

Collaboration and Knowledge Sharing in Grid Applications

Michal Laclavik¹, Emil Gatial¹, Zoltan Balogh¹, Ladislav Hluchy¹

¹ Institute of Informatics, Slovak Academy of Sciences, Dubravska cesta 9
845 07 Bratislava, Slovakia
Laclavik.ui@savba.sk
<http://www.ui.sav.sk>

Abstract. In this paper we describe a solution for collaboration and knowledge sharing based on text notes entered by a user. Theory, implementation and use of such system – EMBET is described. The key idea is that a user enters notes in a particular situation/context, which can be detected by the computer. Such notes are later displayed to other or the same users in a similar situation/context. The context of user is detected from computerized tasks performed by user. In the K-Wf Grid, grid services are semi-automatically composed to workflows, which should solve a user problem. It was identified that even when services and input and output data are well semantically described, there is often no possibility to compose an appropriate workflow e.g. because of missing specific input data or fulfillment of a user and application specific requirements. To help user in workflow construction it is appropriate to display notes and suggestions entered by the same or different users. Thus experts can collaborate and fill up application specific knowledge base with useful knowledge, which is shown to users in the right time. The solution was used and evaluated in the Pellucid IST project and it is further developed in the K-Wf Grid IST project.

1 Introduction

The experience management solutions are focused on knowledge sharing and collaboration among users in organizations. A lot of companies have recognized knowledge and experience as the most valuable assets in their organization. Experience is something that is owned by people only, not obtainable by computer systems. Anyhow, according to the state of the art in the area we can create experience management system, which will manage (not create) experience and will be able to capture, share and evaluate experience among users. We can understand experience through text notes entered by a user. Such form of experience is the most understandable for humans, but it can be grasped by a computer system, though only partially. A computer system needs to return experience in a relevant context. Thus we need to model the context of the environment and capture and store the context of each entered note. In this paper we describe such solution for the experience management based on text notes entered by users.

The key idea is that a user enters notes in a particular situation/context, which can be detected by the computer. Such notes are later displayed to other or the same users in a similar situation/context. The context of a user can be detected from many sources - received emails, a step in a business process or a workflow management system, used files or detection of other events performed in the computer. Not entire detected context is relevant to the entered note and the system with user's assistance needs to detect a relevant context based on the text of the note. The detection of the context is based on techniques such as indexing, semantic annotation or similarity of cases. In addition, the solution uses ranking and voting mechanisms for updating relevance of the text notes. Not the entire EMBET solution is described in this article in detail. The focus is on the context detection from a new entered text note. The solution was used and evaluated in the Pellucid IST project [1] and it is further developed in the K-Wf Grid IST project [2].

The main objective of the solution is to provide a simple and powerful experience management infrastructure, which can be applicable in many areas with users sharing and collaborating through experience. The idea is to return experience to users when they need it. Therefore it is crucially important to model and capture a user context and the described solution can be used only in applications where actions/tasks performed by a user are computerized and can be captured and reported to the system in the form of events.

The EMBET system can be applied in a different application. In scope of this article we focus on its use to support grid applications particularly contraction of grid workflows. When constructing a workflow from grid services, a user needs to have knowledge about problems, services, models or input and output data. Such knowledge can be formalized only partially and workflows solving a user problem can be composed only semi-automatically with user help. Experience/notes entered by experts can help users to create the best workflow for their needs.

The article first discusses a theoretical approach of the EMBET solution and its architecture, followed by examples given for the Flood Forecasting grid application.

2 Theory of the Approach

According to Bergman [3], the experience management is simply the capability to collect lessons from the past associated to cases. We have a person who has a problem p , which is described in a certain space, called the Problem Space (P). In the experience Management system we have Case-Lesson pairs (c, l) , where we have a Case Space (C) and a lesson space (L). We could have a single multidimensional vector in which we distinguish a case part and a lesson part. To be able to collect a lesson we must first devise a problem transformation function that maps problem space to case space.

$$c = f(p) \tag{1}$$

This function should be mono valued. For the simplest cases this function is the identity function, because the developer of the experience management system (EMS) has chosen to characterize the problem with the same attributes of the case. But, even in

this situation, the important fact is that we are faced with a semantic bridge. We do not store problems (which are infinite and cannot be predicted) but cases, which are a formal representation of problems solved in the past, or situations which have happened in the past.

To be able to re-use a particular lesson we should:

1. Characterize a problem (which of course is a separate process, that requires effort)
2. Transform the problem from the space P to the space C.
3. Choose from the cases the most "useful" lesson from the case-lesson pairs stored in the database
4. Apply that lesson.

The point "Choose" introduces the function of utility that, for each problem (transformed) outputs a number, which states the utility of the use of that lesson for the problem. Unfortunately this function is not known in advance, but could be verified only AFTER the lesson is applied to the problem at hand. To overcome this, Bergman introduces the similarity function and postulates that the most useful lesson will come from the most similar case. The problem, of course, is to develop this similarity function, which is quite a complex problem.

In EMBET the described steps are recognized as:

1. User context detection from the environment which describes problem P
2. Our Model is described by ontology and Notes are stored with an associated context, which describes space C
3. Notes represent learned lesson L which is associated with space C (note context). The note context is matched with a user problem described by the detected user context. The user context is wider than the note context and as a result all applicable notes are matched and returned.
4. Applying the lesson is left to the user by reading appropriate notes.

For context modelling we are using CommonKADS [4] methodology. We are able to model and detect context when the application domain is well modelled.

The role of EMBET is to assist the user in relevant knowledge/suggestions, which are applicable to his/her current situation. This is often called "the experience management". Thus EMBET does not create but only manage experience and is able to capture, share and evaluate experience among users of the application.

In EMBET experience is understood through notes entered by the user. Such form of experience is the most understandable for humans, but it can be partially understood by a computer system. A computer system needs to return experience in a context where experience is relevant. Thus we need to model a context of environment and capture and store a context of each entered note. From the text of the note we need to detect what kind of current context of a user is important for the note and what kind is not. We can do this with user's assistance, where the user will get pre-checked list of the detected context and will change it or submit the detected context unchanged. When the context is properly assigned, the note can be displayed to the same or other users in the same or similar context.

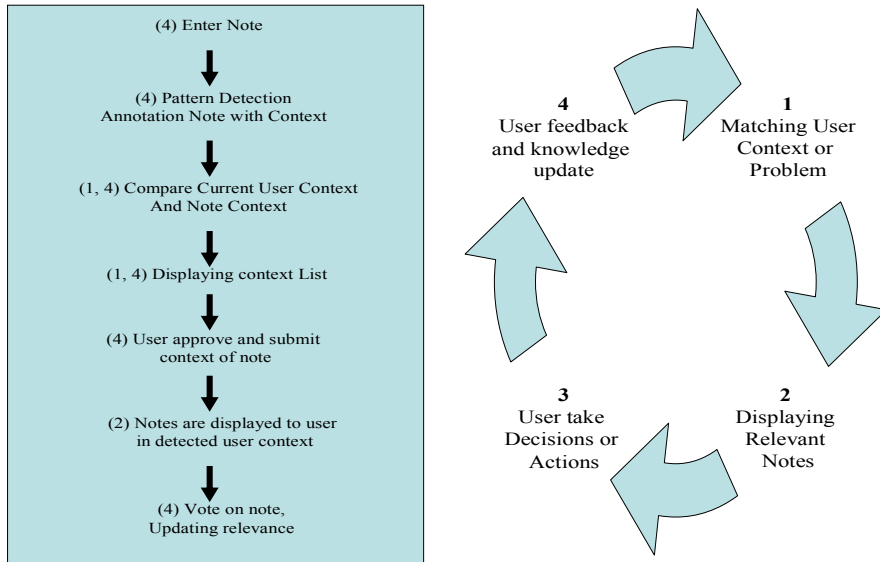


Fig. 1. Experience Management Cycle of EMBET

For the proper experience or knowledge management we need to have a closed knowledge cycle (see Figure 1, on right side). The most crucial point in experience and knowledge management systems is step 4 – “User feedback on knowledge and knowledge update”. In this article we describe mainly this step as solved in EMBET system, which can be seen also in Figure 1 on the left side. Updating of knowledge and experience in EMBET consists of the following steps:

- A user submits a Note, if s/he thinks s/he has relevant knowledge for the current or past situation
- The text of the note is processed and patterns of a semantic annotation are matched and notes are annotated with knowledge concepts. Context concepts are detected.
- Compare the current user context with the Note detected context.
- Displaying a Context List to a user with pre-selected context items detected in the Note.
- The user approves the context and submits the context of the note.
- The notes can be later displayed to users in a similar detected user context
- Users can Vote on Note relevance. The relevance of the note is then updated. If a user does not get relevant knowledge in any situation, and s/he gathers experience, s/he may enter a new note with his/her learned lesson.

The closed knowledge cycle means that experience is managed in the system the way that system together with users controls all phrases: capture capitalization and reuse of information and knowledge and that all the phrases are closed in the system and manages by a regular user of the system. The knowledge management systems often allow a user to see or search for knowledge but introduction of knowledge and the user feedback on knowledge is complicated and can be managed by knowledge experts only.

3 Architecture and Technology

Architecture of EMBET consists of 3 main elements:

- Core of the system
- Graphical User Interface (GUI)
- System Memory

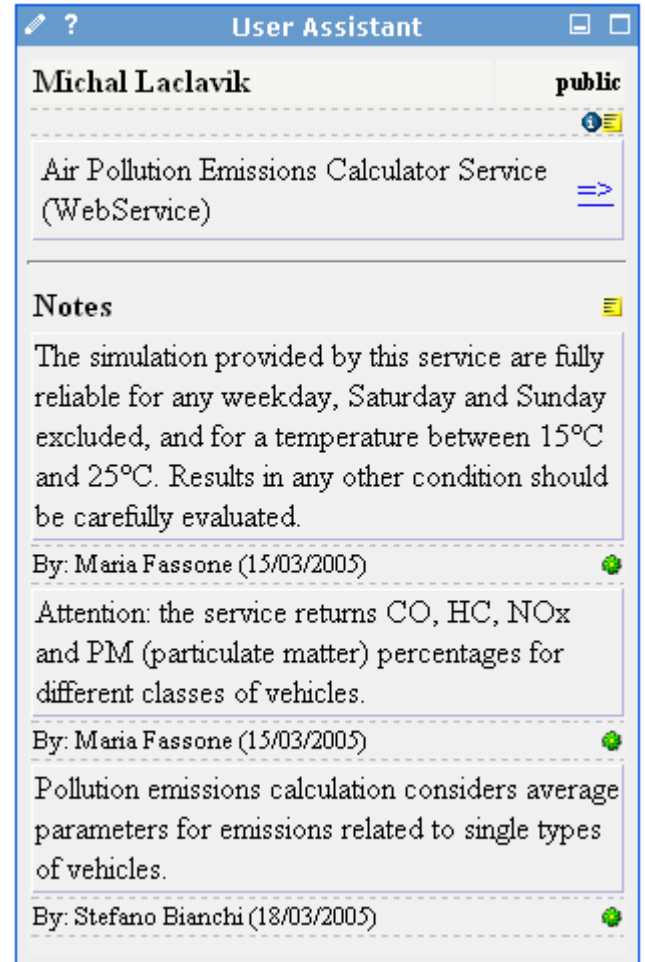
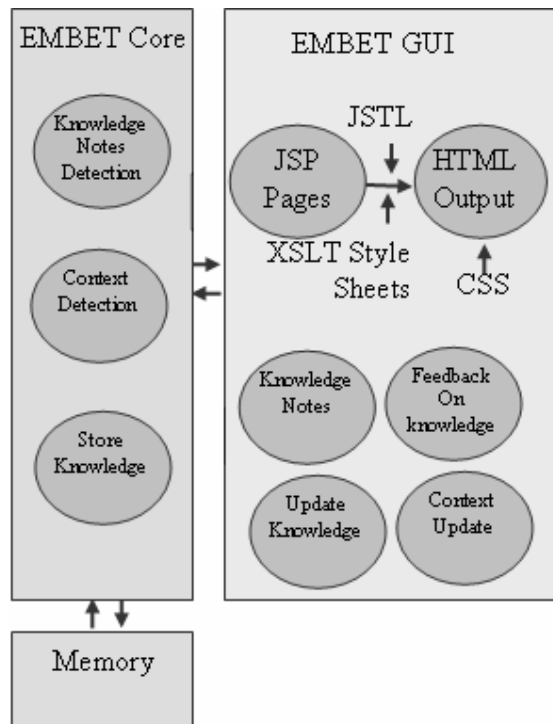


Fig. 2. EMBET Architecture and Graphical User Interface

EMBET Core provides the main functionality of EMBET. It determines a User context and searches for the best knowledge (in a form of text notes) in its Memory. The knowledge is subsequently sent through XML-RPC [5] or SOAP to EMBET GUI. When a user enters a note, the EMBET Core processes the note and determines the context of it from the current user's context and the context detected in the note. When the user confirms the context, the EMBET Core stores the note and user's feedback. The core also handles a user the state (context).

EMBET GUI visualises the knowledge and the user's context information to the user. Furthermore it informs the EMBET core about user context changes. The context can be reported also directly to the EMBET core from external systems (e.g.

from workflow systems, received emails, or file system monitors). EMBET GUI visualises knowledge based on XML [6] transformation to HTML through XSL [7] Templates processing. Moreover EMBET GUI has an infrastructure for a note submission and context visualisation. It further provides a user with feedback (voting) on knowledge relevance. In addition it contains a user interface for knowledge management by experts where an expert can change a note and its context.

EMBET Core - EMBET GUI interface is used for an XML data exchange between EMBET Core and EMBET GUI. The Interface will be based on the SOAP protocol where both components act as web services; currently we use the XML-RPC protocol for an XML message exchange.

Interface to Memory is used for information and knowledge extraction and storage. It is based on RDF/OWL data manipulation using Jena API, which EMBET Core uses to extract and store knowledge.

Experience is represented by text notes, an unstructured text, entered by a user. For the context and environment modeling we use ontology, thus we use a Protégé ontology editor for ontology based modeling. Ontology is stored and managed in the Web Ontology Language (OWL) [8]. The Jena Semantic Web Library [9] is used for knowledge manipulation and knowledge storing. The Java technology is used for developing the system and user Interface is based on the JSP technology. The XSL templates are used to transform XML generated from OWL to displayed HTML. Since the Java technology is chosen as background for the EMBET, a choice of the web server – Jakarta Tomcat [10] and implementation middleware is reasonable. XML is a widely used language for web development; it is regarded as a universal format for structured documents and data on the Web. Therefore in EMBET XML is used in various forms/degrees from description of ontology to the communication content language. The XML/XSLT technology permits visualization of XML documents and display of information presented to the user by EMBET. The JSTL[11] is a native Java library for XML processing.

5 Example of Use

To better illustrate the use of EMBET in the process of user assistance, we present the following example from the K-Wf Grid project's flood forecasting application, which extends the flood prediction application of the CROSSGRID (IST-2001-32243) [12] project. The application's main core consists of simulation models series for meteorological, hydrological and hydraulic predictions. The models are organized in a cascade, with each step of the cascade being able to employ more than one model. For example, the first step - the meteorological weather prediction - can be computed using the ALADIN model, or the MM5 model.

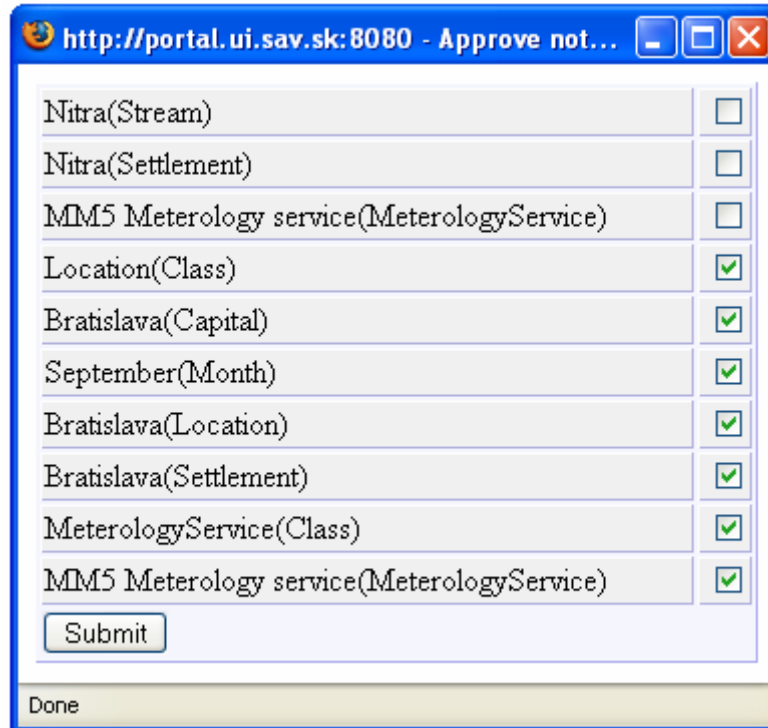


Fig. 3. Note Context Detection: Unchecked items are current user context; checked items are elements detected from text of the note. User selects only those items which are relevant.

Consider that the user has used the MM5 meteorology model and he/she wants to describe its properties (gathered knowledge), which may be relevant for other users. The proposed model of interaction is as follows:

- A user enters a note through EMBET, stating that “the MM5 model is not appropriate for weather forecast in September for Bratislava because it gives results which differ approx. 50% from reality”.
- From the workflow in which the user states this note, we know directly the current user context (unchecked items on Figure 3)
- Some of current context can be relevant to note and some does not have to be. The note is processed and its text related to the context, as well as the relevant context items are selected (figure 3). In this case, by finding the text MM5 we can assume that “Service type MM5” is the relevant part of the context. There is other context relevant information which can be detected like “September”, the time in which this note is valid.
- After the context detection, the user is presented with a checklist (Figure 3) where the user may select only the relevant parts of the context, which will trigger this note.

- A user gets pre-selected parts of the context, which were detected by the system as really relevant. He/she can subsequently modify the contents of the list and finally submit the note.
- Each time anyone works with the MM5 service for Bratislava area in September, the note is displayed.
- Each note can be voted by a user as being “good” or “bad” and the current results are always displayed along with the vote.

This model gives a good basis for experience management and knowledge sharing in a virtual organization as well as for application-related collaboration among users.

6 Conclusion and Future Work

Our solution was evaluated on a selected administration application in the Pellucid IST project [1], where the context or the problem of a user was detected in the Workflow Management Application. Currently we are applying this solution in the K-Wf Grid IST project [2], focused on building grid-based workflows, where users need to collaborate and share knowledge about different applications, computational models, grid resources or data. In case of K-Wf Grid we detect a user problem from the grid environment. Such solution may be applied in many further areas where the user problem can be detected. Usually this is in any business process where actions are performed via a computer, e.g. workflow processes, document management, supply chain management or dynamic business processes where email communication is in use.

People like to enter notes or memos. This is a natural way of notifications for themselves or others to remind them of problems, activities or solutions. Therefore we think that such solution can be successfully applied and commercialized with good results.

In the K-Wf Grid [2] project grid services are semi-automatically composed to workflows which should solve a user problem. It was identified that even when services and input and output data are well semantically described, there is often no possibility to compose an appropriate workflow e.g. because of missing specific input data or fulfilment of a user and application specific requirements. To help user in workflow construction it is appropriate to display notes and suggestions entered by the same or different users. Thus experts can collaborate and fill up application specific knowledge base with useful knowledge which is shown to users in the right time.

Acknowledgments

This work is supported by the project K-Wf Grid, EU RTD IST FP6-511385, VEGA No. 2/3132/23: “Effective Tools and Mechanisms for Scaleable Virtual Organizations”, APVT-51-024604: “Research and development of a knowledge based system to support workflow management in organizations with administrative processes” and SPVV 1025/2004.

References

1. Pellucid Consortium: Pellucid IST Project Website, 2004, <http://www.sadiel.es/Europa/pellucid/>
2. K -Wf Grid Consortium: K -Wf Grid IST Project Website, 2005, <http://www.kwfgrid.net/>
3. Ralph Bergmann: Experience Management: Foundations, Development Methodology, and Internet-Based Applications (Lecture Notes in Artificial Intelligence), 2002
4. August Schreiber et al.: Knowledge Engineering and Management: the Common-KADS methodology, ISBN: 0262193000, 2000
5. XMLRPC.com : XML-RPC Home Page, 2005, <http://www.xmlrpc.com/>
6. W3C : Extensible Markup Language (XML), <http://www.w3.org/XML/>
7. W3C : XSL Transformations (XSLT), 2005, <http://www.w3.org/TR/xslt>
8. W3C: Web Ontology Language OWL, 2005, <http://www.w3.org/TR/owlfeatures/>
9. HP Labs and Open Source Community, Jena Semantic Web Library, 2004, <http://www.sf.net/>
10. Apache Open source Community: The Jakarta Site – Apache Tomcat, <http://jakarta.apache.org/tomcat/>
11. Sun: JavaServer Pages Standard Tag Library, 2005 <http://java.sun.com/products/jsp/jstl/>
12. Hluchy L., Tran V.D., Habala O., Simo B., Gatial E., Astalos J., Dobrucky M.: Flood Forecasting in CrossGrid project. In: Grid Computing, 2nd European Across Grids Conference, Nicosia, Cyprus, January 28-30, 2004, LNCS 3165, Springer-Verlag, 2004, pp. 51-60, ISSN 0302-9743, ISBN 3-540-22888-8