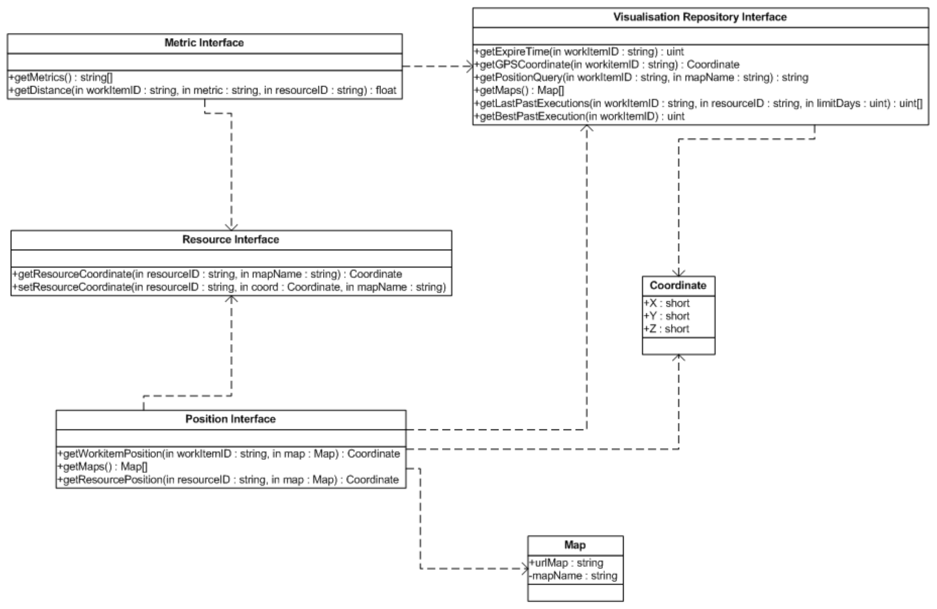

Metric Interface
+getMetrics() : string[]
+getDistance(in workItemID : string, in metric : string, in resourceID : string) : float
Visualisation Repository Interface
+getExpireTime(in workItemID : string) : uint
+getGPSCoordinate(in workitemID : string) : Coordinate
+getPositionQuery(in workItemID : string, in mapName : string) : string
+getMaps() : Map[]
+getLastPastExecutions(in workItemID : string, in resourceID : string, in limitDays : uint) : uint[]
+getBestPastExecution(in workItemID) : uint
Resource Interface
+getResourceCoordinate(in resourceID : string, in mapName : string) : Coordinate
+setResourceCoordinate(in resourceID : string, in coord : Coordinate, in mapName : string)
Coordinate
+X : short
+Y : short
+Z : short
Position Interface
+getWorkitemPosition(in workItemID : string, in map : Map) : Coordinate
+getMaps() : Map[]
+getResourcePosition(in resourceID : string, in map : Map) : Coordinate
Map
+urlMap : string
-mapName : string

Fig. 3. Details of the interfaces provided.

YAWL net variables that the Visualisation Applet has to compute. The work list handler needs to be consulted to retrieve the current values of these variables.

The Metric Interface provides information about available metrics and their values for specific work item - resource combinations.

The Resource Interface is used to update and retrieve information concerning positions of active resources on maps.

The visualisation framework was integrated into the standard work list handler of YAWL through the addition of a JSP (Java Server Page).

All of the services of the visualisation framework share a repository, referred to as *Visualisation Repository* in Figure 2, which stores, among others, XQueries to compute positioning information, resource locations in various maps, and names and URLs of maps. Services periodically retrieve log data through Interface E in order to compute distance metric values for offered work items. For instance, to compute the metric *Relative Past Execution* (Equation 2) for a certain resource, one can see from Equation 1 that information is required about the h past executions of sibling work items performed by that resource.

To conclude this section, we would like to stress that the approach and implementation are highly generic, i.e., it is relatively easy to embed the visualisation framework in another PAIS.

Interface Details. The modules which are collectively named *Service* are implemented as Tomcat web applications. Specifically, each interface is implemented as a web application and methods are provided as servlets, which take inputs and return outputs as XML documents.

Figure 3 summarizes the methods offered by all implemented interfaces. Although they are actually servlets and parameters XML documents, we conceptualise them as methods of classes of an object-oriented programming language.

Interface *Metric* provides two methods to get: 1) all available metrics (specifically `getMetrics()`), which returns the list of metric names and 2) the distance metric value for single work items (i.e., `getDistance()`), which takes a work item identifier and a metric name as input and returns the value for that metric for that work item.

Interface *Resource* provides two methods basically to get and set the resource position with respect to a specified map.

Finally, interface *Position* allows one to request information about all available maps through method `getMaps()`. In particular, it returns an array of objects *Map*. Each object defines two properties: 1) the map name and 2) the URL where the map can be found. Method `getResourcePosition()` takes a resource identifier and a given map as input, and returns the coordinates of such a resource on the map specified. This information is mostly what resources themselves provide through method `setResourceCoordinate()` of interface *Resource*. Method `getWorkitemPosition()` of interface *Position* is very similar but operates on work items instead of resources.

None of the interfaces accesses the *Visualisation Repository* database directly for modularity questions. In fact, the *Visualisation Repository* Interface exists solely for the purpose of masking interaction with database, namely *Visualisation Repository Interface*. As the various methods are sufficiently self-explanatory we are not providing more details.

The only thing worth mentioning is that `getLastPastExecutions` returns the duration of the last h sibling work items offered within the last *limitDays* days. This method is required for computing function *pastExecution*. In order to return the h more recent executions, the method needs to obtain all work items and, then, to sort them in descending order by timestamp when they moved to the offered state (i.e., work item dimension y). Finally, the method considers the first h work items in such a sorted listed. We foresee an initial filtering, discarding all work items that were offered more than *limitDays* days ago. If this filtering was not performed, the sorting operation could be computationally hard, as it could involve thousands of work items. Therefore, filtering is meant to reduce the size of the set to be sorted.

5 Example: Emergency Management

In this section we are going to illustrate a number of features of the visualisation framework by considering a potential scenario from emergency management. This scenario stems from a user requirement analysis conducted in the context of a European-funded project [2]. Teams are sent to an area to make an assessment of the aftermath of an earthquake. Team members are equipped with a laptop and their work is coordinated through the use of a PAIS.

The main process of workflow for assessing buildings is named *Disaster Management*. The first task *Assess the affected area* represents a quick on-the-spot inspection to determine damage to buildings, monuments and objects. For each object identified as worthy of further examination an instance of the sub-process *Assess every sensible object* (of which we do not show the actual decomposition for space reasons) is started as part of which a questionnaire is filled in and photos are taken. This can be an iterative process as an evaluation is conducted to determine whether the questionnaire requires further refinement or more photos need to be taken. After these assessments have finished, the task *Send data to the headquarters* can start which involves the collection of all questionnaires and photos and their subsequent dispatch to headquarters. This information is used to determine whether these objects are in imminent danger of collapsing and if so, whether this can be prevented and how that can be achieved. Depending on this outcome a decision is made to destroy the object or to try and restore it.

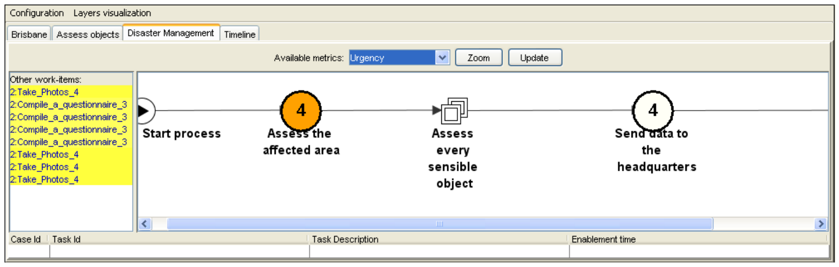
For the purposes of illustrating our framework we assume that an earthquake has occurred in the city of Brisbane. Hence a number of cases are started by instantiating the *Disaster Management* workflow described above.

Each case deals with the activities of an inspection teams in a specific zone. Figure 4 shows three maps. In each map, the dots refer to work items. Figure 4(a) shows the main process of the *Disaster Management* workflow, including eight work items. Dots for work items which are instances of the tasks *Assess the affected area* and *Send data to the headquarter* are placed on top of these tasks in this figure. Figure 4(b) shows the decomposition of *Assess every sensible object*. Here also eight work items are shown. No resources are shown in these diagrams. Note that on the left-hand side is shown a list of work items that are not on the map. For example, the eight work items shown in the map in Figure 4(a) appear in the list of *"other work items"* in Figure 4(b).

Figure 4(a) uses the urgency distance metric to highlight urgent cases while Figure 4(b) uses the familiarity metric to highlight cases closer to the user in terms of earlier experiences.

As another illustration consider Figure 4(c) where work items are positioned according to their deadlines. This can be an important view in the context of disaster management where saving minutes may save lives. In the map shown, the x-axis represents the time remaining before a work item expires, while the y-axis represents the case number of the case the work item belongs to. A work item is placed at location $(100 + 2 * \widetilde{x}, 10 + 4 * \widetilde{y})$ on that map, if $\widetilde{x}$ minutes are remaining to the deadline of the work item and its case number is $\widetilde{y}$. In this example, work items are coloured in accordance with the *popularity* distance metric.

Figures 5 and 6 show some screenshots of a geographical map of the city of Brisbane. On these maps, work items are placed at the location where they should be executed. If their locations are so close that their corresponding dots overlap, a larger dot (i.e., a joint-dot) is used to represent the work items in-

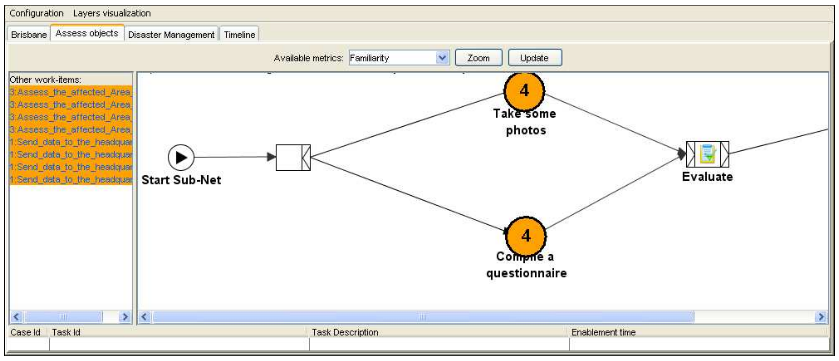


Available metrics: Urgency

Other work-items:
2:Take_Photos_4
2:Compile_a_questionnaire_3
2:Compile_a_questionnaire_3
2:Compile_a_questionnaire_3
2:Compile_a_questionnaire_3
2:Take_Photos_4
2:Take_Photos_4
2:Take_Photos_4

Case Id	Task Id	Task Description	Enablement time

(a) *Disaster Management* process map showing 4+4=8 work items

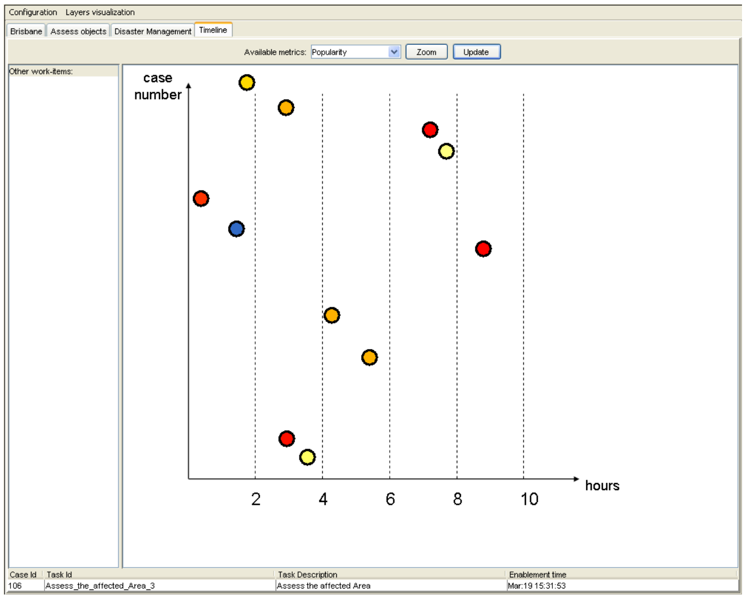


Available metrics: Familiarity

Other work-items:
3:Assess_the_affected_Area
3:Assess_the_affected_Area
3:Assess_the_affected_Area
3:Assess_the_affected_Area
1:Send_data_to_the_headquar
1:Send_data_to_the_headquar
1:Send_data_to_the_headquar
1:Send_data_to_the_headquar

Case Id	Task Id	Task Description	Enablement time

(b) *Assess the affected area* sub-net also showing 8 work items



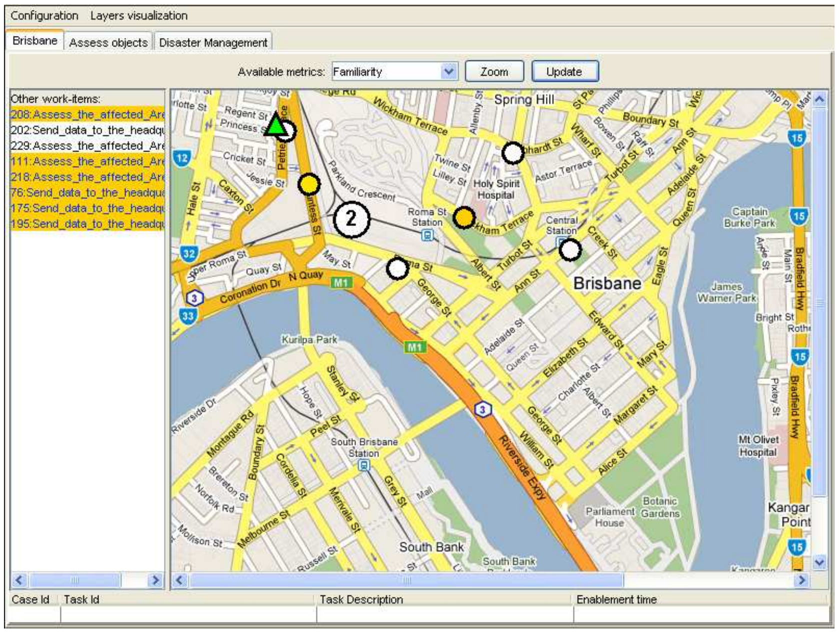
Available metrics: Popularity

Other work-items:

Case Id	Task Id	Task Description	Enablement time
106	Assess_the_affected_Area_3	Assess the affected Area	Mar:19 15:31:53

(c) Example of a timeline map for showing 11 work items

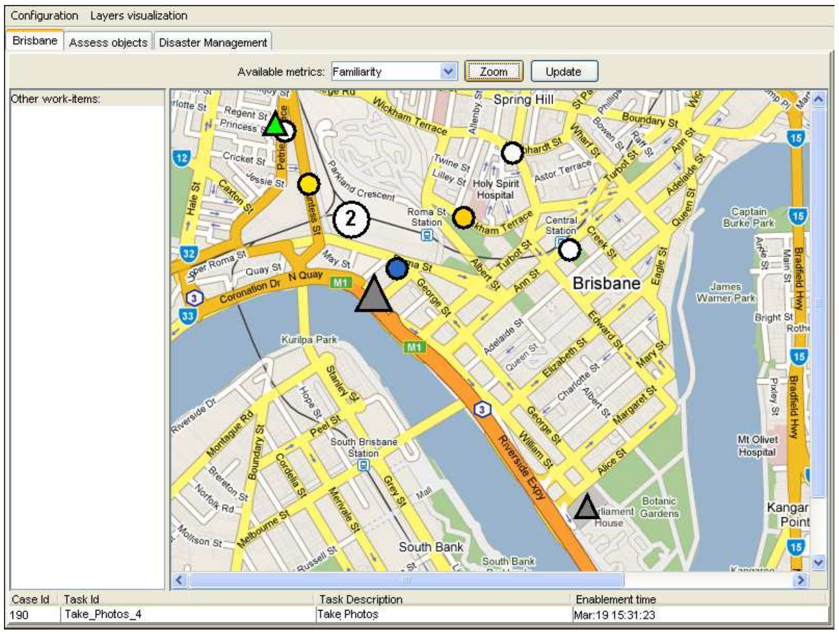
Fig. 4. Examples of Process and Timeline Maps for Disaster Management



Other work-items:
208:Assess_the_affected_Are
202:Send_data_to_the_headqu
229:Assess_the_affected_Are
111:Assess_the_affected_Are
218:Assess_the_affected_Are
76:Send_data_to_the_headqu
175:Send_data_to_the_headqu
195:Send_data_to_the_headqu

Case Id	Task Id	Task Description	Enablement time

(a) Map showing the geographic locations of work items and resources: the triangle represents the resource and most work items are shown as single dots except for the two work items that are clustered into a single dot labeled "2"

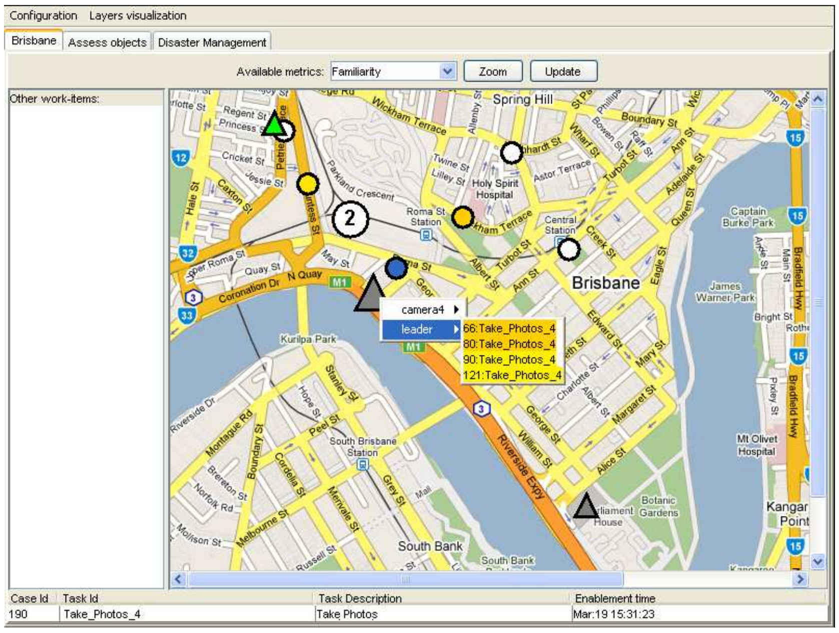


Other work-items:

Case Id	Task Id	Task Description	Enablement time
190	Take_Photos_4	Take Photos	Mar:19 15:31:23

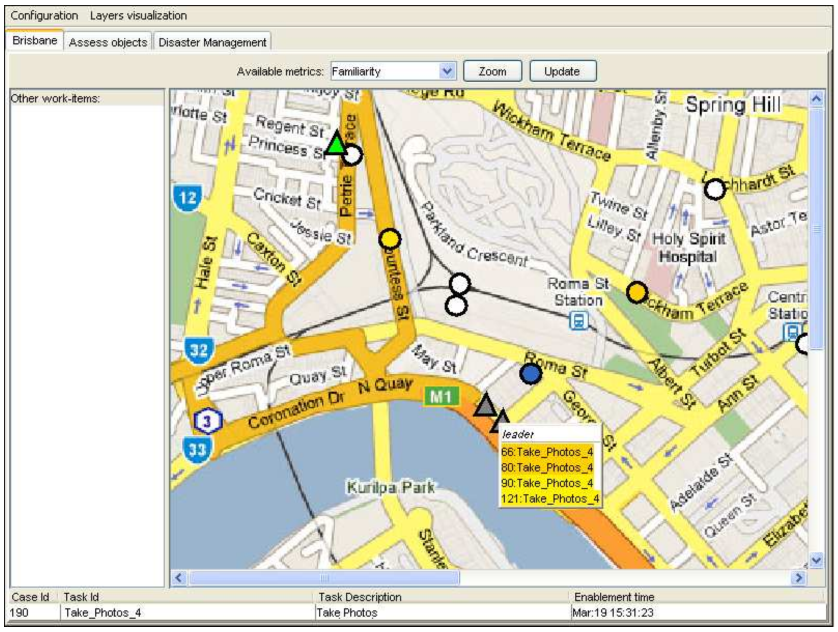
(b) Information about the selected dot (blue dot) is shown and also other resources are shown

Fig. 5. Examples of Geographic Maps for Disaster Management.



Case Id	Task Id	Task Description	Enablement time
190	Take_Photos_4	Take Photos	Mar:19 15:31:23

(a) When a triangle is selected, the corresponding resources and offered work items are shown



Case Id	Task Id	Task Description	Enablement time
190	Take_Photos_4	Take Photos	Mar:19 15:31:23

(b) When zooming in, clustered work items and resources are separated

Fig. 6. Further examples of Geographic Maps for Disaster Management.

volved and the number inside corresponds to the number of these items. The green triangle is a representation of the resource whose work list is visualised here. Work items for tasks *Assess the affected area* and *Send data to the headquarters* are not shown on the map as they can be performed anywhere. In this example, dots are coloured according to the *familiarity* distance metric. A dot that is selected as focus obtains a blue colour and further information about the corresponding work item is shown at the bottom of the screen (as is the case for work item *Take_Photos_4* in Figure 5(b)).

One can click on a dot and see the positions of other resources that have been offered the corresponding work item. For example, by clicking on the dot representing the work item *Take_photo_4*, other resources, represented by triangles, are shown (see Figure 5(b)). As for work items, overlapping triangles representing resources are combined. For examples, the larger triangle shown in Figure 5(b) represents two resources.

Figure 6(a) shows the screen shot after clicking on the joint triangle. A resource can thus see the list of resources associated with this triangle. By selecting one of the resources shown in the list, the work items offered to that resource can be seen. The colour of these work items is determined by their value for the chosen distance metric. A zooming feature is also provided. Figure 6(b) shows the result of zooming in a bit further on the map of Figure 6(a). As can be seen, no dots nor any triangles are overlapping anymore.

6 Conclusions

In this paper a general visualisation framework is proposed that can aid users in selecting the "right" work item among a potentially large number of work items offered to them. The framework uses the "map metaphor" to show the locations of work items and resources. The "distance metaphor" is used to show which work items are "close" (e.g., urgent, similar to earlier work items, or geographically close). Both concepts are orthogonal and this provides a great deal of flexibility when it comes to presenting work to people. For example, one can choose a geographical map to display work items and resources and use a distance metric capturing urgency. The proposed framework was operationalised as a component of the YAWL environment. By using well-defined interfaces the component is generic so that in principle it could be exploited by other PAISs as well under the provision that they are sufficiently "open" and provide the required interface methods. The component is also highly configurable, e.g., it allows resources to choose how distances should be computed for dots representing a number of work items and provides customizable support for determining which resources should be visible.

Finally, it should be pointed out that the implementation for the Visualisation Designer is still lacking. In the current evaluation, we manually updated the information stored in the Visualisation Repository by accessing tables in the DBMS. All other parts are fully operational.

Further research aims at connecting the current framework to geographical information systems and process mining tools like ProM [4]. Geographical information systems store data based on locations and process mining can be used to extract data from event logs and visualise this on maps, e.g., it is possible to make a “movie” showing the evolution of work items based on historic data.

Acknowledgements. The work was primarily conducted during a visit of Massimiliano de Leoni to the Business Process Management Group at Queensland University of Technology. His internship has been partly supported by the European Commission through the project FP6-2005-IST-5-034749 WORKPAD. The authors would like to thank Michael Adams for implementing certain required YAWL interfaces and for answering numerous questions. We also gratefully acknowledge Guy Redding for providing some code to connect to the YAWL interfaces.

References

1. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005)
2. Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Vetere, G., Salvatore, B., Dustdar, S., Juszczyk, L., Manzoor, A., Truong, H.L.: Pervasive Software Environments for Supporting Disaster Responses. IEEE Internet Computing **12** (2008) 26–37
3. Hansen, G.: Automated Business Process Reengineering: Using the Power of Visual Simulation Strategies to Improve Performance and Profit. Prentice-Hall, Englewood Cliffs (1997)
4. van der Aalst, W.M.P., van Dongen, B., Christian, G., Mans, R.S., Alva de Medeiros, A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W., Weijters, A.J.M.M.: Prom 4.0: Comprehensive support for *real* process analysis. In: Proceedings of the 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency ICATPN 2007. Volume 4546 of LNCS., Springer (2007) 484–494
5. Bobrik, R., Reichert, M., Bauer, T.: View-based process visualization. In: Proceedings of the 5th International Conference on Business Process Management BPM 2007. Volume 4714 of LNCS., Springer (2007) 88–95
6. Luttighuis, P., Lankhorst, M., Wetering, R., Bal, R., Berg, H.: Visualising business processes. Computer Languages **27** (2001) 39–59
7. Streit, A., Pham, B., Brown, R.: Visualization support for managing large business process specifications. In: Proceedings of the 3rd International Conference on Business Process Management BPM 2005. Volume 3649 of LNCS., Springer (2005) 205–219
8. Wright, W.: Business Visualization Adds Value. IEEE Computer Graphics and Applications **18** (1998) 39
9. Brown, R., Paik, H.Y.: Resource-centric worklist visualisation. In: Proceedings of OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005. Volume 3760 of LNCS., Springer (2005) 94–111

10. Kumar, A., Aalst, W., Verbeek, H.: Dynamic Work Distribution in Workflow Management Systems: How to Balance Quality and Performance? Journal of Management Information Systems **18** (2002) 157–193
11. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Proceedings of 17th International Conference CAiSE 2005. Volume 3520 of LNCS., Springer (2005) 216–232