Cracow '06 Grid Workshop

October 15 – 18, 2006
Cracow, Poland

K-Wf Grid

The Knowledge-based Workflow System for Grid Applications

Proceedings

Editors: Marian Bubak
Steffen Unger

Published in July 2007

by Academic Computer Centre CYFRONET AGH
ul. Nawojki 11, 30-950 Kraków 61, P.O. Box 386, Poland

ISBN 978-83-915141-8-4

Cover design and book typesetting by Milena Zając.
The Editors did not interfere with the contents of material submitted by the Authors.

Cracow '06 Grid Workshop

Organizers

Academic Computer Centre CYFRONET AGH
Institute of Nuclear Physics PAN
Institute of Computer Science, AGH

Steering Committee

Marian Bubak	ACC CYFRONET AGH, Cracow
Michał Turała	IFJ PAN / ACC CYFRONET AGH, Cracow
Kazimierz Wiatr	ICS / ACC CYFRONET AGH, Cracow
Piotr Bała	N. Copernicus University, Toruń / ICM, Warsaw
Jarek Nabrzyski	PSNC, Poznań
Roman Wyrzykowski	University of Technology, Częstochowa

Organizing Committee

Kazimierz Wiatr	ACC CYFRONET AGH
Michał Turała	IFJ PAN / ACC CYFRONET AGH
Marian Bubak	ICS / ACC CYFRONET AGH

Sponsors

CYFRONET
KATEDRA INFORMATYKI
hp invent
intel

IST-FP6-511385 K-WfGrid

Knowledge-based Workflow System for Grid Applications

http://www.kwfgrid.eu

CONSORTIUM

Fraunhofer Institute for Computer Architecture and Software Technology
(FIRST)

http://www.first.fraunhofer.de

Institute for Computer Science, University of Innsbruck
(UIBK)

http://informatik.uibk.ac.at

Institute of Informatics of the Slovak Academy of Sciences
(II SAS)

http://ui.sav.sk

Academic Computer Centre of the AGH University of Science and Technology
(CYFRONET)

http://www.cyf-kr.edu.pl/en/

SINGULAR LOGIC Information Systems & Applications S.A.

http://www.singularlogic.eu

Softeco Sismat S.p.A.

http://www.softeco.it

Preface

The Sixth **Cracow Grid Workshop** (CGW'06) was – as always – organized jointly by the Academic Computer Centre CYFRONET AGH, the Institute of Nuclear Physics PAN and the Institute of Computer Science AGH. CGW'06 was held in Cracow on the premises of IFJ PAN from 15 to 18 October 2006. The main objective of the Cracow Grid Workshop is to create and support a collaborative community of researchers, developers, practitioners and potential users in Poland and in the neighbouring countries who work in the fascinating field of Grid systems and their applications.

The CGW'06 was organized as usual as the Central European Grid Consortium (CEGC) event, and comprised: keynote and invited lectures, poster sessions, oral presentations and a tutorial. About 160 participants from 20 countries took part in the Workshop. For ACC CYFRONET AGH, the CGW'06 was also part of its activity within the framework of the many Grid projects: EGEE II, K-WfGrid, CoreGRID, ViroLab, int.eu.grid, GREDIA, BalticGrid.

The following keynote speakers honoured the event:

- Wolfgang Boch, Head of F2, Information Society and Media DG, European Commission,
- Fabrizio Gagliardi, Microsoft EMEA and LATAM Director for Technical Computing, Switzerland,
- Wolfgang Gentzsch, Coordinator of D-Grid, Germany,
- Carl Kesselman, University of Southern California, Information Sciences Institute, USA,
- Dieter Kranzlmüller, Johannes Kepler University Linz, Austria,
- Thierry Priol, INRIA, France,
- Peter Sloot, Section Computational Science, University of Amsterdam, The Netherlands.

Invited speakers of the Workshop included:

- Piotr Bała – representative of ICM Warsaw,
- Jarek Nabrzyski from PSNC Poznań,
- Roman Wyrzykowski from the University of Technology in Częstochowa,
- Paweł Pisarczyk representing ATM. S.A.

The tutorial *Principles and Practices of Grid Security* was given by Syed Naqvi, Research Fellow CoreGRID (CETIC Belgium, CCLRC UK).

More than 100 contributed papers, accepted for presentation during the Workshop, provided a very good overview of the research activity in the area of Grid computing. The contributed papers were reviewed by the Steering Committee upon submission of abstracts, then during their presentations at the event, and, finally, after submission of full versions. These papers give a very good overview of the research activity in the following fields of Grid computing:

- grid projects,
- workflows,

- semantics in the grid systems,
- resource management,
- monitoring,
- data management,
- Grid middleware,
- security,
- software development,
- applications.

The Proceedings of CGW'06 are splitted into two volumes; in this second one we have collected all papers submitted by the EU IST Project *K-WfGrid – Knowledge-based Workflow System for Grid Applications*.

We would like to express our gratitude to Prof. Marek Jeżabek, Director of the Institute of Nuclear Physics, and Prof. Krzysztof Zieliński, Director of the Institute of Computer Science AGH for their help and personal involvement. We owe thanks to the Workshop sponsors: Intel, HP and ATM S.A. Poland for generous support.

We are also indebted to all the members of the Workshop Secretariat (Zofia Mosurska, Maria Stawiarska, Milena Zając) and other colleagues from ACC CYFRONET AGH (Teresa Ozga, Mieczysław Pilipczuk, Andrzej Ozięblo) for organizing the event. Special thanks go to Milena Zając for her hard work on preparation of the Proceedings for printing.

We kindly invite you to visit the Web page of the Cracow Grid Workshop (`http://www.cyfronet.krakow.pl/cgw06/`) and, of course, to participate in CGW'07 in October 2007.

Cracow, July 2007

Marian Bubak
Michał Turała
Kazimierz Wiatr

Table of Contents

K-WfGrid – Knowledge-Based Workflow System for Grid Applications

Marian Bubak[1,2], Piotr Nowakowski[2], and Steffen Unger[3]

[1] Institute of Computer Science AGH,
al. Mickiewicza 30, PL-30-059 Kraków, Poland
[2] Academic Computer Centre CYFRONET AGH,
ul. Nawojki 11, 30-950 Kraków, Poland
[3] Fraunhofer Institute for Computer Architecture and Software Technology (FIRST),
Kekulestr. 7, D-12489 Berlin, Germany

Abstract

The IST FP6-511385 STREP Project K-WfGrid "Knowledge-based Workflow System for Grid Applications" addresses the need for a better infrastructure for the future Grid environment. The Grid as a vast space of partially-cooperating, partially-competing Grid services will be a very dynamic and complex environment. In order to address the complexity in using and controlling the next generation Grid, the Consortium adopted the approaches envisioned by semantic Web and Grid communities in a novel, generic infrastructure.

The K-WfGrid system assists its users in composing powerful Grid workflows by means of a rule-based expert system. All interactions with the Grid environment are monitored and evaluated. Knowledge about the Grid itself is mined and reused in the process of workflow construction, service selection and Grid behaviour prediction. Workflows are dynamic and fault-tolerant beyond the current state of the art. The K-WfGrid system is generic by providing domain-independent system components, freeing the user from the burden of complex Grid usage and maintenance. Specific application knowledge is added in a customization phase carried out by system integrators including SMEs. This generality was demonstrated by applying K-WfGrid in three different application domains regarding scientific simulations (flood forecasting simulation) as well as industrial applications (ERP and traffic management).

The project started in September 2004 and concluded in February 2007.

Keywords: Grid, workflow system, knowledge, ontologies, monitoring, performance analysis, pilot applications

1 Motivation

The main challenge for the Knowledge-based Workflow System for Grid Applications (K-WfGrid) [1] is to enable the knowledge-based support of workflow construction and execution in a Grid computing environment.

In order to achieve this objective the Consortium developed a system that enables users to:

- semi-automatically compose a workflow of Grid services,
- execute the composed workflow application in a Grid computing environment,
- monitor the performance of the Grid infrastructure and the Grid applications,
- analyze the resulting monitoring information,
- capture the knowledge that is contained in the information by means of intelligent agents,
- and finally to reuse the joined knowledge gathered from all participating users in a collaborative way in order to efficiently construct workflows for new Grid applications.

2 K-WfGrid Architecture

K-WfGrid undertook research using the latest achievements in Grid and semantic web technologies to facilitate the dynamic construction and execution of knowledge-based workflows in Grid environments.

K-WfGrid defines "workflow" as "the automation of distributed IT processes – in whole or part – during which documents, information or tasks are passed from one participant to another for action, according to semantic description of available resources and information gathered by their monitoring".

K-WfGrid developed technology to compose Grid workflows automatically on the basis of the desired output and of the description of Grid resources available: services are invoked in the correct order and fed the correct data so that the results are compliant with the specification and that only the most optimal resources are used.

The architecture (Fig. 1) is composed of four horizontal layers and the vertical knowledge layer. Each horizontal layer communicates only with the proximate layers so that single layers can be modified or replaced without changing the whole system [1].

3 Workflow Orchestration and Execution Environment

The main innovation of the Workflow Orchestration and Execution Environment [2] is the multi-level approach to application workflow modelling. The idea is to introduce several layers of abstraction for a workflow and to provide the end user with a set of tools that help to compose, refine and execute such workflow. While the tools are usually dedicated to operate on a specified abstraction level, the language that models the workflow structure and behaviour on every defined level is the main integrating element of the Environment. The language is based on the idea of High-Level Petri Nets (HLPN) and expresses additional information regarding its elements through colour code. As all workflow abstraction levels are described by the same description language it is possible that one single workflow contains abstract as well as concrete parts at the same time. This

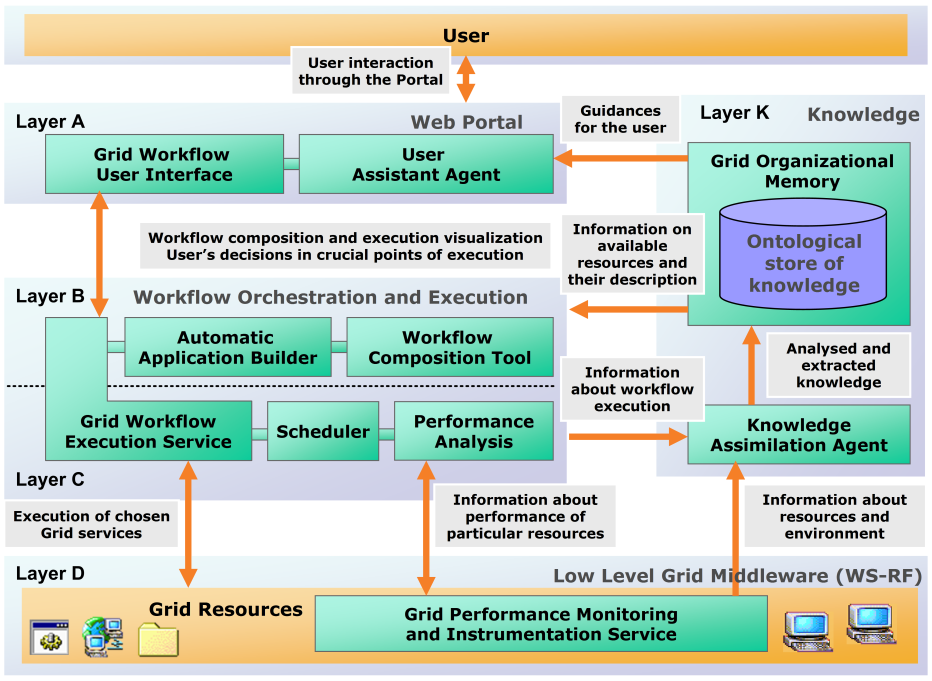

User
User interaction through the Portal
Layer A
Web Portal
Grid Workflow User Interface
User Assistant Agent
Guidances for the user
Layer K
Knowledge
Grid Organizational Memory
Ontological store of knowledge
Workflow composition and execution visualization
User's decisions in crucial points of execution
Information on available resources and their description
Layer B
Workflow Orchestration and Execution
Automatic Application Builder
Workflow Composition Tool
Analysed and extracted knowledge
Information about workflow execution
Grid Workflow Execution Service
Scheduler
Performance Analysis
Knowledge Assimilation Agent
Layer C
Execution of chosen Grid services
Information about performance of particular resources
Information about resources and environment
Layer D
Low Level Grid Middleware (WS-RF)
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 1: K-WfGrid layered architecture.

enables the workflow orchestration system to be highly dynamic and to react on changes in an unreliable Grid environment.

Thus, the main innovations of the K-WfGrid workflow management technology are: support of dynamic workflows, interaction between the workflow orchestration and execution process, application of an expert system for supporting knowledge-based workflow orchestration and Grid Service selection and support of several levels of abstraction within one Grid workflow description language.

4 Workflow Composition Tool (WCT)

In the context of the K-WfGrid framework, WCT [3] provides Grid Workflow Execution Service (GWES) with the ability to refine initial, very raw workflows into abstract but complete solutions that may be further concretised and finally executed/scheduled.

The tool contacts the ontological registry (Grid Organization Memory) in order to find suitable service operations that may produce the required result. The services are described in an ontological form with statements regarding the service operations' inputs, outputs, preconditions and effects (the IOPE set). Through these notions the tool that composes workflows is able to match different operations into a workflow.

By associating the required data with the produced output the tool constructs a data flow between operations containing abstract, domain-specific classes of services. As those operation types are meaningful for a domain expert user, this level of abstraction of a workflow ensures that the application logic is valid. It, however, lacks any implementation and instance-specific details so it is invulnerable to the frequent changes in the Grid environment and allows for the workflow reuse capability.

5 Automatic Application Builder (AAB)

AAB [4] is supported by Semantic Information in order to automatically convert abstract workflows into real ones (abstract workflows specify only a recipe for providing the expected solution; the real workflow consists of service instances deployed in the Grid).

AAB exploits the knowledge about the Grid world gathered in GOM for this automatic construction. It is integrated with the workflow editor used for passing user preferences specified trough User Interface. These user preferences can be exploited during the service selection. The AAB is deployed as web service and the implementation of the AAB client uses web services.

6 Scheduler

The K-WfGrid scheduler [5] is a performance-oriented workflow Grid scheduler, which makes workflows concrete and prepares them for execution. It recognizes the Petri nets model by which the workflows are represented, and creates a schedule only for those workflow nodes which have been prepared for scheduling by other services (only for “blue” nodes, in terms of K-WfGrid). Since the scheduler is aware of the current workflow execution status, it can be applied dynamically many times during a single workflow execution (dynamic scheduling model).

The scheduler uses the information acquired from the Monitoring Service, concerning the infrastructure as well as the service execution. Monitoring information is used to choose the most appropriate resources for the execution of the workflow. It will also use knowledge-based performance predictions to approximate the execution time of services and therefore to improve the scheduling.

It shows that the information collected and processed by the services of K-WfGrid can be applied in practice as a positive feedback to enhance the performance of the environment.

7 Distributed Performance Analysis Service (DIPAS)

DIPAS [6] helps the user and the developer to understand the performance behaviours of their Grid workflows and infrastructure and to find the cause of failures and performance problems by analyzing Grid services and resources and their complex dependencies at runtime. It includes a set of novel techniques for supporting workflow overheads analysis, performance visualization and performance search on the fly. DIPAS provides a performance analysis service and a Web portal for conducting the performance monitoring and analysis of Grid workflows and infrastructure. Moreover, the middleware services also benefit from using DIPAS as it provides performance knowledge for constructing and executing workflows.

The main achievements are:

- the design and implementation of a comprehensive, unified, extensible performance analysis of Web services-based workflows,
- novel request representations alleviate the interaction between performance services and their clients,
- performance of Grid workflows is systematically analyzed according to a classification of workflow overheads, and performance problems can be detected online by interpreting performance metrics at runtime.

8 Generic Monitoring Infrastructure (GEMINI)

GEMINI [7] is a generic monitoring system that supports online monitoring of Grid infrastructure and applications. Part of GEMINI is a sensor infrastructure which allows for easy dynamic deployment of new data sources into the monitoring infrastructure. In contrast to available solutions, GEMINI focuses on a comprehensive support for on-line monitoring of Grid workflows including semi-dynamic instrumentation.

Users of GEMINI interact with it through two interfaces exposed as web services:

- monitoring interface based on PDQS (Performance Data Query Subscribe) language,
- instrumentation interface based on WIRL (Workflow Instrumentation Request Language).

Through the monitoring interface, one can obtain monitoring data by querying it to obtain results immediately or subscribing it to receive a data stream for a specified period of time. The instrumentation interface is used to control the instrumentation of applications, which is a necessary step before monitoring of applications is possible. Through this interface, one can request a high-level abstract representation of the application called SIRWF (Standard Intermediate Representation for Workflows) which allows identifying the parts of applications to be instrumented and specify the corresponding metrics (e.g. start/end events). Once code regions and metrics are identified, a request to enable or disable the instrumentation can be issued. With the described instrumentation approach, various heterogeneous parts of a workflow (services based on different programming languages, code regions, invoked legacy codes) are viewed in a common way, while underlying different instrumentation systems, needed to deal with the different workflow parts, are integrated and hidden behind common interfaces.

9 Grid Organizational Memory (GOM)

Grid Organizational Memory [8] is a distributed knowledge base designed to manage semantic metadata about all kinds of resources available in a Grid environment including the application domains to which the Grid is applied.

GOM uses a set of defined generic ontologies in order to provide common understanding for both users and other middleware components and allows application developers to extend these ontologies with their domain specific concepts and thus provide additional meaning to the workflow elements [9]. Availability of rich semantic descriptions of such elements as data or services is of crucial importance for the workflow composition components.

The main achievements are:

- the definition of the unified semantic metadata model of the Grid called ontology separation scheme,
- design and development of a generic distributed knowledge base [10] with an event system enabling updating of managed ontologies and recovery of the state of the knowledge base from any given time in the past from serialized history of incoming events,
- development of Protege plug-in that enable easy interaction with the knowledge base.

GOM along with its accompanying tools (e.g., tool for semantic description of available services or Graphical User Interface access to GOM) [11,12] provides versatile functionality for deploying and managing ontological knowledge bases in any Grid environment. The easy extensibility and multiple configuration options of the knowledge base provide also high adaptability of the system to various settings and applications.

10 Knowledge Assimilation Agent (KAA)

Knowledge Assimilation Agent (KAA) [13] is a knowledge-based component for Grid service workflows, which comprises three basic functionalities:

- assimilates run-time information about Grid services from different sources and produces performance estimations of future Grid service invocations (the KAA-Web Service),
- performs past workflow analysis and produces workflow result estimations (the KAA-WXA tool),
- discovers new potential services through interactive semi-automatic ontology alignment (the OnTal tool).

The main achievements include the ability of estimation of Grid service performance measures (run-time, availability, accessibility, etc.) based on the invocation parameters and input resource metadata and semi-automatic extension of the knowledge base by ontology evolution methods.

The main scientific innovations of KAA are extension of instance based learning (IBL) technique of WS performance prediction by enabling retrieval and inclusion of semantic resource properties into the feature vector, enabling specification of WS performance prediction profiles which specify feature vector and result to be estimated, and proposition of a novel ontology for IBL-based WS performance prediction together with methodology extending the classical Case-Based Reasoning. Development of knowledge extension by ontology evolution using a combination of lexical and structured similarity and considering neighbourhoods of two ontology concepts when their similarity is assessed is another achievement of KAA in the context of workflow construction for grid applications. Development of knowledge extension by ontology evolution using a combination of lexical and structured similarity and considering neighbourhoods of two ontology concepts when their similarity is assessed is another achievement of KAA in the context of workflow construction for grid applications.

11 User Assistant Agent (UAA)

The User Assistant Agent [14] helps users in collaboration and knowledge sharing in Grid workflow applications. 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. Also intelligent components in the grid middleware such as monitoring, workflow analysis or workflow composition can provide context sensitive notes to be displayed for the user. In the K-WfGrid, 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 requirement. To help user in workflow construction, problem specification or knowledge reuse from past runs it is appropriate to display notes and suggestions entered by users or intelligent middleware components. Thus experts can collaborate and fill up application specific knowledge base with useful knowledge, which is shown to users at the right time.

12 User Environment Interface

The User Environment Interface (UI) [15] is the main access point for K-WfGrid, which is available as a web portal, reachable from anywhere in the world using the Internet. It is a set of portable JSR-168 portlets, deployed in a standard portlet container. These portlets access specific parts of K-WfGrid middleware, grid data storage, and application components. Together, they allow users to:

- create and manage application workflows

- access, view, and modify knowledge available to K-WfGrid components
- upload data files into the grid, and download them from the grid
- access, and create new text notes, and manage user experience
- debug K-WfGrid and application components using integrated log viewer
- communicate with each other
- view, and reuse recent application workflows.

The UI is easily extensible with additional portlets, and all existing portlets may be arranged into different sets, targeting different user groups. Also, it enables user collaboration in execution of application workflows, as well as in the task of application development, debugging, and deployment. It is a central K-WfGrid-wide integration point.

13 The Pilot Applications

In the context of the K-WfGrid project, pilot applications are testbeds meant to verify the possibility to apply the solutions developed in some real contexts and to evaluate the impact and the value added provided by the middleware. In this sense, they can be assumed as proof-of-concepts in different scenarios, with the aim at verifying the applicability of a research-oriented approach in actual applications. As well as for technical test, pilot applications constitute pre-competitive solution examples to real problems and are useful for an early detection of problems related to future exploitation, e.g. how a K-WfGrid-based solution can be designed, which requirements are related to the adoption of the middleware, which constraints this implies etc.

The **Flood Forecasting Simulation Cascade** [16] predicts flooding scenarios using a series of meteorological, hydrological and hydraulic simulations. In the following, a generic problem is illustrated, and then solutions by standard Grid system and by the K-WfGrid system are compared.

A national emergency management agency tries to evaluate possible risks of flooding of a certain area, as weather worsens and heavy rain continues for several days in large part of the country. The emergency management agency is working closely with one or more meteorological institutes, and also with local river authorities. Each of these organizations has its own set of machines for simulation of their respective problems – weather prediction, hydrology, or hydraulics. Each of these organizations has also their experts who work with the sophisticated programs that run these simulations, and who are able to utilize them in optimal way.

The standard solution for this problem is to construct a complicated virtual organization (VO) structure for this effort, and to maintain this VO constantly. In order to do this, a coordination centre has to be selected, which will manage the VO. A management hierarchy is designed, with a board of appointed directors for each participating institution. A separate, dedicated security and certification body is appointed with guarding the VO's resources. Each participant of the VO dedicates certain volume of computation and storage resources, based on their level of participation in the VO, and on their abilities. This machinery is constantly maintained, upgraded, augmented, evaluated, monitored, updated, observed, and guarded.

With K-WfGrid system the level of integration may be lower, without compromising the efficiency of the system. This is mainly due to the sophisticated semantics-based knowledge management system, which stores perceived events and is able to learn from them, or at least bring them up to the user when they may guide him/her. Instead of constantly maintaining the system in semi-ready order, the whole infrastructure can "power down", divert to different tasks, and "power up" quickly, without loss of efficiency, when needed. Also, the used web and grid standards make it easier to add yet more services to the system. This system allows simple incorporation of accounting into the organization, so generic computation power providers, storage centers and communication networks may provide their services to the specialized data and code providers, to mutual benefit.

In the case of **Coordinated Traffic Management** [17] the adoption of K-WfGrid in a public organization – such as the Mobility Department of the Municipality of Genoa – was investigated to point out which possible advantages a K-WfGrid-based system could offer in support of or even as a valid alternative to the current solutions (provided that Service Oriented Architecture and Grid web services are adopted as main technologies). The correct approach for a valuable evaluation of the K-WfGrid in the context of Coordinated Traffic Management moved therefore from "what can be done in this context now with K-WfGrid (that was undoable before)", to "what will K-WfGrid allow in this context once a Grid (or SOA) architecture will be adopted". K-WfGrid in fact enables the adoption and effective exploitation on top of other technologies, whose adoption is therefore a necessary requirement.

Enterprise Resource Planning [18] as current enterprise environment is characterized by rapid changes and fuzzy networks of inter-enterprise correlations and dependencies, the effective decision making, which constitutes crucial factors concerning the company positioning and competitiveness in the enterprise environment, requires computation- and data- intensive tasks due to the complexity of the supporting algorithms and the massive storage of enterprise data. In addition, the enterprise employees, who are the real ERP users, demand fast access to machine processed enterprise data so as to support efficiently and execute effectively back-office business processes, leading to the faster delivery of vital information, to the optimization of the enterprise performance and to the enhancement of customers' satisfaction.

K-WfGrid approach and platform a) allows semantically-assisted search, discovery and selection of appropriate tasks/services, stored in services repository that could typically contain some hundreds of services, with their desired functionality in order to compose a business-driven e-workflow and to establish connections among these tasks (control and data flow); b) allows the user to identify, at design time, the operational metrics of discovered services and grid resources, including timeliness, quality of products delivered, cost of service, and reliability, facilitating, thus, the composition of optimized grid-enabled e-workflows;

c) supports the semi-automated generation, storage, and reuse of business e-workflows with a level of parameterization; d) shortens calculation time of data- and computation- intensive business-driven ERP-based processes, and improves enterprise competitiveness delivering instantly high quality computations; and e) enables simple ERP users, with no experience in grid computing and workflow management, to benefit from the emerging Grid-enabled Service Oriented Computing by offering user-friendly interfaces to a complex Grid-based environment.

14 Achievements

The main scientific and technical achievements of the K-WfGrid project are the following:

- A multi-level approach to application workflow modeling based on introduction of several layers of abstraction for a workflow,
- Domain-oriented semantic descriptions of basic organizational resources, applied to automatic construction of complex Grid application workflows,
- A scheduler for Grid application workflows, able to select between semantically equivalent services,
- A generic monitoring framework that integrates various types of monitoring information described in a common data representation and available through a standardized monitoring interface, particularly useful for monitoring of Grid workflows,
- A set of novel techniques for online performance execution tracing, workflow performance overhead analysis and search for performance problems of Grid workflows at multiple levels of abstraction,
- A framework for reuse of past knowledge for future workflow construction,
- An architecture and knowledge model for collaboration and knowledge sharing in a context sensitive environment, including advanced methods for user context and knowledge context matching,
- An abstract ontological description of the functionality of a Grid environment, expressed as the Grid Organizational Memory,
- A suite of end-user tools, capable of supporting the above goals.

By using K-WfGrid tools the user can construct and execute application workflows according to recommendations offered by a dedicated UI, store workflow history and convert abstract workflows into real ones, monitor the performenace of the Grid through a monitoring and analysis system, reuse workflow-specific knowledge assimilated through the Grid Organizational Memory.

The scientific results of K-WfGrid have been published in about 100 papers in Future Generation of Computer Systems, Journal of Grid Computing, Concurrency and Computation: Practice and Experience, Computer and Informatics, Springer Lecture Notes in Computer Science, IEEE Computer Press, and in Proceedings of Cracow Grid Workshop.

15 Availability of Results

The K-WfGrid system is a connection of several interoperating modules [19]. Some of these modules may be operated separately, some are only supporting modules for others, and some need support of other modules to work correctly. The source code is freely available for non-commercial use. Several components require additional license agreements for commercial use. Project results are available for downloading under the respective section downloads of the project web site [1].

16 Partners

The **Fraunhofer Institute for Computer Architecture and Software Technology**, Berlin, is an institute of the Fraunhofer Gesellschaft, the leading German research organization for applied research that drives economic development and serves the wider benefit of the society. The institute coordinated the project, supported by the Fraunhofer contract and financial departments, and was responsible, in particular, for the design and implementation of the Grid Workflow Execution System.

The **Institute for Computer Science, University of Innsbruck**, is working at the cutting edge of performance monitoring, analysis and prediction. Its work in the K-WfGrid project concentrated on performance analysis, interfaces and data representation as well as scheduling for Grid workflow applications.

The **Institute of Informatics of the Slovak Academy of Sciences**, Bratislava, has extensive experience in scientific computing, knowledge description and management and interactive tool design and implementation. It was responsible for components performing knowledge gathering and components allowing its reuse. Also, it led the testbed construction and pilot applications implementation.

The core domain of work of **Academic Computer Centre CYFRONET of the AGH University of Science**, Cracow, is the Grid Organizational Memory, definition and implementation of ontologies to describe and enable reuse of the gathered information from the different sources and sensors. ACC CYFRONET was responsible for scientific coordination and participated also in the development of workflow orchestration and monitoring activities.

The two SMEs in the project were: **SingularLogic S.A.**, Athens, Greece, one of the largest IT software houses in Greece, and **Softeco Sismat S.p.A.**, Genova, Italy – a major supplier of software and automation systems for industry and research partners operating in complex environments with advanced IT methodologies.

Both acted as active contributors to the scientific development, in particular, in metadata and ontology domain and providers of pilot application examples, and led dissemination and exploitation activities.

References

1. K-WfGrid web site: `http://www.kwfgrid.eu`
2. A. Hoheisel; Grid Workflow Execution Service – Dynamic and Interactive Execution and Visualization of Distributed Workflows, Proc. CGW'06, vol.II, pp. 13-24
3. T. Gubała, D. Harezlak, M. Bubak, and M. Malawski; Constructing Abstract Workflows of Applications with Workflow Composition Tool, Proc. CGW'06, vol.II, pp. 25-30
4. L. Dutka, J. Kitowski, and S. Natanek; Automatic Application Builder – Tool for Automatic Service Selection, Proc. CGW'06, vol.II, pp. 31-38
5. M. Wieczorek, and T. Fahringer; K-WfGrid Scheduler – Scheduling Grid Workflows Based on Prediction and Performance Knowledge, Proc. CGW'06, vol.II, pp. 39-47
6. H.-L. Truong, T. Fahringer, P. Brunner, R. Samborski, and V. Nae; DIPAS: Performance Analysis and Visualization Support for Grid Workflows, Proc. CGW'06, vol.II, pp. 48-59
7. B. Balis, M. Bubak, and B. Łabno; GEMINI: Generic Monitoring Infrastructure for Grid Resources and Applications, Proc. CGW'06, vol.II, pp. 60-73
8. B. Kryza, J. Pieczykolan, M. Majewska, R. Slota, M. Babik, A. Toth, J. Kitowski, and L. Hluchy; Grid Organizational Memory – Semantic Framework for Metadata Management in the Grid, Proc. CGW'06, vol.II, pp. 74-81
9. M. Majewska, B. Kryza, and J. Kitowski; On Translation Common Information Model to OWL Ontology, Proc. CGW'06, vol.II, pp. 82-89
10. B. Kryza, J. Pieczykolan, J. Wach, M. Zuzek, and J. Kitowski; Peer-to-Peer Distribution Model for Grid Organizational Memory Knowledge Base, Proc. CGW'06, vol.II, pp. 90-95
11. M. Kuta, S. Polak, B. Palacz, T. Miłoś, R. Słota, and J. Kitowski; TeToN – a Jena-Based Tool for Text-to-Ontology Approach, Proc. CGW'06, vol.II, pp. 96-105
12. R. Slota, J. Zieba, M. Janusz, and J. Kitowski; OnTal – the Tool for Ontology Extension in Knowledge-Based Grid Systems, Proc. CGW'06, vol.II, pp. 106-113
13. Z. Balogh, M. Šeleng, M. Mališka, L. Hluchý, R. Slota, J. Zieba, and J. Kitowski; Knowledge Assimilation Agent – Component for Workflow-Based Grid Service Performance Prediction, Workflow Result Estimation and Semi-Autonomic Service Discovery, Proc. CGW'06, vol.II, pp. 114-121
14. M. Laclavík, M. Šeleng, and L. Hluchý; User Assistant Agent: Towards Collaboration and Knowledge Sharing in Grid Workflow Applications, Proc. CGW'06, vol.II, pp. 122-130
15. E. Gatial, B. Simo, G. Nguyen, M. Laclavik, L. Hluchý, T. Linden, and B. Kryza; K-Wf Grid Portal: a Web Interface to Semantic Workflows, Proc. CGW'06, vol.II, pp. 131-137
16. O. Habala, M. Mališka, and L. Hluchý; Service-Based Flood Forecasting Simulation Cascade in K-Wf Grid, Proc. CGW'06, vol.II, pp. 138-145
17. M. Masetti, S. Bianchi, and G. Viano; Application of K-Wf Grid Technology to Coordinated Traffic Management, Proc. CGW'06, vol.II, pp. 146-153
18. S. Pantelopoulos, and K. Kalaboukas; The Potentials of Knowledge-Based Workflows and Grid Infrastructures in an ERP Environment: The K-WfGrid Experience, Proc. CGW'06, vol.II, pp. 154-161
19. M. Bubak, M. Malawski, P. Nowakowski, and S. Unger; Software Engineering Aspects of K-WfGrid, Proc. CGW'06, vol.II, pp. 162-169

Grid Workflow Execution Service – Dynamic and Interactive Execution and Visualization of Distributed Workflows

Andreas Hoheisel

Fraunhofer Institute for Computer Architecture and Software Technology,
Berlin, Germany
email: `andreas.hoheisel@first.fraunhofer.de`

Abstract
The main objective of Grid workflows is the automation of complex IT processes in Grid environments. This article describes the Grid workflow execution environment of the K-Wf Grid project, which consists of several system components, such as the *Grid Workflow Execution Service*, the *Grid Workflow User Interface*, and the *Grid Workflow Description Language* that is based on high-level Petri Nets and which is able to model the data flow as well as the control flow within a single formalism. A unique feature of the solution is the completely virtualized resource allocation based on abstract modeling of the process structure and the powerful resource description formalism. The user is able to invoke workflows without any knowledge of underlying hardware and software specifics, allowing to concentrate on the real focus of the work.

1 Introduction

Executing complex processes in heterogeneous distributed computing environments is a major goal in applying Information and Communication Technologies in industry and science. While several tools are established in the market for modeling pure business processes, until now only few products support process execution in Grid environments. They are applicable mostly to restricted categories of processes. Often, they also require user knowledge about the underlying platforms and software components.

The *Grid Workflow Execution Service* (GWES) is the workflow enactment engine of the K-Wf Grid system [16], which coordinates the composition and execution process of Grid workflows. It implements a highly dynamic workflow concept [8], [14] by means of the *Grid Workflow Description Language* (GWorkflowDL) [1], [3], [4], which is based upon the theory of high-level Petri Nets [9]. It provides interfaces to the Web Portal and to a command line client for user interaction as well as to the low-level Grid Middleware for the invocation of application operations.

The main purpose of the *Grid Workflow Execution Service* and its client interfaces is to enable users to automatically execute workflows on distributed resources without being bothered with implementation-specific details, putting more attention to the functionality of the workflow itself. Therefore, the *Grid Workflow Execution Service* provides methods to initiate and analyze Grid workflows, and to coordinate and optimize the execution of these workflows on distributed and inhomogeneous resources regarding the control as well as the data flow. Abstract operations are automatically mapped onto matching software and hardware resources, triggering web service operations, remote executions of programs, or file transfers. The workflow service supports pure Web Services and Globus Toolkit 4 and is easily extendible for further execution platforms, such as UNICORE, Condor, GRIA, and SUN Grid Engine, allowing the reuse of existing workflows on different Grid middleware.

The following Section 2 describes the concept and the formalism of the *Grid Workflow Description Language* which is the basis for most of the components of the K-Wf Grid project. Section 3 then focuses on the concept, implementation and deployment of the *Grid Workflow Execution Service.* The interactive web-based visualization of the workflows is the topic of Section 4. Section 6 gives a brief overview of some of the applications and projects that currently make use of the Grid workflow execution environment. The article closes with the conclusions in Section 5.

2 Workflow Description

The basis for the workflow management is the *Grid Workflow Description Language* (GWorkflowDL), which is a Petri Net-based standard for describing workflows using XML. For the features and the specification of the Grid Workflow Description Language, please refer to the GWorkflowDL software development site at `http://www.gridworkflow.org/gworkflowdl/`. The workflow concept supports several levels of abstraction within one workflow description language, it is simple but universal (in the sense of being Turing complete), it supports data as well as control flow, and it enables dynamic workflows where the structure of the workflow may change during runtime. This section explains the underlying workflow concept and formalism.

2.1 High-Level Petri Nets

In 1962, C.A. Petri introduced in his Ph.D. thesis [20] a formalism for describing distributed processes by extending state machines with a notion of concurrency. Due to the simple and intuitive, but at the same time formal and expressive nature of his formalism, Petri Nets became an established tool for modeling and analyzing distributed processes in business as well as in the IT sector. A Petri Net is one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations. As such, a Petri Net has place nodes, transition nodes, and directed edges connecting places with transitions. If one abstracts from capacity constraints, Petri Nets are Turing complete. There exist several different types of Petri Nets. A common classification is based on a survey by [5], who distinguishes between three levels of Petri Nets:

- Level 1: Petri Nets characterized by places that can represent Boolean values; i.e., a place is marked by at most one unstructured token. Examples of level 1 nets are Condition/Event (C/E) systems, Elementary Net (EN) systems, and State Machines (SM).
- Level 2: Petri Nets characterized by places that can represent integer values; i.e., a place is marked by a number of unstructured tokens. Examples of level 2 nets are Place/Transition (P/T) nets, ordinary Petri Nets (PNs), and Free Choice Nets.
- Level 3: Petri Nets characterized by places that can represent high-level values; i.e., a place is marked by a multiset of structured tokens. Examples of level 3 nets are Colored Petri Nets (CPNs) and High-Level Petri Nets (HLPNs).

The tokens of a level 1 or level 2 net are unstructured (not distinguishable), so they do not carry any information besides their existence and number. These nets are used to describe basic control and data flows but are not suitable to model the data themselves. The tokens of a level 3 net, however, can be used directly in order to store the exit status (control data) or to model the input and output data (real data) of the previous activity, which are then evaluated by a following activity or the condition of a transition.

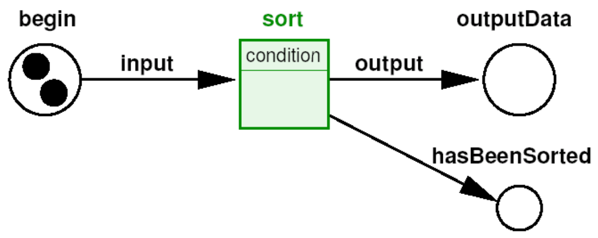


Fig. 1: Example Petri Net that models the input, output, precondition, and effect (IOPE) of a sort transition.

Our approach to using Petri Nets for the description of distributed workflows is to relate the tokens of a level 3 net with classes and instances of real data by means of High-Level Petri Nets (HLPNs) as shown in Fig. 1. HLPNs allow for nondeterministic and deterministic choices simply by connecting several transitions to the same input place and annotating edges with conditions. HLPNs also make the state of the program execution explicit with tokens flowing through the net that represent the input and output data as well as side effects. In contrast, classical directed acyclic graphs (DAGs) only have a single node type, and therefore data flowing through the net cannot be modeled easily. Using the concept of edge expressions, a particular service can be assigned to a transition, and conditions—also known as transition guards—may be used as an additional control flow. The resulting workflow description can be analyzed for certain properties such as conflicts, deadlocks, and liveliness using standard algorithms for HLPNs. High-Level Petri Nets are Turing complete because they overcome the capacity constraints (unbounded places) and therefore can do anything we can define in terms of an algorithm [1].

The following paragraph introduces some commonly used terms. High-Level Petri Nets are directed graphs, containing two distinct sets of nodes: *transitions* – represented by rectangles – and *places* – denoted by circles. Places and transitions are connected by directed edges. An edge from a place p to a transition t is called an *incoming edge* of t, and p is called *input place* of t. *Outgoing edges* and *output places* are defined accordingly. Each place can hold a certain number of individual *tokens* that represent data items flowing through the net. The maximum number of tokens on a place is denoted by its *capacity.* A transition is called *enabled* if there is a token present at each of its input places, and no output place has reached its capacity. Enabled transitions can *fire* by consuming one token from each of the input places and putting a new token on each of the output places. The number and values of tokens each place holds during the workflow execution is called *marking.* The *initial marking* specifies the marking at the beginning, and consecutive markings are obtained by firing transitions. Each edge in the Petri Net can be assigned an *edge expression.* For incoming edges, variable names are used as edge expressions, which assign the token value obtained through this edge to a variable of that name. Additionally, each transition can have a set of *conditions.* A transition can only fire if all of its conditions evaluate to true for the input tokens.

Petri Nets are suitable to describe the sequential and parallel execution of tasks with or without synchronization; it is possible to define loops and the conditional execution of tasks. As mentioned above, we use Petri Nets not only to model, but furthermore to control the execution of the workflow. In most cases, the workflow within Grid applications is equivalent to the dataflow, i.e., the decision when to execute a software component is taken by means of availability of the input data. Therefore, the tokens of the Petri Net represent real data that is exchanged between the software components. In this case, we use Petri Nets to model the interaction between software resources represented by transitions that are linked to Web Service operations, and data resources represented by places linked to SOAP parameters. In other cases, however, the workflow should be independent from the dataflow, and in addition, we have to introduce control places and control transitions. Control transitions only evaluate logical conditions. In the case that a transition is linked to a Web service operation, the tokens store the output parameter returned by the method call. The state of the workflow is represented by the marking of the Petri Net, which makes it easy to implement tools for workflow check pointing and recovery.

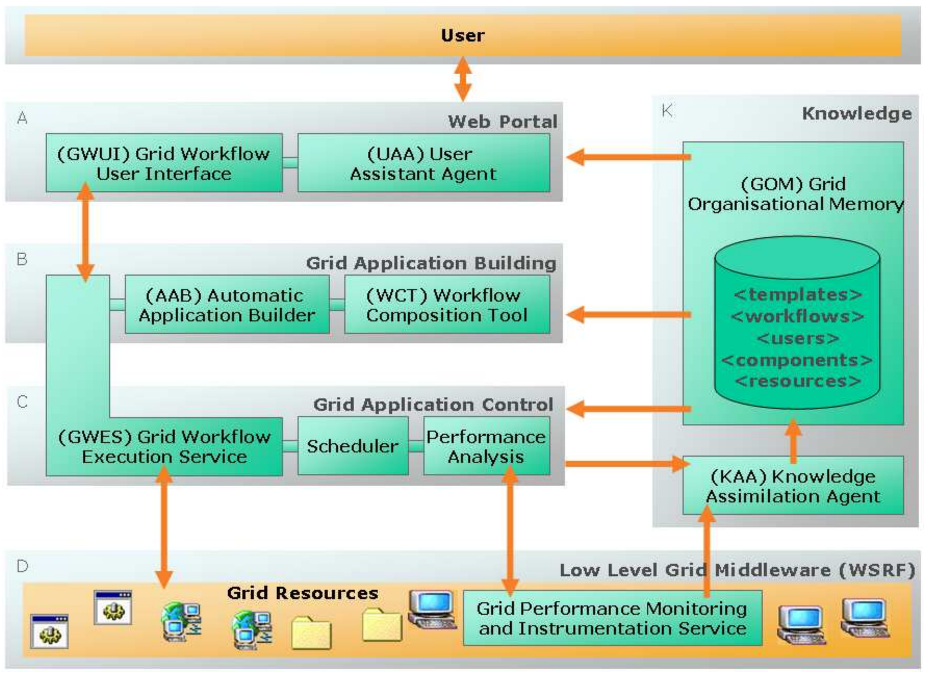

User
A
Web Portal
(GWUI) Grid Workflow User Interface
(UAA) User Assistant Agent
K
Knowledge
(GOM) Grid Organisational Memory
<templates>
<workflows>
<users>
<components>
<resources>
B
Grid Application Building
(AAB) Automatic Application Builder
(WCT) Workflow Composition Tool
C
Grid Application Control
(GWES) Grid Workflow Execution Service
Scheduler
Performance Analysis
(KAA) Knowledge Assimilation Agent
D
Low Level Grid Middleware (WSRF)
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 2: The K-Wf Grid architecture with the component layers *web portal* (A), *Grid application building layer* (B), *Grid application control layer* (C), the *vertical knowledge layer* (K) and *the lower level Grid middleware layer* (D).

3 Workflow Enactment

This section describes the concept, implementation and deployment of the *Grid Workflow Execution Service* (GWES). If you are interested in further documentation or in downloading the software, please refer to the GWES home page at `http://www.gridworkflow.org/gwes/`.

3.1 Architecture and Deployment

The Grid Workflow Execution Service is deployed as a standard Web Service, so it seamlessly integrates into common system architectures as shown in Fig. 2. There is a strict separation of the enactment machine and its client interfaces. The GWES delegates parts of the refinement process from abstract to concrete (executable) workflows to external web services, such as the Workflow Composition Tool (WCT) the Automatic Application Builder (AAB), the Resource Matcher, and the Scheduler [10], [11], [13], [19].

3.2 Workflow Refinement Concept – from Abstract to Concrete Workflows

The GWorkflowDL and the GWES support several levels of workflow abstraction that are displayed in Fig. 3. An initial input workflow provided by the user or by the user assistant may possess different abstraction levels, ranging from an abstract user request to the concrete (executable) workflow which can be invoked directly on the available Grid resources. The supported abstraction levels are:

- User Request (Red): The user request represents an abstract operation which has not been mapped onto potential workflows yet.
- Abstract Workflow (Yellow): An abstract (non-executable) workflow consists of operations of Web Service or program execution classes. The Workflow Composition Tool (WCT) automatically composes abstract workflows from the user requests, if the corresponding ontologies are represented within the Grid Organizational Memory (GOM). The user can also directly provide abstract workflows without using the WCT.
- Workflow of Service Candidates (Blue): Consists of lists of Web Service or program execution candidates, which match the Web Service (or program execution) classes. The mapping is done by the Automatic Application Builder (AAB) (or the resource matcher).
- Workflow of Service Instances (Green): Consists of concrete (executable) instances of Web Service operations or program executions. The selection of the optimal instance out of the list of candidates is delegated to the Scheduler or resource selector.
- Control Flow (Black): Black transitions within the workflow denote a control flow (separate from the data flow) which is not connected to any real operation.

Only green or black workflow nodes can directly be executed by the GWES. If the workflow contains red, yellow or blue nodes, the WCT, AAB, Resource Matcher, or Scheduler is invoked, in order to refine the workflow. If these components are not able to provide a solution, the GWES will suspend the workflow and the user has to refine the workflow, or to wait until a mapping comes available.

3.3 Workflow Enactment Concept

As High-Level Petri Nets are composed of four different kinds of graph elements (transitions, places, edges, tokens). Therefore it is relatively easy to implement a corresponding workflow enactment engine. Fig. 4 represents the enactment algorithm, which is implemented as kernel of the GWES.

First, the workflow engine parses, verifies, and analyzes the incoming workflow description. Next, the engine collects all enabled transitions according to the mathematical definition of the term *enabled* (refer to Section 2.1). For each enabled transition, a condition checker evaluates the attached conditions (also known as transition guards). If the condition is true and if the transition references a concrete activity, this activity is started, e.g., invoking a remote Web

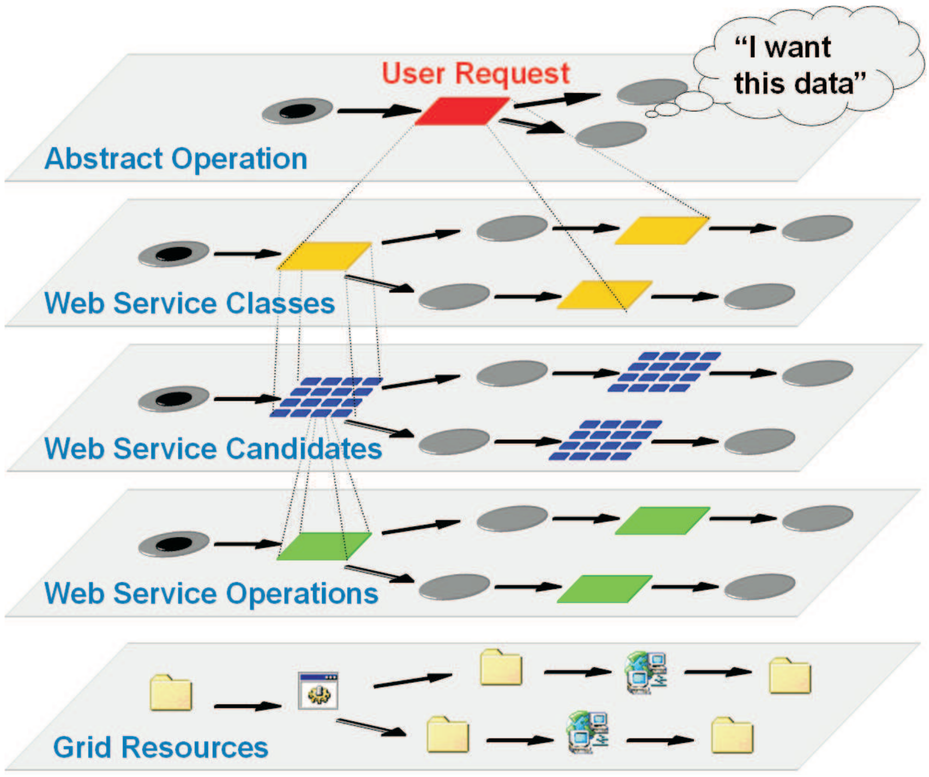


Fig. 3: The workflow abstraction levels that are supported by the GWorkflowDL and the GWES.

Service operation, using the input tokens as input parameters. If the activity completes, the corresponding transition fires; i.e., one token is removed from each input place and the activity results (data, side effects) are placed as new tokens on the output places. If the transition refers to an abstract activity, the transition has to be refined first. The new marking of the Petri Net enables subsequent transitions and their corresponding activities. If there are no more enabled transitions nor active activities remaining in the workflow, the workflow completes.

An advantage of Petri Net-based workflow engines is that they can process almost every workflow pattern without modification of the enactment engine. The Petri Net concept is very expressive and simple at the same time, and there is no need to implement any special functionality for workflow constructs, such as loops, if/then clauses, and synchronization points. All these workflow constructs are supported implicitly by the Petri Net approach, and the workflow engine itself does not have to bother about them if it implements the basic Petri Net rules.

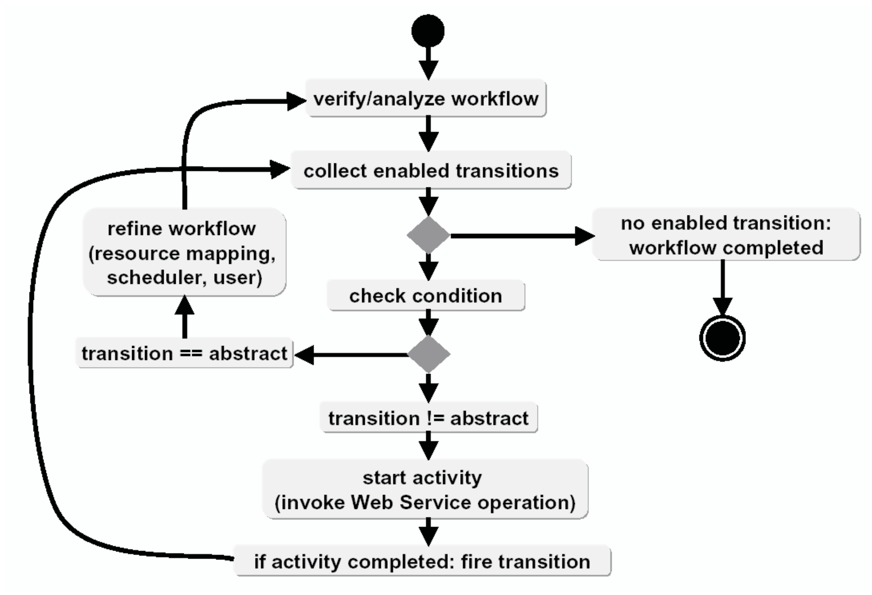

verify/analyze workflow
collect enabled transitions
refine workflow (resource mapping, scheduler, user)
no enabled transition: workflow completed
check condition
transition == abstract
transition != abstract
start activity (invoke Web Service operation)
if activity completed: fire transition

Fig. 4: The algorithm inside the Grid Workflow Execution Service's kernel to enact a Grid workflow.

In order to make the workflows persistent, the execution service uses a native XML database for storing snapshots of the workflows including the workflows' state, which is represented by the marking of the corresponding Petri Net. The system provides a simple and robust checkpoint-restore functionality, and the user may initiate each of these snapshots as a new workflow instance.

4 Workflow Visualization

The graphical workflow user interface is realized by a set of Java Applets and Servlets which can be easily integrated in generic or application-specific web portals. In addition, there exists a command line client program that implements the full set of workflow management features and simplifies the integration into legacy frameworks. The workflow synchronization between the execution service and the graphical user interface is done by means of an advanced protocol which makes use of the XUpdate syntax in order to transfer workflow modifications or status changes from the execution service to the clients and vice versa.

The *Grid Workflow User Interface* (GWUI) is an interactive graphical interface to the *Grid Workflow Execution Service.* It offers a set of features to initiate, monitor, and manipulate workflows. It also includes mechanisms to support user interactions during workflow execution like input data specification and decision making.

The intention of GWUI is to provide a library of GUI components suitable for workflow monitoring, manipulation and execution which can be differently combined so that different GUI setups are possible. Thus, graphical interfaces for different user requirements can be constructed on the basis of the GWUI library.

Fig. 5 shows a screenshot of the GUI component that dynamically visualizes the Petri Net of a running workflow. Further components include inspectors for the details of single elements inside the workflow. The inspectors can be used for monitoring and manipulation of diverse workflow properties. Furthermore, dialogs for workflow status control are offered as well as the possibility to upload and run workflows on the Grid. The GWUI also offers a dynamic task list showing all tasks the user has to accomplish in order to execute a certain workflow. This task list interfaces with a generic data input dialog framework so that application- and data-specific input dialogs can be triggered from within the GWUI.

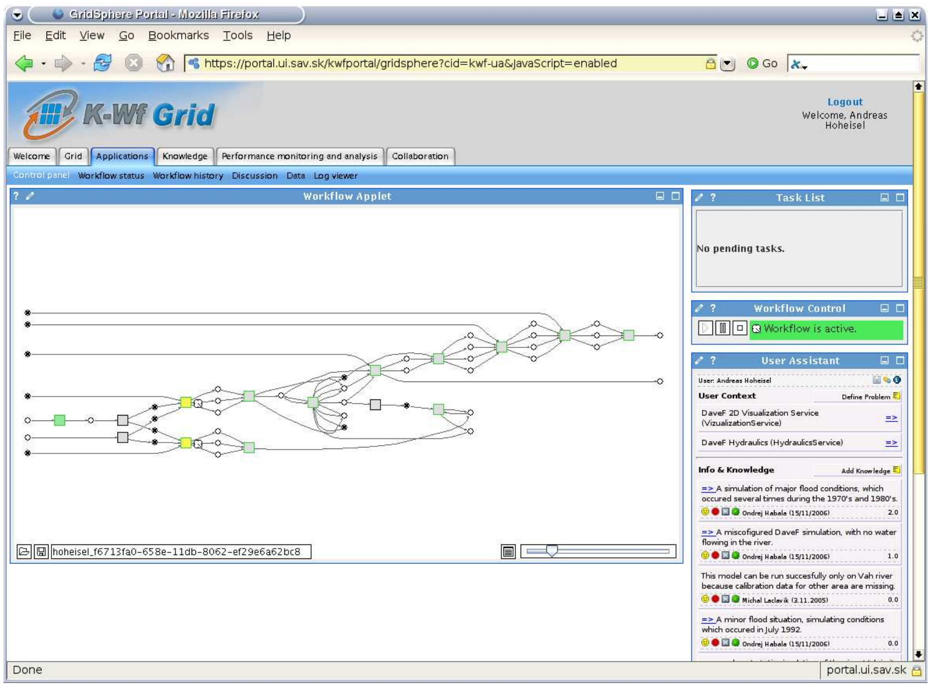
The GWUI has been designed regarding usability guidelines like flexibility and consistency as well as guidelines for information visualization. The diverse needs of different user groups have been addressed in the design. Furthermore, the GWUI is suitable for collaborative workflow execution and manipulation. Several instances of GWUI can be run on several computers and share the same workflow instance. By means of a distributed object model which is part of the GWorkflowDL library, all changes made to the workflow by one party will be instantly distributed to all others working on the same workflow instance.

The GWUI is implemented in Java 1.4.2 in order to support also older Browser Java plugins previous to Java 5, and it is based on the Java Swing GUI framework. The single GUI components are included in several Java applets which run inside the web browser of a client computer. In the K-Wf Grid portal, these applets are located inside single portlets offering the user the flexibility to construct a custom arrangement of the GUI components he/she requires.

5 Conclusions

The main scientific achievements of this work include the definition of a theoretically well-founded Workflow Description Language, which is very simple, but expressive at the same time, using the well-known concept of high-level Petri Nets. With this approach, we are able to model arbitrary algorithms, including parallel branches or loops. In addition, transitions can possess conditions that we express using the XPath 1.0 syntax. Another innovation is the overall concept of iteratively mapping abstract workflows onto concrete resources, taking into account the most current monitoring information. Each of the abstraction levels uses the same workflow description language, so it is possible to mix several levels of abstraction in one workflow.

Petri Nets are a well-established approach in computer science for modeling and analyzing distributed processes, whereas many workflow management systems in the e-Science domain use other workflow formalisms, such as BPEL


GridSphere Portal - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://portal.ui.sav.sk/kwfportal/gridsphere?cid=kwf-ua&JavaScript=enabled
Go
K-Wf Grid
Logout
Welcome, Andreas Hoheisel
Welcome Grid Applications Knowledge Performance monitoring and analysis Collaboration
Control panel Workflow status Workflow history Discussion Data Log viewer
Workflow Applet
hoheisel_f6713fa0-658e-11db-8062-ef29e6a62bc8
Task List
Workflow Control
User Assistant
User: Andreas Hoheisel
User Context
Define Problem
DaveF 2D Visualization Service (VizualizationService)
DaveF Hydraulics (HydraulicsService)
Info & Knowledge
Add Knowledge
Done
portal.ui.sav.sk

No pending tasks.

Workflow is active.

=> A simulation of major flood conditions, which occured several times during the 1970's and 1980's.
Ondrej Habala (15/11/2006) 2.0

=> A miscofigured DaveF simulation, with no water flowing in the river.
Ondrej Habala (15/11/2006) 1.0

This model can be run succesfully only on Vah river because calibration data for other area are missing.
Michal Laclavik (3.11.2005) 0.0

=> A minor flood situation, simulating conditions which occured in July 1992.
Ondrej Habala (15/11/2006) 0.0

Fig. 5: Screenshot of the GridSphere portlet representing the Grid Workflow User Interface together with the User Assistant Agent.

and DAG. The reasons for this are on the one hand the strong influence of industrial groups enforcing their own standards (e.g., BPEL) and on the other hand the wish to keep things very simple (DAG). The Petri Net approach is, nevertheless, a good candidate for becoming a vendor-independent standard for graph-based modeling of e-Science workflows, as it has formal semantics—which offer a profound theory background—and provides advanced analysis methods. An encouraging alternative is to base the workflow engine on Petri Nets and to map other high-level workflow formalisms (e.g., BPEL, UML) onto Petri Nets just before the workflow enactment. It is worth mentioning that many commercial workflow management systems in the business process domain are based on Petri Nets and that the semantics of UML 2.0 activity diagrams have been strongly influenced by them.

There exist several classes of Petri Nets that are suitable for different purposes.

In order to apply the Petri Net approach to the choreography, orchestration, and enactment of real-world e-Science workflows, High-Level Petri Nets (HLPNs) provide an adequate solution. However, we propose to extend the classical definition of HLPN for this purpose. We introduce a special notation for conditions (using the XPath 1.0 standard) to facilitate reactive workflow management in addition to the control and data flows that are explicitly modeled by the edges of the Petri Net. Transitions do not fire instantaneously, as they represent the invocation of real-world software components or services. The content of data tokens represents the real data that are produced by external software components or services. We use edge expressions to link places with specific software components or service parameters.

One drawback of the Petri Net approach is the fact that the graph may become very huge for complex and fine-grained systems. One solution to this problem is the use of hierarchical Petri Nets, where one transition represents a whole sub-Petri Net. The main application area for Petri Nets is in loosely coupled systems that exhibit a certain granularity of components.

6 Applications and Acknowledgements

The Grid Workflow Execution Service has been successfully applied in several application domains, such as city traffic management, environmental simulations [12] and media post processing. The research described here is supported in part by the European Union through the IST-2002-511385 project K-Wf Grid [16] and the IST-2002-004265 Network of Excellence CoreGRID [18]. Further known projects that currently make use of the GWES are Instant-Grid [15], MediGRID [17], Fraunhofer Resource Grid [6], Enterprise Grids [7], and BauVOGrid.

References

1. W. van der Aalst and A. ter Hofstede. Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. Technical report, Department of Technology Management, Eindhoven University of Technology P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, 2002.
2. Martin Alt, Sergei Gorlatch, Andreas Hoheisel, Hans-Werner Pohl: A Grid Workflow Language Using High-Level Petri Nets. CoreGRID Technical Report Number TR-0032, Institute on Grid Information, Resource and Workflow Monitoring Services, CoreGRID, 2006
3. Martin Alt, Andreas Hoheisel, Hans-Werner Pohl, Sergei Gorlatch: A Grid Workflow Language Using High-Level Petri Nets. R. Wyrzykowski et al. (Eds.): PPAM05, LNCS 3911, pp. 715-722, Springer-Verlag Berlin Heidelberg, 2006
4. Martin Alt, Andreas Hoheisel, Hans-Werner Pohl, Sergei Gorlatch: Using High Level Petri-Nets for Describing and Analysing Hierarchical Grid Workflows. In: Proceedings of the CoreGRID Integration Workshop 2005, Pisa, 2005
5. L. Bernardinello and F. de Cindio. A survey of basic net models and modular net classes. In Advances in Petri Nets 1992, The DEMON Project, volume 609 of LNCS, pages 304–351, London, UK, 1992. Springer-Verlag.
6. Fraunhofer Resource Grid – `http://www.fhrg.fraunhofer.de/`
7. Enterprise Grids Project – `http://www.epg.fraunhofer.de/`
8. Tomasz Gubala, Andreas Hoheisel: Highly Dynamic Workflow Orchestration for Scientific Applications. In: Proceedings of the CoreGRID Integration Workshop 2006, Cracow, 2006

9. Andreas Hoheisel and Martin Alt: Petri Nets. In: Workflows for eScience, Ian J. Taylor, Dennis Gannon, Ewa Deelman, and Matthew S. Shields (Eds.), Springer, 2006
10. Andreas Hoheisel, Thilo Ernst, Uwe Der: A Framework for Loosely Coupled Applications on Grid Environments. In: Grid Computing: Software Environments and Tools. Cunha, Jose C.; Rana, Omer F. (Eds.), 2006 ISBN: 1-85233-998-5
11. Andreas Hoheisel: User Tools and Languages for Graph-based Grid Workflows. In: Special Issue of Concurrency and Computation: Practice and Experience, Wiley, 2005
12. Andreas Hoheisel: Workflow Management for Loosely Coupled Simulations. In: Pahl-Wostl, C., Schmidt, S., Rizzoli, A.E. and Jakeman, A.J. (eds), Complexity and Integrated Resources Management, Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society, Manno, Switzerland, Volume 1, pp. 500-505
13. Andreas Hoheisel, Uwe Der: An XML-based Framework for Loosely Coupled Applications on Grid Environments. P.M.A. Sloot et al. (Eds.): ICCS 2003. LNCS 2657, 245-254, Springer-Verlag Berlin Heidelberg, 2003
14. Andreas Hoheisel, Uwe Der: Dynamic Workflows for Grid Applications. In: Proceedings of the Cracow Grid Workshop '03. Krakau, Polen, 2003
15. Instant-Grid Project – `http://www.instant-grid.de/`
16. K-Wf Grid Project – `http://www.kwfgrid.eu/`
17. MediGRID Project – `http://www.medigrid.de/`
18. Network of Excellence CoreGRID – `http://www.coregrid.net/`
19. Falk Neubauer, Andreas Hoheisel, Joachim Geiler: Workflow-based Grid Applications. In: Peter Sloot (Ed.) Future Generation Computer Systems, Volume 22, Number 1-2, pp. 6-15, Elsevier, 2006
20. C.A. Petri. Kommunikation mit Automaten. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962.

Constructing Abstract Workflows of Applications with Workflow Composition Tool

Tomasz Gubała[1,2], Daniel Harezlak[1], Marian Bubak[1,3], and Maciej Malawski[3]

[1] Academic Computer Center CYFRONET, ul. Nawojki 11, 30-950 Kraków, Poland
[2] Section Computational Science, Informatics Institute, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
[3] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Kraków, Poland
emails: `gubala@science.uva.nl, d.harezlak@cyfronet.pl,`
`{bubak, malawski}@agh.edu.pl`

Abstract

The idea of an application workflows is gaining popularity as a method of developing complex, distributed systems in a convenient way. Both in the Grid community and the area of service-oriented computation there are many research initiatives regarding the subjects of application workflow composition, execution, monitoring etc. What is more, there are solid indications that some results of that research entered realization and exploitation phase: there are end-user tools and solutions finding the way to the developers of distributed applications [7, 1]. In this paper we present the results of design, implementation and application of Workflow Composition Tool [8] that was created in the scope of K-WfGrid Project [3]. Its main purpose is to analyze user's requirements regarding the desired application results and work with information registries (that provide information on resources available at runtime) to finally provide an automatically composed proposition of a workflow of such application.

Keywords: Grid computations, workflow composition, web services

1 Introduction to Dynamic Scientific Workflows

The notion of an application workflow as a method of decomposition of complex systems into smaller, reuseable functional parts is already well-known [16]. Also the scientific community pays more attention to that type of processing since many research-related activities are decomposable into such interconnected processes. This approach allows for better structured and more legible approach to given task, its repeatability and reusability. What is more, when one combines that approach with the abilities of computational Grids, the combination offers the opportunity of executing large-scale, distributed computations on the Grid while conserving the natural and easy way of modeling them.

One of the features of workflow system especially crucial for scientific users is the ability of dynamic reformulation. Dynamic workflow may change during runtime – certain facts that occur during workflow execution may affect both the way further workflow steps are enacted and the structure of the workflow as well. This dynamic approach is better suited for the Grid execution environment due to its inherent ever-changing nature. A popular way to delivering the feature of dynamism in such systems is through introduction of different levels of abstractness [4]. The workflow described on the purely abstract level provides information on the business logic that forms the entire application and its decomposition structure and, as such, is well suited for reuse on many execution platforms [6]. Every time an abstract workflow is executed, its description has to be mapped onto certain realization composed out of the resources present in the environment. Development of such a system is part of the K-WfGrid Project [3] and the phase of composing abstract workflows is conducted by the Workflow Composition Tool (WCT) described within this work.

This work presents a summary of this research: it explains the designed composition algorithm, the way it was realized by a software system and what results were obtained with its application. For thorough, more detailed explanation of formal foundations of the composition algorithm and the other work that is related to this subject, please refer to [9].

2 Workflow Composition Tool Design

The Workflow Composition Tool is designed and developed as a component of a larger workflow orchestration and execution environment [10]. The aim of this tool is to construct a new abstract workflow as a solution to the input problem specified by a user as a set of application results he requires. The tool delivers a workflow of service operations that would eventually provide the specified results. The solution is abstract enough to support further reuse yet it is not executable straight away – it requires another refinement (concretization) step [5]. Later on, the refined workflow is ready to be executed in a dedicated enactment service.

The initial workflow description document that specifies all the results required by the user is provided in GWorkflowDL notation [2]. It may contain some parts of workflow already specified and it may have another (yet unknown) parts to be extended by the composer. Such parts (called unsatisfied dependencies) usually contain specification of data or effect that is required at a certain stage of workflow processing. Therefore the main objective now is to find suitable providers for this data (or producers of this effect) in form of one or more Grid services. To this end the composer queries an external ontological registry [12] which delivers in return an identification of suitable operations of Grid services. While the newly found operations may have their own dependencies (for instance in the form of required input data), WCT follows the same algorithm again to complete the workflow. When all the dependencies (both the original ones specified by the user and the ones introduced by new operations) are satisfied the workflow is complete. When there are no appropriate service operations available that would fulfill current requirements, the unresolved parts of the workflow are labeled and may be refined again when the environment changes (as the output is also given in the same GWorkflowDL notation, the tool is capable of re-reading its own workflow in order to extend it later). Figure 1 shows the process.

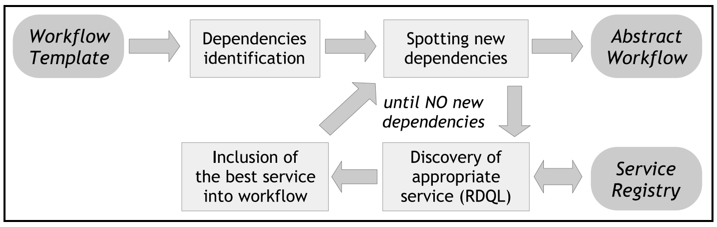

Workflow Template
Dependencies identification
Spotting new dependencies
Abstract Workflow
until NO new dependencies
Inclusion of the best service into workflow
Discovery of appropriate service (RDQL)
Service Registry

Fig. 1: Coarse-grained view of the composition algorithm within WCT.

In order to provide better request-to-service matching the tool's algorithms are based on semantic description of services' operations as building blocks of the composed workflow. By employing an operation functional matchmaking process based on Input-Output-Preconditions-Effect vectors the composer is able to find the best available provider of the data requested by the user. The algorithm of input-to-output comparison is as follows. Every input required or output produced by an operation is described with an element called *Data Template.* It defines a class of fitting data through a set of constraints: what is contained (*content constraint*), how it is expressed (*format constraint*) and how it is stored (*storage constraint*). Using this formally defined constraints the tool compares different data templates and decides whether given data belongs to specific class (and, therefore, could be used by certain service that requires it). The constraints values are URIs that point to specific concepts in a taxonomy stored inside dedicated knowledge store. The content constraint may point to a concept in domain-specific ontologies while the format constraint may use other means, e.g. an external taxonomy of data formats. Such a solution allows not only for free reuse of previously defined, well-established upper ontologies but also makes it possible to apply distinct comparison algorithms with regard to different aspects of data description [14].

3 Realization of the Workflow Composition Tool

The WCT is implemented as a service that offers its functionality through a remote interface and on its own contacts other needed services remotely. This loosely coupled architecture allows for easier distribution of components – the composer is essentially a stateless transformation that stores all the information it needs in an external memory (see Fig. 2). As the local resources are not crucial for the tool it may be freely relocated and replicated as long as it maintains a connection with the ontological store that contains registry of the available services. Internally the tool contains three main functional blocks, one that retrieves every unsatisfied dependency from the workflow. Another part uses the specification of the dependency to deliver a solution (for instance, through querying an external registry for suitable services or through searching the description of the workflow whether there are already appropriate services there). The final block is formed by solution appliers that incorporate the found solutions into

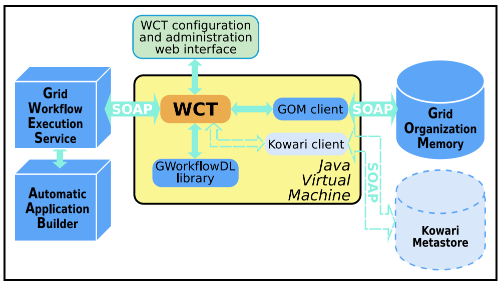


Fig. 2: Interaction of WCT with other K-Wf Grid components.

the current state of the composed workflow. Those three parts are activated in turns in subsequent iterations until the workflow is finally ready. The tool uses widely accepted SOAP standard for communication with external systems and it relies on the recent achievements of semantic web to obtain and process semantic information (RDQL/ITQL [15], OWL). In order to increase the possible applicability the tool is able to cooperate with different services that provide ontologies. It also uses XPath query standard in order to retrieve reusable subworkflow documents from the eXist XML database and to incorporate them as parts of constructed solutions.

4 Application of WCT and Measurements

In order to apply the presented research two applications were chosen, both of them being distributed simulations of complex phenomenons. Prior to application, the experts and maintainers of these simulations described the related domains and their tools in ontological taxonomies. One application is a large-scale scientific flood disaster simulation cascade [11] and the other is a simulation of traffic-generated pollution in a city. As a server for the composition tool, we use a PC-class machine with a single 3 GHz Pentium 4 CPU, 1 GB of memory and running SUN JDK1.5. Communication with the external ontology registry took place over a broadband connection with an average latency of 0.5 ms. The result in Fig. 3 show the average time of composition compared to the complexity of given workflows (that is the number of services in those workflows and interdependencies between them).

One may compare the time required for the computation (the dotted line) with the percentage of this time spent on contacting the external registry (the dashed line). The latter measurement includes a search through a large space of ontological triples with reasoning functionality applied and the time required to send and receive the result through the SOAP protocol. As may be seen this takes a considerable percentage of the entire composition time. For comparison, when the ontology registry is installed locally on the same computer, the resultant composition times (the rightmost plot) are much shorter and the registry communication also had less impact on the complete composition time.

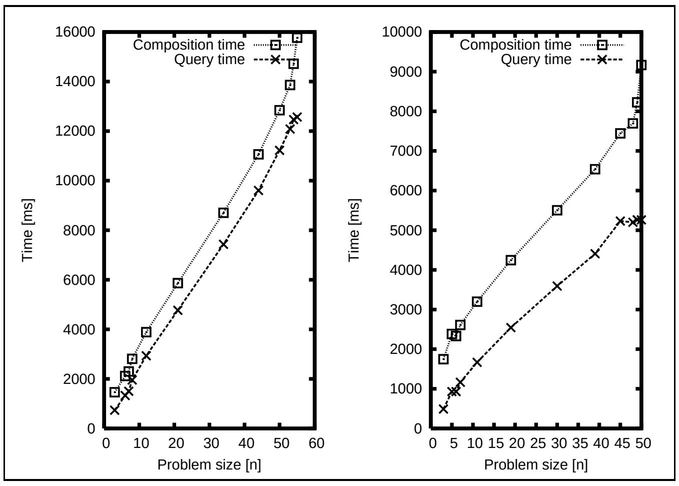


Fig. 3: Time to compose a workflow with the given number of operations.

In either case the obtained values are acceptable as the composition provides abstract, reusable workflows the creation of which is not required every time a workflow is executed.

5 Achievements and Conclusions

From the scientific point of view, we regard as the most important achievement the algorithms and heuristics [9] that are able to combine theory of description logic of service operations with the strict formalism of PetriNet-based workflows [13]. Special care was given to guarantee there are no deadlocks or undesired computation redundancy in the composed abstract workflows. Additional post-processing optimization algorithms are there to assure that the amount of needed user-provided data is minimized and to also reduce the size of the workflow graph.

From the realization point of view, the most important factor is the ability to provide formally and semantically valid workflows. As the tool uses domain-specific ontologies for workflow element description the entire processing is understandable to the end user as the ontologies provide common vocabulary. Due to the high level of abstraction applied the user is provided with meaningful functional blocks of the workflow rather than just resource addresses or technology-dependent endpoints. As the solutions proposed by the tool are abstract their descriptions do not outdate soon and thus may be used again (even automatically by the same tool in the course of building another workflow that requires similar processing as a previous one).

Finally, the feasibility of the implementation was finally proven with two applications from different computational science domains that represent complex, multi-level simulations.

Acknowledgements. This work was partly funded by the European Commission, Project IST-2004-511385, K-Wf Grid and the Polish State Committee for Scientific Research, SPUB-M 112/E-356/SPB/5.PR UE/DZ 224/2002-2004.

References

1. Netbeans enterprise pack. `http://www.netbeans.org/products/enterprise/`.
2. M. Alt, A. Hoheisel, H-W. Pohl, and S. Gorlatch. A grid workflow language using high-level petri nets. In R.Wyrzykowski, editor, *6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005*, volume 3911, pages 715–722. Springer-Verlag LNCS, 2005.
3. K-WfGrid Project Consortium. `http://www.kwfgrid.eu`.
4. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Metha, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, pages 25–39, 2003.
5. L. Dutka, B. Kryza, K. Krawczyk, M. Majewska, R. Slota, L. Hluchy, and J. Kitowski. Component-expert architecture for supporting grid workflow construction based on knowledge. In *Innovation and the Knowledge Economy. Issues, Applications, Case Studies*, pages 239–246. IOS Press, 2005.
6. T. Fahringer, S. Pllana, and A. Villazon. A-gwl: Abstract grid workflow language. In *Int. Conf. on Computational Science (ICCS)*, pages 42–49. Springer, 2004.
7. Eclipse Foundation. Eclipse bpel project. `http://www.eclipse.org/bpel/`.
8. T. Gubala, M. Bubak, and M. Malawski. Support for automatic workflow composition based on ontologies in semantic grid environment. In *5th Cracow Grid Workshop (CGW05) – Workshop Proc.*, pages 34–40. ACC Cyfronet AGH, 2006.
9. T. Gubala, D. Harezlak, M. Bubak, and M. Malawski. Semantic composition of scientific workflows based on the petri nets formalism. In *The 2nd IEEE International Conference on e-Science and Grid Computing*, page 12. IEEE Computer Society Press, 2006.
10. T. Gubala and A. Hoheisel. Highly dynamic workflow orchestration for scientific applications. In *CoreGRID Intergation Workshop 2006 (CIW06)*, pages 309–320. ACC CYFRONET AGH, 2006.
11. Ondrej Habala, Martin Maliska, and Ladislav Hluchy. Service-based flood forecasting simulation cascade in k-wf grid. In *6th Cracow Grid Workshop (CGW06) – Workshop Proceedings*. ACC Cyfronet AGH, to appear.
12. B. Kryza, R. Slota, M. Majewska, J. Pieczykolan, and J. Kitowski. Grid organizational memory – provision of a high-level grid abstraction layer supported by ontology alignment. *Future Generation Computer Systems*, 23(3):348–358, 2007.
13. V. Pankratius and W. Stucky. A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets. *Australian Computer Science Communications (ACS)*, 27(6):79–88, 2005.
14. M.A. Rodriguez and M.J. Egenhofer. Determiniing semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003.
15. A. Seaborne. Rdql – a query language for rdf. Technical report, Hewlett-Packard through W3C Consortium, 2004. `http://www.w3.org/Submission/RDQL/`.
16. J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *Special Issue on Scientific Workflows, SIGMOD Record*, 34(3), 2005.

Automatic Application Builder – Tool for Automatic Service Selection

Lukasz Dutka[1], Jacek Kitowski[1,2], and Szymon Natanek[1,2]

[1] Academic Computer Centre CYFRONET AGH,
ul. Nawojki 11, 30-950 Cracow, Poland
[2] Institute of Computer Science, AGH University of Science and Technology,
al. Mickiewicza 30, 30-059 Cracow, Poland
email: `dutka@cyfronet.krakow.pl`

Abstract
In this paper we try to describe process of automatic service selection as a part of workflow creation process if K-Wf Grid project. We present Automatic Application Builder (AAB) – tool developed in K-Wf Grid project, which retrieves information about services from knowledge repository and using rule-based expert system performs selection of best them. Thus AAB is able to match abstract workflow operations onto concrete one – ready for execution.
We also describe rule-based expert system [1, 2, 3, 4, 5] – the main part of developed tool, which takes into account user preferences allowing to customize and extend its functionality.

1 Introduction

The global evolution of grid systems [6, 7] and trends towards breaking any barriers in the access to grid, makes the environment much more complex then it used to be. At the moment the grid systems evaluate into a gigantic set of grid or web services deployed across to world where every minute might bring new tools deployed as services. The trend makes obvious that the future grid applications will be workflows consisted of distributed services. On the other hand gird must be user-friendly and accessible for everybody. Thus in K-Wf Grid we developed infrastructure automatizing process of construction workflow-based application trying to provide application basing only on general description of expected results provided by the user of the application.

Basing on general description of the expected results provided by the users, one of the K-Wf Grid tool tries to find a way of providing expected by the user results without going into details – the result of this phase we call an abstract workflow. Such a workflow is not ready for execution – it contains definition of tasks, which are described by abstract service classes and need to be concretized.

To obtain workflow, which is ready for execution, we developed Automatic Application Builder, which is responsible for matching mentioned service classes onto concrete service instances.

2 Abstract Service Operations

In K-Wf Grid project workflow we define several levels of abstraction of workflow task. Mapping between different levels of abstraction is done by workflow refinement by different tools.

Red – Abstract operation. The most abstract level of workflow task represents operation with known input and output data, but without specified details. For such an operation Workflow Creation Tool (WCT) creates proposition of workflow containing inter alia tasks described by class of web services.

Yellow – Operation on a Web Service class. A workflow of several WS-ClassOperations (abstract workflow) may represent a refinement of one abstract operation. Operations on Web Service classes are described by their interfaces including the semantic description of their functional behaviour.

Blue – List of Web Service operations. AAB produces list of concrete service instances which match given operation class and are best for performing given task. From among this service instances one is selected for execution

Green – Concrete operation. Last level of task abstraction is concrete operation. Such task contains concrete web service operation which is ready and scheduled for execution by Scheduler.

Black – Control operation. We define additional level of abstraction for tasks responsible for controlling execution of workflow. Such operations allow to define workflow control construct like for example splitting into branches or joining branches.

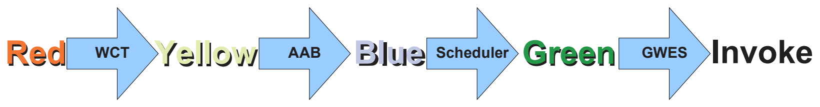


Fig. 1: Levels of workflow task abstraction.

Workflow (or its part) abstraction level is defined by highest abstraction level of its tasks. Only green and black tasks can be executed, thus in order to be executed, each abstract part of workflow have to be refined to achieve suitable level of abstraction.

3 Matching Abstract Service Class

AAB transfers workflow task from yellow abstraction level to blue one by matching abstract service class onto list of concrete service instances.

To achieve this goal, AAB exploits semantic knowledge gathered by a central K-Wf Grid knowledge repository (called Grid Organization Memory – GOM). It retrieves information about current status of system and currently registered service instances which match to given class and process it to get best services which can perform given workflow task.

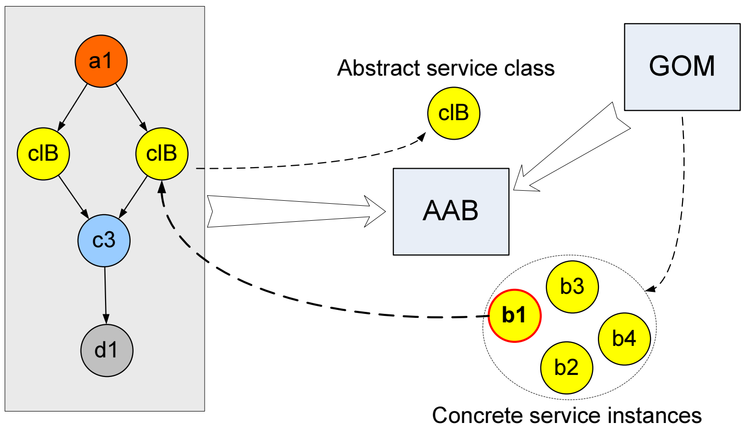


Fig. 2: Matching service class onto concrete instances.

Depending on current grid infrastructure, GOM can return varied, but in most cases large number of services. All those services realise same functionality, but they differ in non-functional parameters of execution. Such differences allow to define criteria for defining which services are better and worse, and finally create hierarchy among them.

4 Selection of Best Service Instances

Non-functional parameter of service execution can be treated as a separate criterion in their comparison. These criteria create multidimensional space where each service is represented as a point with coordinates represented by values of execution parameters. For example execution time and reliability create two dimensional space in which services with less execution time and greater reliability are more desirable as shown on figure 3.

AAB selects such a services which represent optimal compromise of each parameter value, thus are best for the realisation of the given workflow task. In the end AAB tries to facilitate execution and optimise workflow from the performance point of view.

5 Rule-Based Expert System

In dynamically changing environment of distributed services process of selection of best candidates for execution of wotkflow task requires a tool which can be

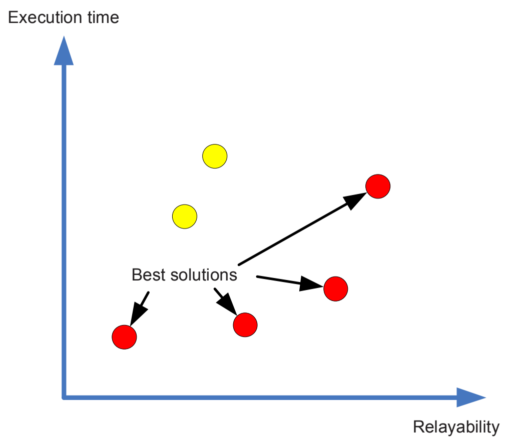


Fig. 3: Example of selection criterion.

dynamically adjusted to this environment. Registration of new services in GOM can deliver crucial information or new criterion data to making a decision about service selection. Moreover, evolution of system can cause appearance of new sources of knowledge which also can be decisive in selection. Therefore AAB includes rule-based expert system which is responsible for making decision about service selection.

Applied expert system uses Java Expert System Shell [8, 9, 3, 10] as an engine. It allows to load rules which are to be executed in run time, thus allowing to dynamically change behavior of system. Moreover, this feature allows to extend functionality of expert system by adding set of rules which are responsible for making decision.

AAB expert system [11, 12, 13] used extendable set of rules which define criteria of selection. Each set of rules is responsible for creating a hierarchy among services in relation to their one concrete non-functional parameter, and in the end selecting set of best of them.Each set of rules that is used by AAB is loaded in run time, thus it is possible to add some criterion of selection without necessity of stopping AAB. Aside from possibility of extending set of criterion, there is possibility to remove some criterion from system. In case that set of rules required by AAB does not exist, selection criterion related to given set of rules is omitted.

Use of extendable set of rules as a criteria for evaluating best fitted services, allowed us to change behavior of created tool in run time and easily customise it or extends its functionality. Moreover, it brings possibility to add handling of new service non-functional parameters which will be defined in future, or to create new criterion of selection.

Process of selection is divided into two stages as shown of figure 4. In first one rules sets which define criteria of selection are applied, giving set of services

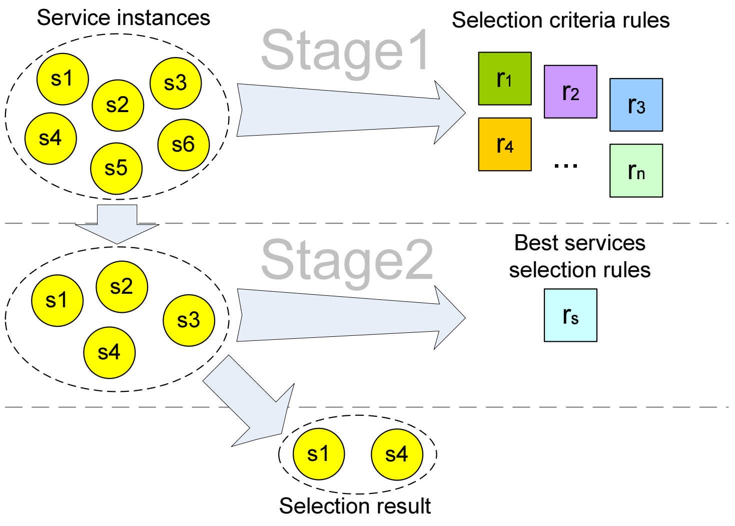


Fig. 4: Best services selection process.

which are best with respect to each of criterion. If there is no criterion of selection defined, it means that it does not matter which service is going to be executed and it is only required that service is executed. Therefore in such case best service is selected randomly from among propositions returned by GOM.

Second stage is responsible for selecting services which are best with respect to all criteria. We consider all criteria equally important, thus to decide which services are best we define relation which creates order among services.

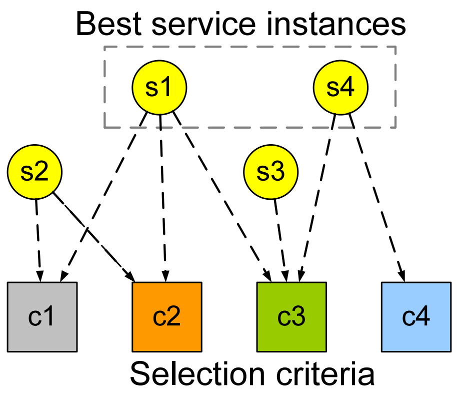


Fig. 5: Best services.

Service s_1 is considered as better then s_2 if set of criteria which s_1 meets includes set of criteria meet by s_2. In other words, s_1 is better then s_2 if best with respect to at lest all criteria that s_2.

Using given relation set of best service instances is calculated. Each service instance is compared with all other, and if exist instance that is better then it is removed from set of best services. Remaining services are considered to be equally good, and to be best with respect to all criteria.

6 User Preferences

Different service instances registered in GOM which represent certain class of service, can vary with non-functional properties such as for example execution time or reliability. These parameters may be crucial for user which can have different preferences in selection services for execution in different workflows. Therefore it is important to allow user to define how services should be selected with regard to these properties.

AAB strongly takes into account users preferences or requirements which should be fulfilled by services being of the part of the final workflow-based application. Each set of rules used by expert system which represent separate criterion in selection of best service is considered as users preference. User can choose which sets of rules should be used during selection thus, define criteria for selection of best services.

For each workflow task user can define list of rules sets which are used during process of selection of best services for execution of given task. It is possible that for the same task in different parts of workflow user can choose different sets of rules which will cause selection of different service instances in different stages of workflow execution.

7 Use of External Source of Knowledge

AAB makes it possible to use external sources of information derived from external systems or services in selection process. One can define own set of rules which retrieve information from external sources of information and use them decide which service instance is best for performing an operation.

AAB uses Java Expert System Shell [14] as an engine for expert system which select best services. It allows to execute any Java code inside of rules, therefore it is possible to perform processing of data retrieved from external sources, and prepare them to be used in selection of best services.

During integration of AAB in K-Wf Grid project, we already made use of external source of information to retrieve estimated execution time of given service operation. We prepared set of rules which invoke Knowledge Assimilation Agent web service to retrieve proper information, and then using it decide which service is best.

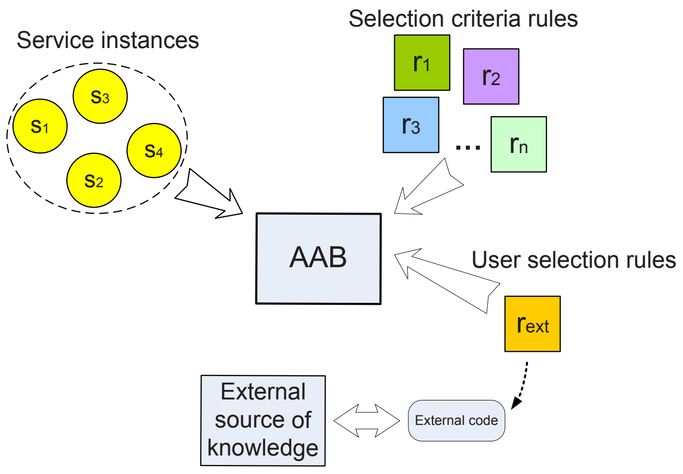


Fig. 6: Use of external sources of knowledge.

8 Summary

We have presented process of automatic service selection as a part of K-Wf Grid workflow refinement using rule based expert system. We have shown possibilities of customizing behavior and functionality of created tool, and mentioned its exploitation in K-Wf Grid project.

Presented expert system with extendable set of rules creates opportunities for further usage of AAB in other systems, and other application domains in K-Wf grid system. Moreover it allows to extends functionality of AAB with external sources of knowledge and data.

Acknowledgements. The work described in this paper was supported in part by the European Union through the project "K-WF Grid" funded by the European Commission program IST-2004-511385.

References

1. L. Dutka, R. Slota, and J. Kitowski. Component-Expert Architecture as Flexible Environment for Selection of Data-handlers and Data-Access-Estimators in CrossGrid. In M. Turala M. Bubak, M. Noga, eds., *Proceedings Cracow Grid Workshop '02*, pp. 201-209, Cracow, Poland, Dec. 11-14, 2002.
2. L. Dutka R. Slota D. Nikolow J. Kitowski K. Stockinger, H. Stockinger. Access Cost Estimation for Unified Grid Storage Systems. In *4th International Workshop on Grid Computing (Grid2003)*, Phoenix, Arizona, Nov. 17, 2003. IEEE Computer Society Press. (in print).

3. L. Dutka and J. Kitowski. Application of Component-Expert Technology for Selection of Data-Handlers in CrossGrid. In D. Kranzlmüller, P. Kacsuk, J. Dongarra, and J. Volkert, eds., *Proc. 9th European PVM/MPI Users' Group Meeting*, volume 2474 of *Lecture Notes on Computer Science*, pp.25-32, Linz, Austria, Sept.29-Oct.2, 2002. Springer, 2002.
4. L. Dutka, R. Slota, D. Nikolow, and J. Kitowski. Optimization of Data Access for Grid Environment. In *1st European Across Grids Conference*, Lecture Notes in Computer Science, Universidad de Santiago de Compostela, Spain, Feb. 13-14, 2003. Springer. (in print).
5. L. Dutka and J. Kitowski. Flexible Component Architecture for Information WEB Portals. In: A. Bogdanov J. Dongarra A. Zomaya Y. Gorbachev P. Sloot, D. Abramson, eds., *Proc. Computational Science – ICCS 2003 International Conference*, Lecture Notes in Computer Science, Saint Petersburg Russian Federation, Melbourne Australia, June 2-4 2003. vol. 2657, pp.629-638, Springer, 2003. (in print).
6. J. Nick S. Tuecke I. Foster, C. Kesselman. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG. Argonne National Lab., Global Grid Forum, June 22, 2002.
7. I. Foster and D. Gannon. *Open Grid Services Architecture Platform.* Argonne National Lab., Feb. 16, 2003.
8. C. Duvel. *Establishing rule-based models to implement workflow within construction organizations.* PhD thesis, University of Florida, 1999. UMI 9976532.
9. C. Duvel and R. R. A. Issa. The application of expert systems to controlling workflow within the construction management environment. In: B. Kumar and B.H.V. Topping, eds., *Artificial Intelligence Applications in Civil and Structural Engineering*, pp.61-72. Civil-Comp Press, 1999.
10. K. Parsaye and M. Chignell. *Expert Systems for Experts.* John Wiley, 1988.
11. L. Dutka and J. Kitowski. Expert Technology in Information Systems Development Using Component Methodology. In *Proc. of Methods and Computer Systems in Science and Engng.*, pp. 199-204, Cracow, Nov.19-21, 2001. ONT. (in Polish).
12. L. Dutka, P. Nowakowski, J. Kitowski: "Java-Based Component Expert Framework for Automatic Application Builder". In Proceedings of the Cracow Grid Workshop 2004, pp.270-274, Academic Computer Centre CYFRONET AGH, ISBN 83-915141-4-5, Poland 2005
13. L. Dutka, B. Kryza, K. Krawczyk, M. Majewska, R. Slota, L. Hluchy, J. Kitowski: "Component-expert Architecture for Supporting Grid Workflow Construction Based on Knowledge". In Cunningham P. Cunningham M. (Eds) Innovation and the Knowledge Economy. Issues, Applications, Case Studies. Vol. 2. IOS Press 2005. pp. 239-246.
14. Java Expert System Shell home page: `http://www.jessrules.com/`

K-WfGrid Scheduler – Scheduling Grid Workflows Based on Prediction and Performance Knowledge

Marek Wieczorek, and Thomas Fahringer

Distributed and Parallel Systems Group,
Institute of Computer Science, University of Innsbruck
email: marek@dps.uibk.ac.at

Abstract

The K-WfGrid scheduler is a performance oriented workflow Grid scheduler, responsible for making workflow applications ready for execution, by assigning workflow transitions to the resources where the web services corresponding to the transitions should be executed. It is meant as a tool which uses the performance estimation techniques (knowledge-based performance predictions and performance monitoring) developed in the K-WfGrid project, and applies them to make a reliable workflow scheduling. We present the scheduler in context of the whole project, explaining the way how it is adjusted to the workflow representation used in the project, and how it uses other components of the project to support the scheduling decisions. We also explain in detail the scheduling techniques implemented in the scheduler.

1 Introduction

The K-WfGrid project [1] provides a complete application execution and composition environment for the Grid. The workflow description language used in K-WfGrid called GWorkflowDL [2] is based on the Petri Net notation which is a successful representation used to describe the behavior of distributed systems. Workflow elements can be described on different abstraction levels (represented by different colors), of which only the lowest *green* level allows for execution. The Web Service technology is used in K-WfGrid as the representation describing activity interfaces. At the lowest abstraction level, every workflow activity (every workflow *transition*, using the Petri Net notation) is described by a web service interface, as a web service operation. The service responsible for processing and execution of the workflows is called Grid Workflow Execution Service (GWES). GWES can process the workflows which are at the lowest abstraction level. The services responsible for service creation are called Workflow Conversion Tool (WCT) and Automatic Application Builder (AAB). The scheduler is a service which prepares workflows for execution, by transforming the transitions to the lowest abstraction level (i.e., by solving all ambiguities in the workflow structure). The dependences between different components of the K-WfGrid environment are depicted in Fig. 1. WCT and AAB apply sophisticated reasoning techniques to create a valid workflow which will produce proper results based on the given input data. The result workflow may have multiple semantically equivalent web service deployments mapped to a single transition, only one of which can be used in the real execution (such a workflow is called *workflow of service candidates*). When creating workflows, WCT and AAB do not take into consideration any performance aspects of workflow execution. In this way, the workflow created by WCT and AAB is neither fully *concrete* nor optimized with respect to performance.

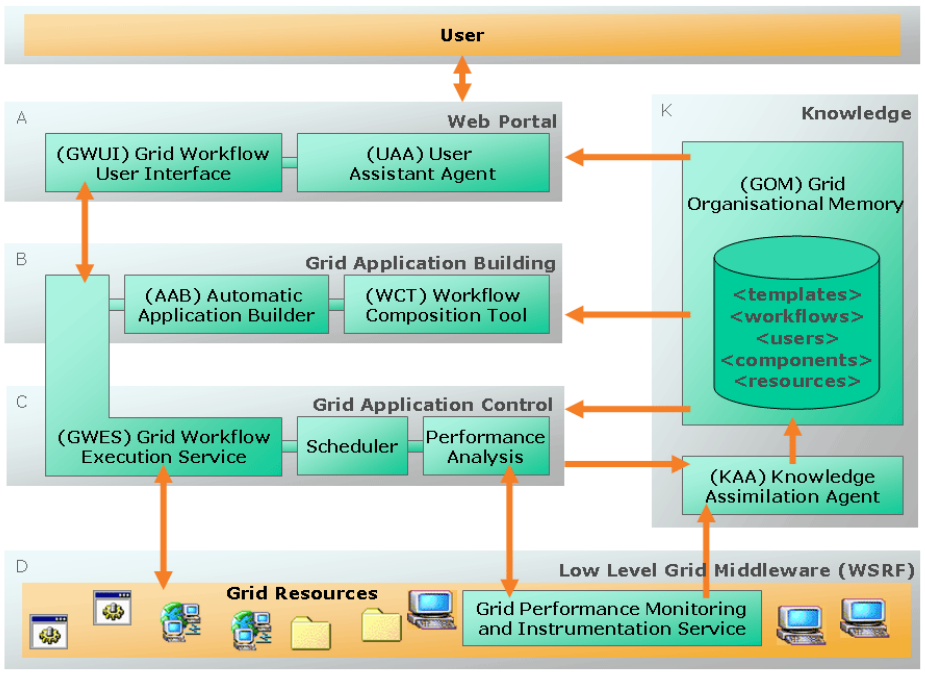

User
A
Web Portal
(GWUI) Grid Workflow User Interface
(UAA) User Assistant Agent
K
Knowledge
(GOM) Grid Organisational Memory
<templates>
<workflows>
<users>
<components>
<resources>
B
Grid Application Building
(AAB) Automatic Application Builder
(WCT) Workflow Composition Tool
C
Grid Application Control
(GWES) Grid Workflow Execution Service
Scheduler
Performance Analysis
(KAA) Knowledge Assimilation Agent
D
Low Level Grid Middleware (WSRF)
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 1: K-WfGrid environment.

The scheduler processes the workflows created by WCT and AAB, and selects between semantically equivalent variants of web service deployments (*service candidates*), assigning in that way transitions to the resources where the candidates are deployed. The goal is to optimize the performance of workflow execution, by trying to find a workflow mapping which will result in the shortest possible workflow execution time. Multiple scheduling algorithms implementing different graph traversing approaches (full-graph analysis or individual transitions analysis), supported by different types of auxiliary data (performance predictions or performance monitoring), can be used interchangeably. As the general workflow scheduling problem is known to be NP-complete, the implemented scheduling algorithms are heuristics rather than exact algorithms.

The reminder of the paper is organized as follows. In Section 2 we describe the basic architecture of the scheduler designed to work in the K-WfGrid environment depicted in Figure 1. In Section 3 we present the scheduling methods implemented in the scheduler, showing the way in which the knowledge data generated in the K-WfGrid environment is applied to support the scheduling decisions. Finally, Section 4 concludes the paper.

2 Scheduler Architecture

The scheduler consists of a single software component, which can be accessed by the user via different interfaces. It can be deployed as an Axis web service and accessed via a web service interface, or used as a normal Java library and accessed via Java API. The user can configure and monitor the deployed service using a graphical web interface (a Java servlet).

The scheduler is used by GWES when a workflow created by WCT and AAB contains *blue* transitions (i.e., transitions assigned to alternative *service candidates*). The scheduler tries to make the best possible scheduling, supported by the information obtained from auxiliary services. Performance predictions which allow for estimation of expected execution time for different service candidates can be obtained from the component of K-WfGrid called Knowledge Assimilation Agent (KAA). Performance measurements (both static and dynamic information about the resources) can be obtained from the Performance Monitoring and Analysis service. If the auxiliary data is not available for some resources or deployments, the scheduler uses default values. The interactions of the scheduler with different components are depicted in the use case diagram in Fig. 2.

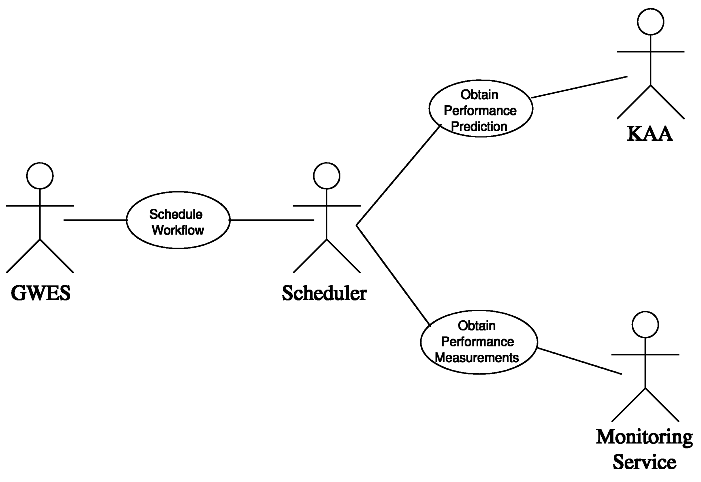

Obtain Performance Prediction
KAA
Schedule Workflow
GWES
Scheduler
Obtain Performance Measurements
Monitoring Service

Fig. 2: Use Case diagram of the Scheduler.

Fig. 3 shows the class diagram of the scheduler, including different algorithms implemented. The scheduler implements several scheduling algorithms which can be used as interchangeable plugins. The *easy* algorithm (class *EasyScheduling*) provides a basic functionality of the scheduler, as it changes workflow transitions from *blue* to *green* by selecting always the first option in the list of web service candidates. The *random* algorithm (class *RandomScheduling*) is a simple modification of the *easy* algorithm, which selects service candidates in a random way (rather than selecting always the first candidate from the list). The other three algorithms implement more advanced scheduling techniques, using data obtained from other K-WfGrid services. The *performance-based* algorithm (class *PerformanceDrivenScheduling*) uses the performance data from the Performance Monitoring and Analysis service to select the resources which offer the most appropriate execution conditions, from the point of view of performance. The *prediction-based* algorithm (class *KAABasedScheduling*) uses execution time predictions from the KAA service to minimize the execution time of the workflow. The *prediction-and-performance-based* algorithm (class *KAAPerformance-BasedScheduling*) combines the two aforementioned algorithms, and applies the predictions modified by the current performance information to optimize the execution time of workflows. The last two algorithms implement an extension of the Heterogeneous Earliest Finish Time (HEFT) [4] algorithm which proved to be an efficient approach for scheduling of scientific workflows [5].

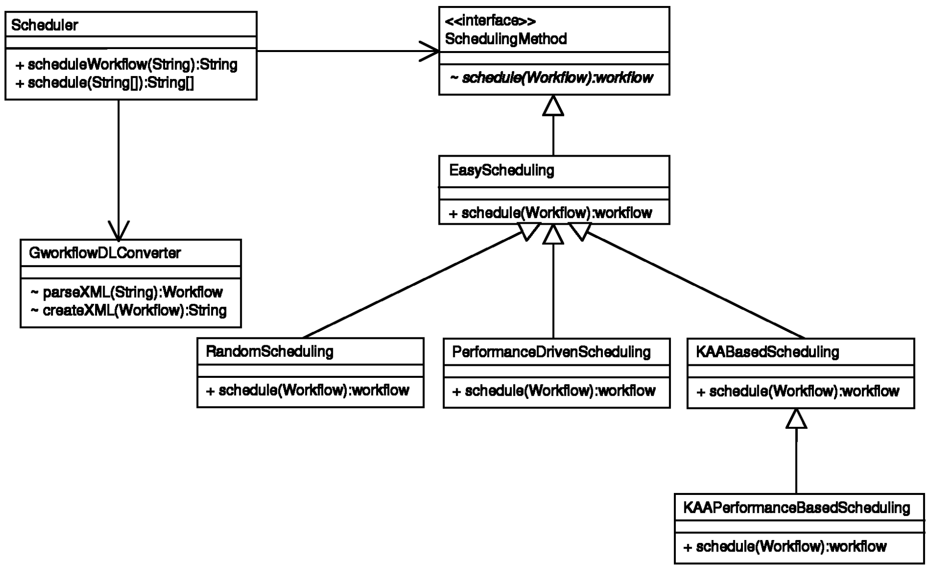

Scheduler
+ scheduleWorkflow(String):String
+ schedule(String[]):String[]
<<interface>>
SchedulingMethod
~ schedule(Workflow):workflow
EasyScheduling
+ schedule(Workflow):workflow
GworkflowDLConverter
~ parseXML(String):Workflow
~ createXML(Workflow):String
RandomScheduling
+ schedule(Workflow):workflow
PerformanceDrivenScheduling
+ schedule(Workflow):workflow
KAABasedScheduling
+ schedule(Workflow):workflow
KAAPerformanceBasedScheduling
+ schedule(Workflow):workflow

Fig. 3: Class diagram of the Scheduler.

3 Scheduling Algorithms

Different scheduling algorithms apply different strategies to select the service candidates to be executed on the Grid. Fig. 4 shows the operation of an example scheduling algorithm processing an example workflow.

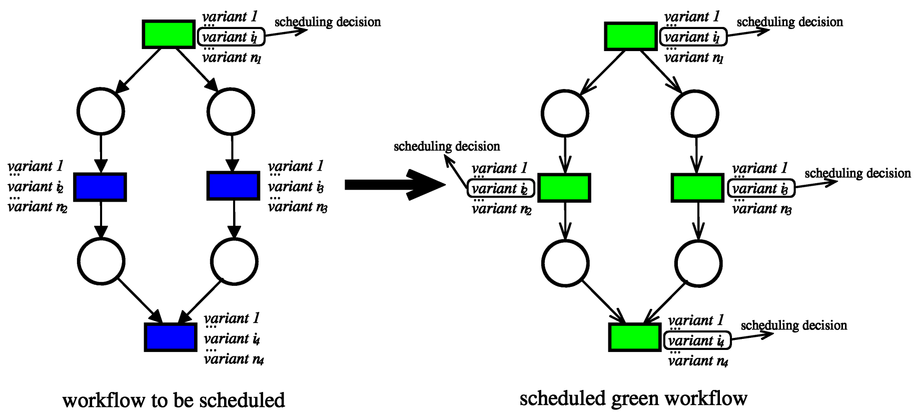


Fig. 4: Example of workflow scheduling.

As mentioned in Section 2, there are five different algorithms implemented in the scheduler: *easy*, *random*, *performance-based*, *prediction-based*, and *prediction-and-performance-based.* The two first algorithms are *basic* algorithms which can schedule workflows even when no performance data is available. The three last algorithms are *advanced* algorithms, and require performance data to support the scheduling decisions.

3.1 Basic Algorithms

The *easy* scheduling algorithm traverses a whole workflow and schedules all the *blue* transitions in the workflow. The algorithm selects always the *first* web service candidate assigned to each transition. The *random* scheduling algorithm is similar to the random algorithm. The only difference is that it selects the service candidates in a random way, not always the first candidate as in case of the easy algorithm.

3.2 Advanced Algorithms

Performance-Based Algorithm

The *performance-based* algorithm is based on performance data (both static and dynamic), taken from the Performance Monitoring and Analysis service. The scheduler queries the service to get the following data:

- for each physical resource r:
  - number of CPUs of the resource: n
  - CPU speed of the processors on the resource: $cpu\_speed(p)$
- for each CPU p_1, p_2, ..., p_n, fraction of CPU which is idle (available):

$available\_cpu(p_k)$, $available\_cpu(p_k)$ $\in$(0%,100%)

The algorithm calculates a special performance indicator P for each physical resource:

$$P\left(r\right) = cpu\_speed\left(r\right) \cdot \sum\nolimits_{i=1}^{n} available\_cpu\left(p_i\right) \cdot q^{i-1}$$

r – a physical resource

q – arbitrarily determined coefficient for multiple CPUs, $q \in$[0,1] (by default q=1.0)

The scheduling decisions made by the algorithm try to maximize the performance indicator, by selecting always the instance of the service which is deployed on the resource with the maximum value of P (see Fig. 5).

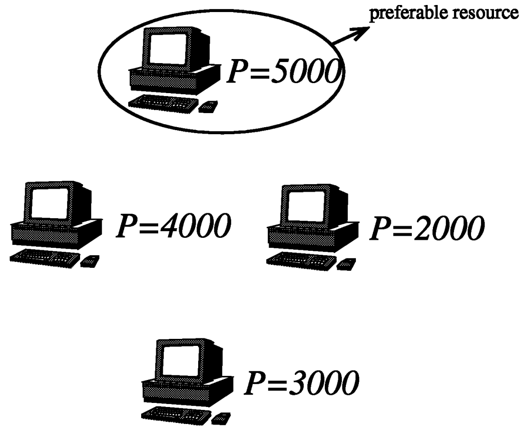
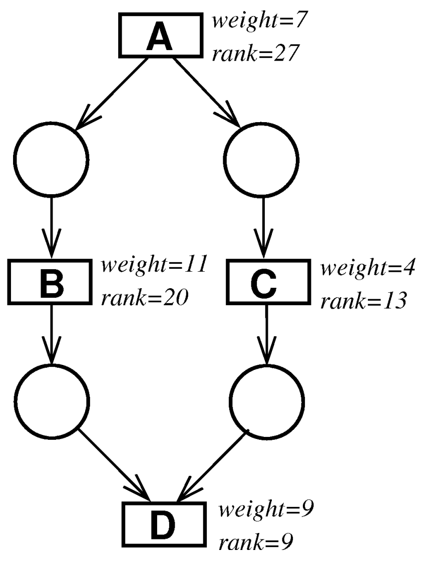


Fig. 5: Performance indicators for an example Grid.

Prediction-Based Algorithm

The *prediction-based* algorithm uses execution time predictions produced by the KAA service to minimize the execution time of workflows. The predictions for every service candidate are used to support the scheduling. The algorithm extends the HEFT algorithm mentioned in Section 2 which is a list scheduling algorithm designed for Directed Acyclic Graphs (DAGs). HEFT has been successfully applied in the ASKALON project [3]. The extensions applied in the prediction-based algorithm consist in adjusting the algorithm to the workflow model based on colored Petri Nets which may include loops and conditional constructs, and where not all transitions can be processed by the scheduler (only *blue* transitions can be processed). In order to address this workflow model, the prediction-based algorithm converts the workflows to DAGs by *unrolling* loops and by considering conditional branches as parallel executions. In addition, the algorithm removes from the workflow all transitions which are not *blue* (the dependencies between transitions are preserved). The result DAG can be scheduled using HEFT.

HEFT is a *full-ahead* list scheduling algorithm which first orders the workflow transitions to form a list, and then processes the transitions from the list one after another, selecting the service candidate which is most optimal for each transition. The order in the list is determined according to the decreasing *rank values* of the workflow transitions. The rank value of a transition is calculated as its distance (i.e., minimal remaining execution time) from the transition to the end of the workflow. The performance predictions from KAA are used to predict the execution times for different service candidates. In order to calculate the rank value for a workflow transition, we first assign to each transition its *weight value* (i.e., the average value calculated over the performance predictions for all the service candidates assigned to the given transition). Then, the workflow is traversed bottom-up, and the rank value for each workflow transition is calculated. An example workflow scheduled using HEFT is shown in Fig. 6. In the figure, a pair (*transition*, *resource*) represents a service candidate which implements the *transition* and is deployed on the *resource*.



	r1	*r2*	*r3*	*w*
A	5	8	8	7
B	9	13	11	11
C	3	4	5	4
D	7	10	10	9

execution times on different resources (performance cache)

Ordered list:
A, B, C, D

Mapping:
A –> r1
B –> r1
C –> r2
D –> r1

Fig. 6: Example workflow scheduled with HEFT.

Prediction-and-Performance-Based Algorithm

The *prediction-and-performance-based* algorithm combines the two aforementioned algorithms (the performance-based algorithm and the prediction-based algorithm). It takes the execution time predictions from the KAA service and modifies them according to the performance data taken from the Performance Monitoring and Analysis service. Then, an algorithm similar to the prediction-based algorithm described above is executed, which uses the *modified execution time predictions* instead of the normal performance predictions. The modified execution time predictions are calculated based on the following data taken from the Performance Monitoring and Analysis service:

- for each physical resource r:
  - number of CPUs of the resource: n
- for each CPU p_1, p_2, ..., p_n, fraction of CPU which is idle (available):
  - $available\_cpu(p_k)$, $available\_cpu(p_k) \quad \in (0\%, 100\%)$

The algorithm calculates a special performance indicator P for each physical resource:

$$P(r) = \frac{1 - q^n}{(1-q) \cdot \sum_{i=1}^{n} available\_cpu(p_i) \cdot q^{i-1}}$$

r – a physical resource

q – arbitrarily determined coefficient for multiple CPUs, $q \in [0,1]$ (by default q=1.0)

Finally, the modified execution time prediction is calculated in the following way:

$$METP(d) = P(resource(d)) \cdot ETP(d)$$

d – a service candidate (deployment)

$resource(d)$ – resource on which the service candidate d is deployed

ETP(.) – execution time prediction obtained from KAA

$P(.)$ – performance indicator

The prediction-and-performance-based algorithm is the most advanced one used in the scheduler, as it applies both the knowledge-based performance predictions and the performance data to support the scheduling. By doing this, the algorithm takes advantage of the full potential of the K-WfGrid environment – both of the knowledge-based components applying sophisticated reasoning techniques to reason about the expected execution times, and of the monitoring services providing different types of performance data.

4 Conclusions

We presented the scheduler implemented in the K-WfGrid environment, which schedules workflows based on knowledge-based performance predictions and performance monitoring data. We described the scheduler as a component of the K-WfGrid which addresses the performance objective when preparing workflows for execution, by using the advanced reasoning and monitoring techniques implemented in K-WfGrid. In this way, the scheduler helps to fulfill the main goal of the K-WfGrid project which is the knowledge-based support of workflow construction and execution. The scheduler implements a number of scheduling algorithms designed for the workflow representation used in K-WfGrid, which are able to schedule workflows with and without performance support provided by the auxiliary components.

The future work on the K-WfGrid scheduler may focus on exploring some *just-in-time* scheduling techniques, based on a more dynamic communication protocol between the scheduler and GWES. As another possible research direction, also rescheduling techniques using the existing functionalities of the Performance Monitoring and Analysis service may be applied for workflow scheduling.

Acknowledgements. This research has been supported by the Project K-WfGrid (EU IST-2002-511385).

References

1. K-WfGrid project – `http://www.kwfgrid.eu`
2. M. Alt, S. Gorlatch, A. Hoheisel, and H.-W. Pohl. A Grid Workflow Language using High-level Petri Nets. In *Second Grid Resource Management Workshop*, Poznan, Poland, Sept. 2005.
3. T. Fahringer, R. Prodan, R. Duan, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazón, M. Wieczorek. ASKALON: A Grid Application Development and Computing Environment. *6th International Workshop on Grid Computing*, © IEEE Computer Society Press, November 2005, Seattle, USA.
4. H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE *Transactions on Parallel and Distributed Systems*, 13(3), pp. 260-274, March 2002.
5. M. Wieczorek, R. Prodan, T. Fahringer, Scheduling of Scientific Workflows in the ASKALON Grid Environment, *ACM SIGMOD Record*, 09/2005

DIPAS: Performance Analysis and Visualization Support for Grid Workflows

Hong-Linh Truong[1], Thomas Fahringer[2], Peter Brunner[2], Robert Samborski[2], and Vlad Nae[2]

[1] Distributed Systems Group, Vienna University of Technology, Austria
[2] Institute of Computer Science, University of Innsbruck, Austria
emails: `truong@infosys.tuwien.ac.at`, `{tf, brunner, robert, vlad}@dps.uibk.ac.at`

Abstract

The performance analysis services in the K-WfGrid project aim at providing online information about the performance of Grid workflows as well as Grid resources involved in the workflow execution. Such performance information provides not only to the user insights into the execution of workflows but also to the K-WfGrid middleware services, e.g., workflow execution and scheduler, knowledge about the performance for semi-automatically constructing and executing workflows. In this paper, we summarize the development of DIPAS, a novel integrated system which supports online performance monitoring and analysis of service-based workflows. DIPAS provides a Web portal for the user and offers open interfaces for workflow middleware service to conduct performance monitoring and analysis of Grid workflows, thus substantially simplifying the way the user and the middleware service interact with the workflow performance tool. DIPAS introduces several novelties by supporting performance monitoring, analysis, and search for performance problems of advanced workflows composed from Web/Grid services at multiple levels of abstraction and by unifying the analysis of Grid workflows and infrastructure into a single system.

1 Introduction

The K-WfGrid project [7] aims at supporting semi-automatic composition of Grid workflows based on Grid and Web services and execution of these workflows in the Grid environment. In doing so, the K-WfGrid system relies on various sources of information such as workflow knowledge, user input, and of course, performance information of Grid workflows and resources on which the workflows are executed. Such performance information is required not only after workflows finish their execution but also during runtime. The performance information is an important source from which semi-automatic workflow composition and execution can be supported. However, providing performance information for Grid workflows is not a trivial task as it requires us to be able to monitor many different services, integrate monitoring information from servies, and analyze the information and present the result to not only the end-user but also the workflow middleware service.

To fulfil the above-mentioned requirements, in K-WfGrid project, we have developed a distributed performance analysis service for Grid workflows. DIPAS – our performance analysis service for K-WfGrid – is able to support online workflow execution tracing, performance overhead analysis, and search for performance problems for Grid workflows, and to visualize the performance results in a Web portal. This paper summarizes our results in developing DIPAS by presenting and illustrating main features of DIPAS. Further detailed techniques implemented in DIPAS can be found in [19, 2, 20, 22].

The rest of this paper is organized as follows: Section 2 presents the architecture of the distributed performance analysis system. Performance data representations and interfaces are discussed in Section 3. We outline main performance analysis features in Section 4. We discuss how workflow middleware services invoke performance analysis features in Section 5. The monitoring and analysis portal is described in Section 6. Section 7 discusses related work and Section 8 summarizes the paper.

2 Integrated Architecture for Workflow Performance Monitoring and Analysis

The K-WfGrid DIPAS (Distributed Performance Analysis Service) controls the monitoring and instrumentation service, conducts the performance analysis of workflows at runtime, and provides performance metrics to clients. DIPAS supports both the user and the workflow middleware service. It collects the data dynamically from services running in distributed Grid sites, analyzes performance and monitoring data, and visualizes the performance results in a Web portal.

DIPAS includes three main parts: DIPAS Portal, DIPAS Portlets/PortletServices and DIPAS Gateway. The *DIPAS portal* provides a single place for the user to conduct the performance monitoring and analysis of Grid infrastructure and workflows. This portal is built on portlet and Java technologies. It provides various features for analyzing and visualizing performance data of both workflows and resources. The core component of the portal is a Java-based GUI for performance trace visualization, overhead analysis and search for performance of workflows. The DIPAS portal is deployed into a web container based on Tomcat [1]. The *DIPAS Portlet/PortletServices* are implemented by using Gridsphere [13]. These portlets process user-specified performance requests through web interfaces and generate contents displayed in the portal. In our implementation, DIPAS Portlets/PortletServices are used to provide monitoring data of Grid infrastructure to the portal. The *DIPAS Gateway* is the central component of our performance analysis and search for performance problems. A DIPAS Gateway is implemented as a WSRF service [11] based on GT4 [12]. A DIPAS Gateway acts as a mediator between the portal and various other K-WfGrid services (e.g., GOM – Grid Organizational Memory [4], GEMINI – Grid Monitoring and Instrumentation Service [3], GWES – Grid Workflow Execution Service [5]) involved in the performance monitoring and analysis of workflows. Moreover, it implements the performance overhead analysis and search for performance problems.

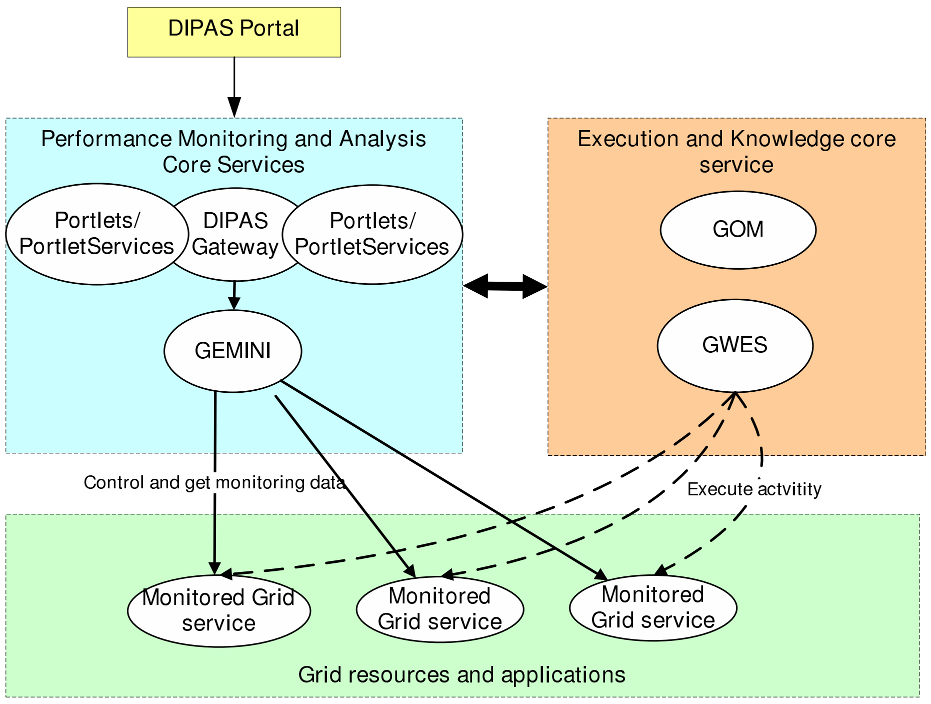

DIPAS Portal
Performance Monitoring and Analysis Core Services
Portlets/ PortletServices
DIPAS Gateway
Portlets/ PortletServices
GEMINI
Execution and Knowledge core service
GOM
GWES
Control and get monitoring data
Execute actvitity
Monitored Grid service
Monitored Grid service
Monitored Grid service
Grid resources and applications

Fig. 1: The K-WfGrid performance analysis and visualization system.

3 Performance Data Representations and Languages

In K-WfGrid, workflows are executed by GWES which is monitored by GEMINI. Considering complex interactions among various services (e.g., GWES, GEMINI and DIPAS) involved in the performance analysis of Grid workflows, we have to focus on the development of data representations and interfaces between these services in order to simplify the interoperability among different services. Our approach in ensuring smooth integration among different services relies on standard interfaces. To this end, firstly our services are built based on Web/WSRF technologies thus they can provide well-defined interfaces, e.g., based on WSDL. Secondly, we define common data representations to describe various types of monitoring data and common languages to describe various types of performance monitoring and analysis requests by using XML. Using Web/WSRF interfaces with XML-based data representations/requests, we could achieve the communication and data exchange interoperability among different services. In our

```
<MonitoringData dataTypeID="wfa.event"
resourceID="truong-03331c50-4537-11da-953e-e3f268ddc24d">
<event>
   <eventname>activity_created</eventname>
   <eventtime>1130231357715</eventtime>
   <eventdata>
         <attrname>ACTIVITY_NAME</attrname>
        <attrvalue>ctm_apec_transition_area_odm-flw</attrvalue></eventdata>
    <eventdata>
        <attrname>ACTIVITY_ID</attrname>
        <attrvalue>truong-03331c50-4537-11da-953e-e3f268ddc24d-0000000002</attrvalue>
    </eventdata>
    <eventdata>
        <attrname>TRANSITIONID</attrname>
        <attrvalue>ctm_apec_transition_area_odm-flw</attrvalue></eventdata>
     <eventdata>
        <attrname>ENDPOINTURL</attrname><attrvalue>http://grid02.softeco.it/Traffic
           FlowCalculator/services/TrafficFlowCalculator</attrvalue>
     </eventdata>
      <eventdata>
          <attrname>WSDLURL</attrname>
          <attrvalue>http://grid02.softeco.it/TrafficFlowCalculator/wsdl/TrafficFlow
             Calculator.wsdl</attrvalue>
      </eventdata>
      <eventdata>
          <attrname>OPERATIONNAME</attrname>
          <attrvalue>calculateTrafficFlow</attrvalue>
     </eventdata>
  </event>
</MonitoringData>
```

Fig. 2: Example of workflow events.

work, we have defined a rich set of schemas for describing workflow events, performance metrics, machine information [18] as well as instrumentation requests and application structure representations [9], etc. We briefly discuss the most relevant monitoring data representations and performance request languages in the following.

XML representations for performance and monitoring data: to facilitate the performance data exchange, in our system, each type of performance and monitoring data is specified by an XML schema and is identified by a unique name. Monitoring information is associated with a resource to be monitored. Thus, a message containing monitoring data of a monitored resource (e.g. machine, network path, code region) will consist of information about the unique data type name, the resource identifier, and detailed performance data. For example, Figure 2 gives a snapshot of a monitoring data related to an execution status of a workflow activity. The monitoring data includes an event named `activity_created` with the time at which the event occurs. Detailed information associated with the event is described by `eventdata` elements that describe various types of information such as activity name and ID, URL of the Web operations, the operation that the activity invokes, etc.

Performance data query and subscription requests: the analysis service needs to access various types of monitoring data from the monitoring services. Therefore, a well-defined language for specifying monitoring data request will simplify the integration among different services. We have defined a performance data query and subscription (PDQS) language [19, 20]. Using PDQS, requests expressed will be used in service interfaces for data query and subscription. PDQS allows us to specify the type of monitoring data associated with a monitored resource, the duration during which a subscription is valid, filter for the content of performance data, and the aggregation of requested data (e.g., MIN and MAX). Data filter in PDQS is expressed in XPath/XQuery [24] and based on specific data representations. Note that while we define a generic PDQS language, it is dependent on a specific implementation how PDQS is supported. For example, in K-WfGrid GEMINI [3], filters and aggregation features have not been supported. Furthermore, in K-WfGrid, PDQS requests could be constructed based on OWL [15] descriptions of monitoring data published in the K-WfGrid GOM [4].

Workflow analysis request language: one of the main issues in the K-WfGrid project is that performance analysis support is targeted not only to the end-user but also to the middleware services such as workflow composition and execution services. Therefore, we have to define a generic means based on that both supporting tools for end-user and the workflow middleware services can interact with performance analysis components in the similar fashion. To this end, a novel workflow analysis request language (WARL) [19, 20] has been devised. WARL allows any client to specify (i) constraints on objects to be analyzed, (ii) performance overheads to be analyzed and (iii) performance problem specifications. Constraints basically consist of a set of workflow concepts being monitored and analyzed, and their properties. In K-WfGrid, workflow concepts can be the entire *workflow*, a *workflow region* or an *activity*. Performance overheads that should be analyzed and provided are given by a set of metrics, each identified by a unique name. Performance problem specifications include a set of performance conditions, each condition includes a metric name, an operator (e.g., greater than or less than), and a value (e.g., indicating a threshold).

4 Performance Analysis and Visualization for Workflows

The web portal is the main interface through which the user conducts the monitoring of the Grid infrastructure and the performance analysis of Grid workflows. DIPAS supports four main features: infrastructure monitoring, workflow execution tracing, workflow performance overhead analysis, and search for performance problems.

Infrastructure monitoring: infrastructure monitoring is conducted through the DIPAS Portal that supports the user to query and subscribe interesting monitoring data of Grid infrastructure. The portal interacts with different services, using portlets and portlet services which retrieve the monitoring information from the monitoring service chosen. The portlets are used to obtain the input from the user, process the input and output the data when the result is returned from the different services. Portlet services are used to store the requests and the output for a user so that the user can access this data later on, after logging out from the system. When logging into the system, the user can access the data that has been processed previously. The portal is developed based on Gridsphere [13] which is an open-source framework for developing portlet-based Web portal for the Grid. We have implemented several portlets to access different types of monitoring data provided by the K-WfGrid GEMINI [3] and SCALEA-G [21] and to visualize the performance results in the portal. Furthermore, we have also supported dependability analysis for K-WfGrid services [23].

Workflow execution tracing: we support tracing workflow execution online. The user can request DIPAS Gateway to provide the existing workflows being executed or to be executed. Then, the user can select one of the existing workflows and start the monitoring of the workflow. The status of the workflow activities will be updated in the visualization when the execution status changes. During the execution of the workflow, the user can conduct an online performance and monitoring analysis. To do all these actions, the portal will communicate with the DIPAS Gateway, which interacts with the other services involved in the performance monitoring and analysis.

Workflow performance overhead analysis: performance of Grid workflows is analyzed based on well-defined overhead metrics. In doing so, clients can prepare a WARL-based request and send the request to the DIPAS Gateway. The resulting overhead will be computed by the *Overhead Analysis Component* which is a part of DIPAS Gateway. The overheads are determined based on the events and the workflow graph. Events are retrieved from GEMINI whereas the workflow graph is built based on workflow description obtained from GWES. The performance result will be described using XML representation and returned to the client.

Search for performance problems: during runtime of a workflow, performance problems of the workflow can be checked based on performance constraints established by the client. Clients can send analysis requests, described in WARL, to search for performance problems. These requests are then processed by the *Performance Search Component* which is a part of DIPAS Gateway. This component analyzes the performance of the workflows (running or completed) and reports performance problems, if any, to the client/user during runtime. Search for performance problems can be applied to both individual activities and workflow regions.

5 Invoking Performance Analysis Features inside Workflow Middleware

Since performance analysis services offer WSRF operations and accept XML-based requests, it is relatively simple to invoke performance analysis features inside workflow middleware services, e.g, for supporting runtime adaptation of workflow execution. Figure 3 presents the main service operations of DIPAS Gateway that can be invoked by any middleware service to control the performance analysis of workflows.

```
public String analyze(AnalysisRequestMessage request) throws RemoteException;
public DataEntries getResult(String resultID) throws RemoteException;
public CancelResultResponse cancelResult(String resultID) throws RemoteException;
```

Fig. 3: Main service operations used to control the performance analysis.

In DIPAS, a workflow middleware service can describe its performance request by using `AnalysisRequestMessage` which can be used to specify different requests, e.g. PDQS- and WARL-based requests using the same representation. The middleware service can specify (i) the content of the request whose representation is dependent on the language of the request, (ii) the language of the request (PDQS or WARL), and (iii) the command (e.g, query/subscribe monitoring data or analyze performance overhead/performance problems). Then, the middleware service can invoke the operation `analyze` to ask DIPAS to conduct the request. The performance information can be retrieved by invoking the operation `getResult`. Depending on specific request, the performance analysis can be conducted over a period of time. During the analysis, the middleware service can ask DIPAS stopping the analysis by calling `cancelResult` operation. For example, Figure 4 presents a code excerpt that is used to subscribe all events of a workflow identified by the ID `truong_96eeb880-125c-11db-a105-ce90a766c196` during the execution of the workflow. Based on the resulting events, the middleware service can react accordingly, for example, to reschedule the execution of an activity.

6 Integrated Performance Monitoring and Analysis Portal

In this section, we illustrate some features of our performance analysis portal. Figure 5 presents the portal for analyzing the availability of services and computation resources. Monitored resources, including Web/WSRF services, can be selected and their availability over a certain period of time can be analyzed. Results from the availability analysis can be used by the K-WfGrid scheduler and GWES in order to plan the execution of workflows.

Figure 6 presents the main portal through which the user can conduct the performance monitoring and analysis of workflows. A workflow and its regions are visualized in `Workflow Visualization` and the `Workflow Regions` panes, respectively. The `Workflow Regions` pane shows to the user the structured view of workflows that includes workflow regions like *branching, loop* or *sequence*, showing how activities are mapped together in the workflow. During runtime of a workflow, execution status of activities being executed will be updated in the `Workflow Visualization` pane. During runtime activity execution phases, such as `queueing` or `processing`, of activity instances are displayed, in detail, in the left pane of Figure 6. When observing the workflow execution at runtime, the user can select an activity and examine in detail performance metrics associated with the activity. Furthermore, the user can also specify performance analysis

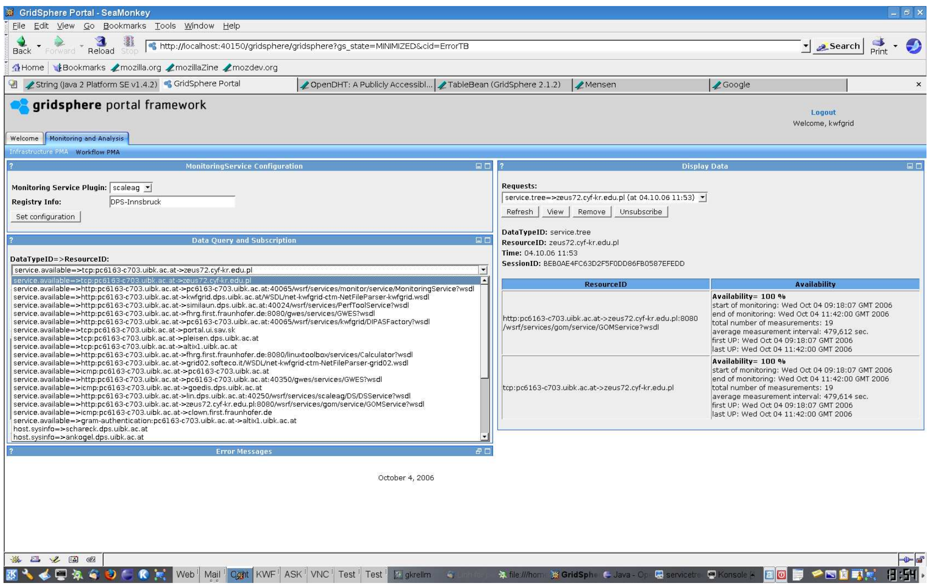
```
//example of pdqs, when subscription time =0, the subscription
//will be finished when the workflow execution finishes.

String pdqsRequest =<?xmlversion=\"1.0\" encoding=\"UTF-8\"?>
<PDQS>
<dataTypeID>wfa.event</dataTypeID>
<resourceID>
   truong_96eeb880-125c-11db-a105-ce90a766c196
</resourceID>
 <subscriptionTime><from>0</from><to>0</to>
 </subscriptionTime>
</PDQS>";
...
//create an analysis request message
AnalysisRequestMessage message = new AnalysisRequestMessage();
//specify the PDQS language
message.setLanguage("PDQS");
//fill the content
message.setContent(pdqsRequest);
//subscribe data
message.setCommand("COMMAND_SUBSCRIBE");
//call the analysis service
final String resultID = resource.analyze(message);
//if result is not null, we have data
if (resultID!=null) {
//update the current monitored workflow
  boolean poll =true ;
  ...
  while (poll) {
  DataEntries entries= resource.getResult(resultID);
  //no more data
  if (entries==null) {
    poll =false ;
    cancel();
   // end Timer if  no more data
   }
   //get the result
   String [] results=entries.getEntry();
   for (int i=0; i <results.length;i++) {
      DataEntries entries= resource.getResult(resultID);
      ...
   }
   ...
```

Fig. 4: Example of code excerpt used to request performance information.

requests such as search for performance problems or overhead analysis, and apply the requests to any activity. Figure 7 shows an example of performance problems associated with workflow regions `Branching0` and `Loop0`. Figure 8 presents all performance metrics, including performance overhead, associated with a workflow. Performance metrics are represented a breakdown tree which allows the user to examine in detail which factors strongly influence on the performance of the workflow.

7 Related Work

Supporting performance analysis in existing business workflow systems is limited. With the WebLogic Process Integrator [8] the user can examine status of


GridSphere Portal - SeaMonkey
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:40150/gridsphere/gridsphere?gs_state=MINIMIZED&cid=ErrorTB
Search
Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
String (Java 2 Platform SE v1.4.2)
GridSphere Portal
OpenDHT: A Publicly Accessibl...
TableBean (GridSphere 2.1.2)
Mensen
Google
gridsphere portal framework
Logout
Welcome, kwfgrid
Welcome
Monitoring and Analysis
Infrastructure PMA
Workflow PMA
MonitoringService Configuration
Data Query and Subscription
Error Messages
Display Data
Refresh
View
Remove
Unsubscribe
Set configuration

Monitoring Service Plugin: scaleag

Registry Info: DPS-Innsbruck

DataTypeID=>ResourceID:

```
service.available=>tcp:pc6163-c703.uibk.ac.at->zeus72.cyf-kr.edu.pl
service.available=>tcp:pc6163-c703.uibk.ac.at->zeus72.cyf-kr.edu.pl
service.available=>http:pc6163-c703.uibk.ac.at->pc6163-c703.uibk.ac.at:40065/wsrf/services/monitor/service/MonitoringService?wsdl
service.available=>http:pc6163-c703.uibk.ac.at->kwfgrid.dps.uibk.ac.at/WSDL/net-kwfgrid-ctm-NetFileParser-kwfgrid.wsdl
service.available=>http:pc6163-c703.uibk.ac.at->similaun.dps.uibk.ac.at:40024/wsrf/services/PerfToolService?wsdl
service.available=>http:pc6163-c703.uibk.ac.at->fhrg.first.fraunhofer.de:8080/gwes/services/GWES?wsdl
service.available=>http:pc6163-c703.uibk.ac.at->pc6163-c703.uibk.ac.at:40065/wsrf/services/kwfgrid/DIPASFactory?wsdl
service.available=>tcp:pc6163-c703.uibk.ac.at->portal.ui.sav.sk
service.available=>tcp:pc6163-c703.uibk.ac.at->pleisen.dps.uibk.ac.at
service.available=>tcp:pc6163-c703.uibk.ac.at->altix1.uibk.ac.at
service.available=>http:pc6163-c703.uibk.ac.at->fhrg.first.fraunhofer.de:8080/linuxtoolbox/services/Calculator?wsdl
service.available=>http:pc6163-c703.uibk.ac.at->grid02.softeco.it/WSDL/net-kwfgrid-ctm-NetFileParser-grid02.wsdl
service.available=>icmp:pc6163-c703.uibk.ac.at->pc6163-c703.uibk.ac.at
service.available=>http:pc6163-c703.uibk.ac.at->pc6163-c703.uibk.ac.at:40350/gwes/services/GWES?wsdl
service.available=>icmp:pc6163-c703.uibk.ac.at->goedis.dps.uibk.ac.at
service.available=>http:pc6163-c703.uibk.ac.at->lin.dps.uibk.ac.at:40250/wsrf/services/scaleag/DS/DSService?wsdl
service.available=>http:pc6163-c703.uibk.ac.at->zeus72.cyf-kr.edu.pl:8080/wsrf/services/gom/service/GOMService?wsdl
service.available=>icmp:pc6163-c703.uibk.ac.at->clown.first.fraunhofer.de
service.available=>gram-authentication:pc6163-c703.uibk.ac.at->altix1.uibk.ac.at
host.sysinfo=>schareck.dps.uibk.ac.at
host.sysinfo=>ankogel.dps.uibk.ac.at
```

October 4, 2006

Requests: service.tree=>zeus72.cyf-kr.edu.pl (at 04.10.06 11:53)

DataTypeID: service.tree
ResourceID: zeus72.cyf-kr.edu.pl
Time: 04.10.06 11:53
SessionID: BEB0AE4FC63D2F5F0DD86FB0587EFEDD

ResourceID	Availability
http:pc6163-c703.uibk.ac.at->zeus72.cyf-kr.edu.pl:8080 /wsrf/services/gom/service/GOMService?wsdl	**Availability= 100 %** start of monitoring: Wed Oct 04 09:18:07 GMT 2006 end of monitoring: Wed Oct 04 11:42:00 GMT 2006 total number of measurements: 19 average measurement interval: 479,612 sec. first UP: Wed Oct 04 09:18:07 GMT 2006 last UP: Wed Oct 04 11:42:00 GMT 2006
tcp:pc6163-c703.uibk.ac.at->zeus72.cyf-kr.edu.pl	**Availability= 100 %** start of monitoring: Wed Oct 04 09:18:07 GMT 2006 end of monitoring: Wed Oct 04 11:42:00 GMT 2006 total number of measurements: 19 average measurement interval: 479,614 sec. first UP: Wed Oct 04 09:18:07 GMT 2006 last UP: Wed Oct 04 11:42:00 GMT 2006

Fig. 5: Availability analysis portal.

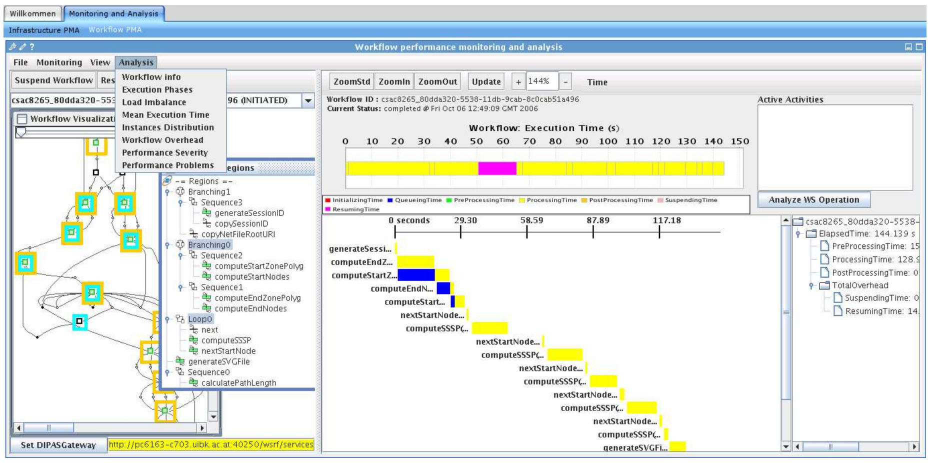
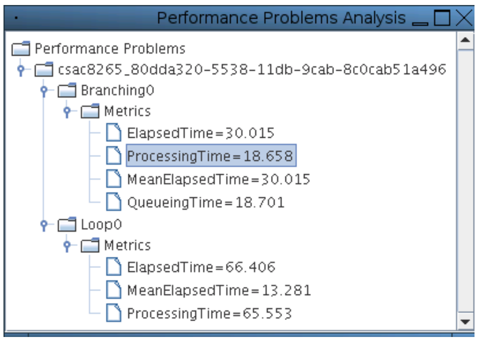


Fig. 6: K-WfGrid performance visualization portal for workflows.

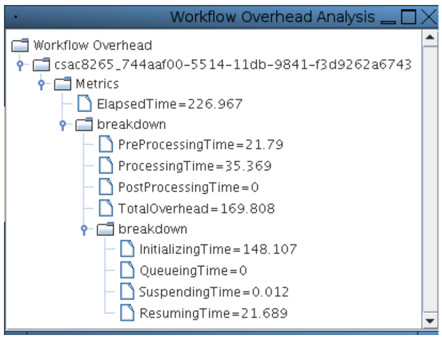
workflow instances. However, WebLogic Process Integrator monitoring is limited to the activity level. The Web Service Navigator [16] provides visualization techniques for Web service based applications.

Although many Grid workflow systems exist, as shown in [25], there is a lack of monitoring and analysis tools for workflows of Grid/Web services. P-GRADE [14] supports tracing of workflow applications. In contrast to K-WfGrid, it does



```
Performance Problems Analysis
Performance Problems
  csac8265_80dda320-5538-11db-9cab-8c0cab51a496
    Branching0
      Metrics
        ElapsedTime=30.015
        ProcessingTime=18.658
        MeanElapsedTime=30.015
        QueueingTime=18.701
    Loop0
      Metrics
        ElapsedTime=66.406
        MeanElapsedTime=13.281
        ProcessingTime=65.553
```

Fig. 7: Example of performance problems.



```
Workflow Overhead Analysis
Workflow Overhead
  csac8265_744aaf00-5514-11db-9841-f3d9262a6743
    Metrics
      ElapsedTime=226.967
    breakdown
      PreProcessingTime=21.79
      ProcessingTime=35.369
      PostProcessingTime=0
      TotalOverhead=169.808
      breakdown
        InitializingTime=148.107
        QueueingTime=0
        SuspendingTime=0.012
        ResumingTime=21.689
```

Fig. 8: Example of performance overheads.

not support cyclic and Web service-based workflows. Taverna Workbench [17] also monitors status of activities within Grid workflows, but provides information in simple tables. We notice that knowledge-based description for monitoring data is not in the focus of most performance tools. Considering the complexity of Grid workflows, metrics have to be well-described for extracting knowledge from monitoring data. The OntoGrid project [6] also uses knowledge gained from monitoring data to debug workflows, but not to monitor and analyze the performance. K-WfGrid performance monitoring and analysis services differ from these tools in many aspects. Our services are WSRF-based services, supporting online performance overhead analysis and detection of performance problems.

Many K-WfGrid monitoring and analysis methods and concepts are related to those in ASKALON [10] because they are contributions of the ASKALON team to the K-WfGrid project. However, K-WfGrid supports workflows of Web/Grid services while ASKALON works with workflows of C/Fortran-based scientific applications. As a result, performance analysis techniques in K-WfGrid are quite different from those in ASKALON. Also K-WfGrid provides a Web portal for conducting performance monitoring and analysis that is not available in ASKALON.

8 Conclusions and Future Work

In this paper, we have briefly presented main features of the K-WfGrid performance analysis and visualization system for Grid workflows. Our performance analysis system not only supports the end user but also the middleware to conduct the performance analysis of workflows at runtime. Various features such as performance execution tracing, workflow performance overhead analysis and search for performance problems have been implemented. We have discussed how we facilitate the complex integration among different services involved in the performance analysis of a Grid workflows. Part of the work mentioned in this paper has been discussed in [19]. Technical details and usage of DIPAS are presented in [20, 22]. Further information about the K-WfGrid DIPAS and its implementation can be obtained from [2].

Acknowledgements. The work described in this paper is supported by the European Union through the IST-2002-511385 project K-WfGrid. We are grateful to the support of our colleagues in the K-WfGrid consortium and Simon Ostermann.

References

1. Apache Tomcat, `http://tomcat.apache.org/`.
2. DIPAS (Distributed Performance Analysis Service for Grid Workflows), `http://www.dps.uibk.ac.at/projects/kwfgrid/`.
3. K-WfGrid GEMINI, `http://gemini.icsr.agh.edu.pl/`.
4. K-WfGrid GOM (Grid Organizational Memory), `http://gom.kwfgrid.net/web/space/Welcome`.
5. K-WfGrid GWES (Grid Workflow Execution Service), `http://www.gridworkflow.org/kwfgrid/gwes/docs/`. Last accessed: 5 June, 2007.
6. Ontogrid project, `http://www.ontogrid.net`.
7. The K-WfGrid Project. `http://www.kwfgrid.eu`. Last access: 06 June, 2007.
8. WebLogic Process Integrator Overview, `http://edocs.beasys.com/wlpi/wlpi11/studio/index.htm`.
9. Bartosz Balis, Hong-Linh Truong, Marian Bubak, Thomas Fahringer, Krzysztof Guzy, and Kuba Rozkwitalski. An Instrumentation Infrastructure for Grid Workflow Applications. In *International Symposium on Grid Computing, High-Performance and Distributed Applications (GADA06)*, LNCS, 2-3 November 2006.
10. Peter Brunner, Hong Linh Truong, and Thomas Fahringer. Performance monitoring and visualization of grid scientific workflows in askalon. In Michael Gerndt and Dieter Kranzlmüller, editors, *HPCC*, volume 4208 of *Lecture Notes in Computer Science*, pages 170–179. Springer, 2006.
11. Ian Foster el al. Modeling Stateful Resources with Web Services. Specification, Globus Alliance, Argonne National Laboratory, IBM, USC ISI, Hewle tt-Packard, January 2004.
12. Globus Project. `http://www.globus.org`.
13. GridSphere Portal Framework. `http://www.gridsphere.org`.
14. P. Kacsuk, G. Dozsa, J. Kovacs, R. Lovas, N. Podhorszki, Z. Balaton, and G. Gombas. P-GRADE: a Grid Programming Environment. *Journal of Grid Computing*, 1(2):171–197, 2003.
15. OWL Web Ontology Language Reference. `http://www.w3.org/TR/owl-ref/`.
16. Wim De Pauw, Michelle Lei, Edward Pring, Lionel Villard, Matthew Arnold, and John F. Morar. Web services navigator: Visualizing the execution of web services. *IBM Systems Journal*, 44(4):821–846, 2005.
17. Taverna, `http://taverna.sourceforge.net/`. Last access: 06 June, 2007.
18. Hong-Linh Truong. Performance Service Interfaces and Data Representation – Developer Manual, October 2006. K-WfGrid Deliverable. `http://www.dps.uibk.ac.at/projects/kwfgrid/deliverables/KWF-WP3-UIBK-v1.0-DRDeveloperManual.pdf`.

19. Hong-Linh Truong, Peter Brunner, Thomas Fahringer, Francesco Nerieri, Robert Samborski, Bartosz Balis, Marian Bubak, and Kuba Rozkwitalski. K-WfGrid Distributed Monitoring and Performance Analysis Services for Workflows in the Grid. In *2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, The Netherlands, 4-6 Dec 2006. IEEE Computer Society.
20. Hong-Linh Truong, Peter Brunner, Vlad Nae, and Robert Samborski. Performance Analysis Service – Developer Manual, October 2006. K-WfGrid Deliverable. `http://www.dps.uibk.ac.at/projects/kwfgrid/deliverables/KWF-WP3-UIBK-v1.1-PASDeveloperManual.pdf`.
21. Hong-Linh Truong and Thomas Fahringer. SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid. *Scientific Programming*, 12(4):225–237, 2004. IOS Press.
22. Hong-Linh Truong, Robert Samborski, and Peter Brunner. Performance Analysis Service – User Manual, October 2006. K-WfGrid Deliverable. `http://www.dps.uibk.ac.at/projects/kwfgrid/deliverables/KWF-WP3-UIBK-v1.1-PASUserManual.pdf`.
23. Hong-Linh Truong, Robert Samborski, and Thomas Fahringer. Towards a framework for monitoring and analyzing qos metrics of grid services. *e-science*, 0:65, 2006.
24. XML Path Language. `http://www.w3.org/TR/xpath.html`.
25. Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3-4):171–200, September 2005.

GEMINI: Generic Monitoring Infrastructure for Grid Resources and Applications

Bartosz Balis[1], Marian Bubak[1,2], and Bartłomiej Łabno[2]

[1] Institute of Computer Science, AGH, Krakow, Poland
[2] Academic Computer Centre CYFRONET AGH, Krakow, Poland

Abstract
We present GEMINI, a Generic Monitoring Infrastructure for the Grid. We focus on monitoring of distributed Grid workflows with GEMINI, the most challenging case for monitoring. After presenting GEMINI architecture, and its usage in the K-Wf Grid Project, we describe the problem of distributed collection of monitoring data concerning a monitored workflow, including a proposition of a distributed monitoring data collection protocol. We also describe and propose a solution for the problem of correlation of workflow monitoring data. We conclude with the example of monitoring of the Coordinated Traffic Management workflow.

1 Introduction

This paper presents GEMINI (GEneric Monitoring INfrastructure) – a monitoring framework for the Grid. Unlike most existing monitoring solutions for the Grid [9] [15], we propose an integrated monitoring system for arbitrary data sources, notably infrastructure and applications. We provide a comprehensive support for monitoring of Grid workflows, including a standard instrumentation service for fine-grained, dynamic and selective instrumentation. GEMINI also defines standard interfaces, protocols and data representations, based on XML with semantic ontology-based annotations, which are indispensable elements in a modern computation infrastructure, oriented towards loosely-coupled services and knowledge.

Many aspects of GEMINI have been presented in our previous publications [1] [3] [18]. This paper concentrates on the aspects of monitoring of distributed Grid workflows with GEMINI not presented so far. Monitoring of Grid workflows is the most challenging case for monitoring, both conceptually and technically. Grid Workflows are (1) distributed across multiple sites, (2) loosely coupled in space and time, (3) composed of multiple computation layers, (4) heterogeneous with respect to the programming language, execution platform and virtualization technology. All those characteristics are challenging from the point of view of monitoring of running workflows (aka *experiments*). Those challenges concern data collection and correlation, instrumentation, and standardization of interfaces, protocols and data representations to provide a layer of abstraction on top of different technologies, programming languages and platforms. In this paper, we focus on workflow monitoring data collection and correlation.

In Section 2 the architecture of GEMINI is presented, while in Section 3 a sample deployment of GEMINI in the K-Wf Grid Project is described. Section 4 focuses on the collection of monitoring data in GEMINI. Section 5 describes the problem of workflow monitoring data correlation. A monitoring example for a Coordinated Traffic Management (CTM) workflow is presented in Section 6.

2 GEMINI Architecture

Architecture of GEMINI is shown in Fig. 1. A *Monitor* is an entity one of which is usually deployed at each site, and which exposes monitoring system functionality. Monitors manage underlying *Sensors* and *Mutators* whose task is the actual collection of monitoring events and performing manipulations on monitored objects, respectively. Monitoring events are generated by the *instrumentation* placed in the code of applications. A *Directory Service* is an external system to which Monitors can advertise themselves as producers of monitoring data for specific resources. Consumers refer to the Directory Service to discover appropriate Monitors.

Sensors and Mutators are collectively referred to as Local Monitors, as they usually reside locally with respect to the monitored resources. Their presence is justified by both functional and non-functional reasons:

1. Some operations might not be feasible to perform remotely, e.g., process manipulations or access to some process-related information.
2. When efficiently implemented, local monitors improve performance of data collection, e.g. by buffering and aggregation of events. The additional processing caused by another process on the node is easily outweighed by the elimination of other processing, e.g., the execution of network protocol stacks.

There are three interfaces involved in the monitoring system: (1) the producer interface, (2) the consumer interface, (3) and the mutator interface. The producer interface is used to request the monitoring data. This data can be pushed to consumers via the consumer interface. The mutator interface can perform various types of manipulations. In GEMINI, we support one type of mutator functionality expressed by the *instrumentation* interface.

A Monitor exposes two services – the Monitoring Service and the Instrumentation Service – which implement the producer and the instrumentation interface, respectively. Those services are implemented as Globus Toolkit 4 WSRF ones[1]. The producer and instrumentation interfaces are also implemented by Sensors and Mutators, respectively. However, they are exposed in a more tightly coupled technology, ICE (Internet Communication Engine)[2] in order to improve performance of data transfers. Also for performance reasons, the consumer interface is implemented in ICE.

The producer interfaces is based on a producer protocol defined by PDQS (Performance Data Query Subscribe), an XML-based language for specification

[1]WS-Resource Framework, `http://www.globus.org/wsrf/`

[2]`http://www.zeroc.com`

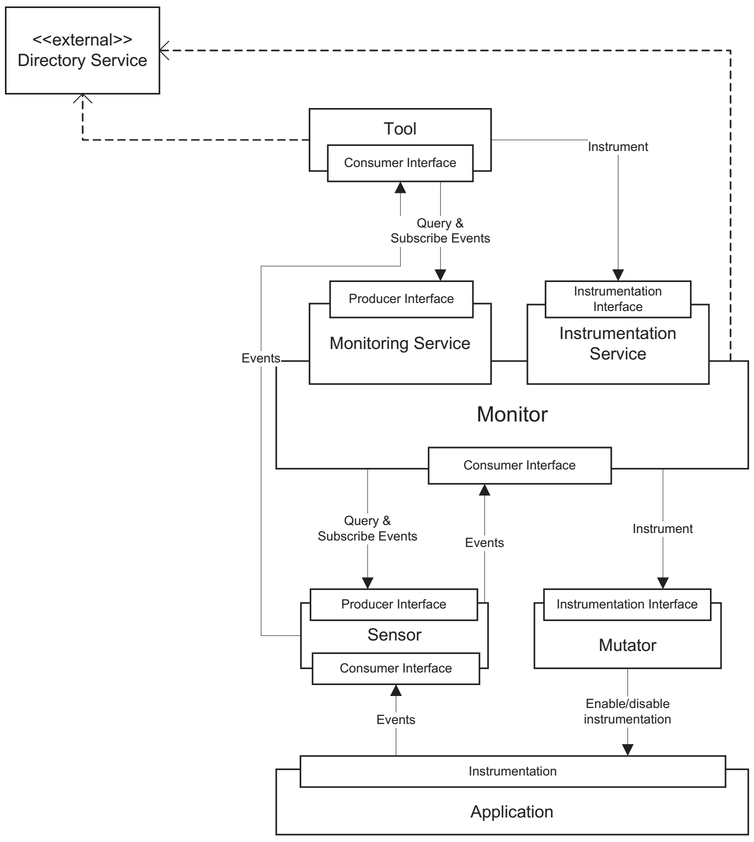


Fig. 1: Architecture of the GEMINI monitoring infrastructure.

of monitoring data requestes. A monitoring request typically contains the following elements:

- type of request (query or subscribe)
- idenfitication of the monitored resource
- monitoring data type
- subscription period (only for the subscribe request)

PDQS allows to express two types of requests: *query* and *subscribe.* A query for monitoring data synchronously returns the requested data. A subscription request, in contrast, allows to specify a subscription period in which the monitoring data will be asynchronously delivered whenever it occurs, via the consumer interface.

The monitoring data in the subscription mode is in fact delivered through Publish/Subscribe subscription Mediators (P/S Mediators). A single mediator is usually assigned to serve as a communication channel for either a given monitored workflow, or all data from a given site. Mediators are implemented as ICE Storm

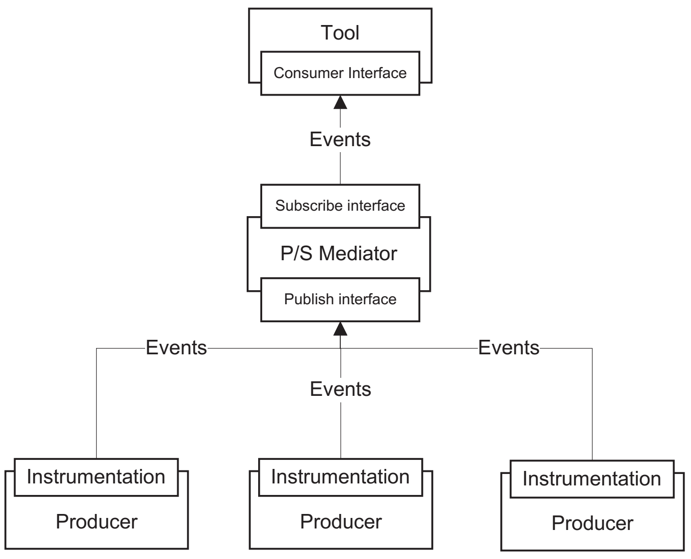


Fig. 2: Data transfer with a publish/subscribe mediator.

services which allow for event subscription and notification based on topics. Fig. 2 presents an example of data transfer with a mediator.

The role of the mediator is twofold:

1. As with all publish/subscribe channels, to decouple data producers from data consumers.
2. As a data-transfer relay component. Since the monitored data is, due to performance reasons, transfered by means of low-level, tightly-coupled communication channels, communication between producers and consumers across firewalls or private networks may pose a problem. In such cases, the mediator channels could be deployed at a neutral, public location to make the communication possible, at some performance penalty.

The instrumentation interface realizes the instrumentation protocol defined by the WIRL language (Workflow Instrumentation Request Language). WIRL, as PDQS, is an XML-based language which can be used to specify instrumentation requests. An instrumentation request consists of the following elements:

- type of request (get standard intermediate representation, enable/disable instrumentation)
- specification of application's processing unit to which the request should be applied
- instrumentation specification (for enable/disable instrumentation): which code regions should be affected and the position where the instrumentation should be applied (before or after a code region)

The instrumentation is dynamic and selective, i.e. we can dynamically request to enable or disable the instrumentation for selected parts of the applica-

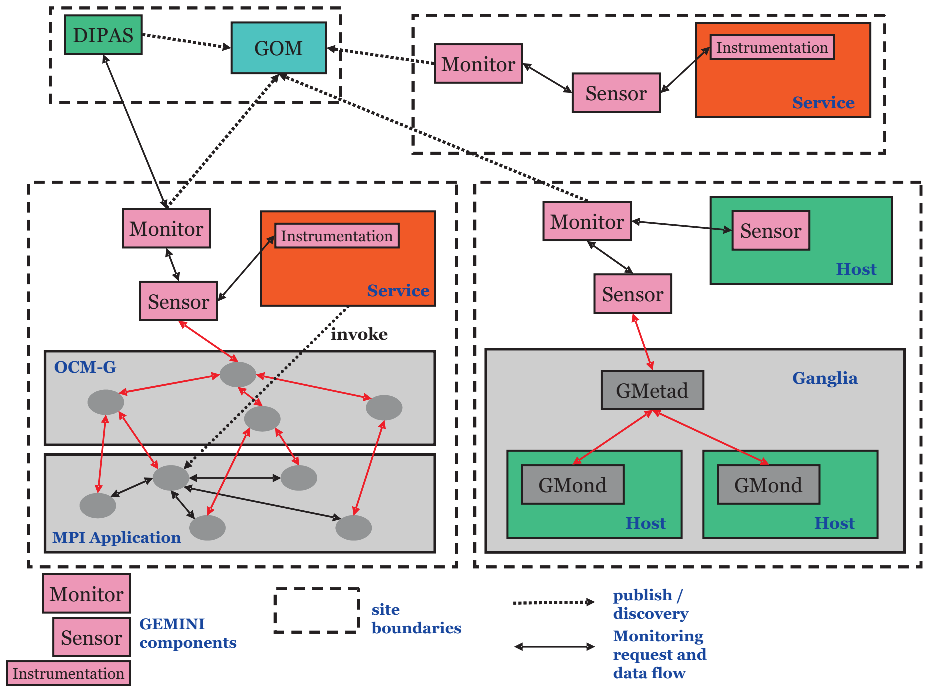


Fig. 3: Sample deployment of GEMINI in K-Wf Grid.

tion. In order to enable this, the instrumentation interface allows to obtain an abstract specification of the application's structure expressed as a list of code regions in an XML-based language SIRWF (Standard Intermediate Representation for WorkFlows), an extension of SIR [16]. More information about instrumentation in GEMINI can be found in [3].

3 GEMINI in K-Wf Grid

In Fig. 3, a sample deployment of GEMINI in the K-Wf Grid Project[3] [7] is depicted. In this project, GEMINI was used to collect monitoring data about Grid infrastructure and workflows. The data was collected from such diverse sources, as Ganglia system for monitoring of infrastructure elements, instrumented services being parts of workflows or middleware, and legacy MPI applications invoked from services and monitored via the OCM-G [2] monitoring system. Grid Organization Memory (GOM) [13] was used as a directory service.

Consumers of the monitoring data included the DIPAS portal [18], which uses it for performance analysis of workflows, and Knowledge Assimilation Agent (KAA) [4] which needs the monitoring data in order to extract knowledge about

[3]`http://www.kwfgrid.eu`

running workflows, e.g. for prediction of the execution time of individual service instances.

4 Collection of Monitoring Data

This section describes how on-line collection of monitoring data for distributed workflows is realized in GEMINI. As far as workflows are concerned, there are some special circumstances and requirements concerning the collection of workflow monitoring data.

First of all, workflows (especially those running on the Grid) are *distributed* and *dynamic* resources. The latter characteristic means that distinct parts of a workflow emerge and disappear during the workflow's execution. As a result, new producers of the workflow's monitoring data dynamically emerge at an unpredictable time and location.

Second, it would be convenient for clients to request *all* monitoring data concerning a workflow, including this generated in the past (before the client's request was issued) and that which will come in the future.

Those problems would not be severe unless we required *on-line* collection of monitoring data, i.e. collection in which we are ensured to receive monitoring events with a small delay with respect to time when they actually occurred. If we did not require that, the traditional Grid Monitoring Architecture with producers, consumers, and a directory service [17] would be perfectly sufficient. The producers would register in the directory service as they would emerge. They would also buffer the monitoring data in order to return it in the case some clients asked for the past workflow events. The consumers would periodically check the directory service for new producers in order to subscribe to them in the case they emerge. Upon subscription they would receive archive data gathered in buffers, and also subsequent events until the end of subscription.

However, it has been pointed out that traditional directory services are not suitable for frequently changing resources [8]. While this is partially because of technical reasons (slow technologies), in part it also is due to a conceptual flaw of directory services related to their 'inactive' interfaces which force a client interested in a resource to issue an explicit request. The authors propose thus a *proactive directory service* which supports 'push' style of communication allowing clients to receive notifications about resource changes. While this is an interesting proposal which could present an adequate solution, the research presented in the mentioned work is still not fully mature. We have therefore concentrated on an alternative solution.

The requirement for on-line collection of workflow monitoring events can be essentially brought down to the problem of **fast and automatic resource discovery**. Automatic resource discovery means that the consumers are automatically notified whenever new producers of monitoring data for a given workflow appear, and a fast delivery of this notification is guaranteed.

Fig. 4 presents a distributed monitoring data collection protocol. For simplicity, only producers and consumers are shown, not the real components of

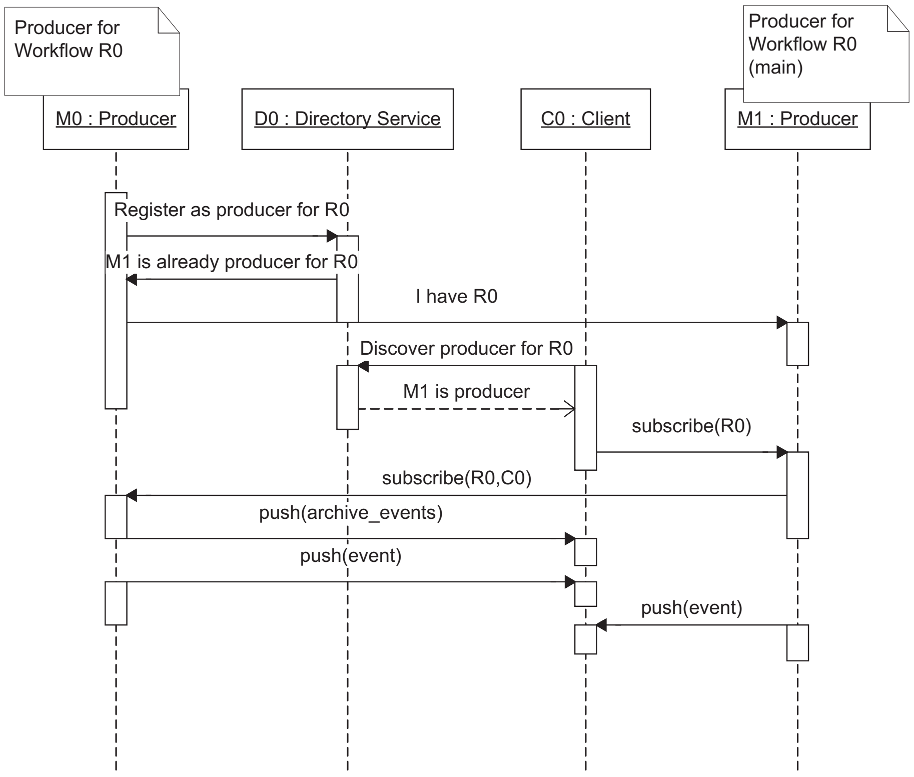


Fig. 4: Distributed monitoring data collection protocol.

GEMINI architecture (Monitors, Sensors, P/S Mediators).

In this example of monitoring a workflow $R0$, we have two producers of monitoring data – $M0$ and $M1$. A Directory Service $D0$ in this protocol serves a special purpose as a kind of *data producer broker*. Both $M0$ and $M1$ attempt to register in $D0$ as producers for monitoring data for $R0$. However, only the first one to come is actually elected as the 'main' producer, $M1$ in this example (registration not shown). The others – $M0$ in this case – are returned information that the producer is already $M1$. Thus, $M0$ reports to $M1$ that it is also a producer of monitoring data for $R0$. When a client $C0$ discovers a producer for workflow $R0$, it gets a reference to $M1$ and subscribes in it. $M1$ is responsible to forward the subscription request to all remaining producers with a return handle to client $C0$. The other producers ($M0$) start sending workflow events to the consumer, beginning with the archive events stored in their local buffers.

In other words, in this protocol, resource discovery is partially up to an external directory service, and partially within responsibilities of the monitoring infrastructure itself. Thanks to this solution, the described protocol has the following advantages:

- The client needs to discover data producers only once.
- The client also needs to connect to a single producer only.
- Automatic resource discovery is as fast as the request to the Directory Service. This is thus faster than a registration of a new resource in and a notification from the Directory Service. In fact, the Directory Service could be replaced by a specialized, fast data producer broker.
- Though the Directory Service here must ensure mutually exclusive registration attempts (and thus it probably should be centralized), it will not be a bottleneck, since the registration events are relatively rare.
- Since the first producer of data for a given workflow is elected as the 'main' producer, the overall scalability is guaranteed provided that workflows will start execution at different, random places.
- Clients will obtain the complete workflow monitoring data, including the archive one, i.e. that which was produced before the client made a subscription request.

5 Correlation of Monitoring Data

The term *correlation* in the context of distributed systems has at least two important meanings:

1. In distributed systems that use asynchronous messaging the problem of matching a request with a response is called the *correlation problem.* The solution is to use a unique *correlation identifier*, which should be the same in the request and in the matching response. This solution is known as the *Correlation Identifier Pattern* ([12], pp. 163-170).
2. In event notification or monitoring systems, the term *event correlation* denotes the process of detecting event patterns (aka *global events*, *composite events*) in a number of seemingly unrelated events. For example, a detected event pattern may reveal a possibility of a security attack.

In the context of workflows, correlation is mentioned in [5]. In general, a process P (i.e. a workflow) may be orchestrated by multiple orchestration engines P_k. The correlation problem is thus, according to the authors, *to associate different parts of process P executing in different P_k*. However, distributed orchestration engines are not of our concern. We focus on a different problem. For monitoring of workflows we shall define the correlation problem as follows: **Workflow monitoring data correlation problem** is **to associate different pieces of monitoring data as parts of data related to the same experiment**.

This problem has been, of course, partially recognized. In [11], Grid Workflow Identifiers (GID) are proposed. The GIDs should be created in a "workflow originator" and "propagated to all workflow components". While this is the right way to go, it does not go far enough. The reason for this is that "workflows" described in this work are not *scientific workflows* composed of services and executed by a generic orchestration engine. Rather, a "workflow" is meant as a sequence of different Grid "services" (such as portal, broker, replica manager) whose coordinated cooperation results in execution of a *job*, which in turn does the actual work, and later the results are retrieved.

This model of a 'flat' grid identifier is **not sufficient for monitoring of scientific workflows**, whose execution needs the cooperation of many Grid services as well, but which are themselves composed of many distinct parts, namely activities, and invoked applications, the last often being separate *legacy jobs*, which may well be parallel jobs, again composed of multiple distinct processes.

The correlation of workflow monitoring data must enable association of different pieces of monitoring data at their different levels in the workflow hierarchy, for example events from "*workflow* $W0$*, activity* $A1$", or "*workflow* $W1$*, activity* $A1$*, legacy job* $L2$". Consequently, for the purpose of monitoring we introduce an *Experiment correlation identifier* which is characterized as follows:

1. **Experiment Correlation IDentifier** (ECID) is an extendable XML document which is passed between different parts of executing scientific workflow, as well as Grid services involved.
2. ECID should be included in each event produced by monitoring and it should reflect the **exact space location** of the event in the executing workflow.
3. ECID is **extended on a handover of workflow's execution control** from P_i to P_k, P_i and P_k being different, **physically distinct**, parts of the executing workflow.
4. Actor responsible for extension of ECID on handing over the workflow execution to task P_k is the **initiator** I_k of P_k; I_k is normally either the orchestration engine or another workflow task P_i which invokes a subtask P_k.

An example of an ECID for an experiment task's subtask is shown below:

```
<ecid>
  <experiment id="e1">
      <task id="t1">
         <subtask id="s1"></subtask>
      </task>
  </experiment>
</ecid>
```

Experiment correlation identifier is also **important for request-reply correlation**. A client could request a subset of monitoring events that belong only to a certain part of the workflow (e.g. MPI job L_1 invoked from activity A_3). The hierarchical ECID enables the monitoring system to deliver only this subset of events to the client, instead of making the client responsible for filtering.

The technical problem how to pass the Experiment Correlation Identifier to workflow parts remains. Unfortunetaly, this problem is technology-dependent. In the case of workflow activities implemented as web services this is actually a specific case of a more general problem of *passing context to web services.* Current web service implementations do not support context information in a standard way. For pure, stateless web services, the only information passed to an invoked service is via operation's parameters. There are a few possibilities to pass context information, as discussed in [6], for example:

- by extending service's interface with additional parameter for context information,
- by extending the data model of the data passed to a service,
- by inserting additional information in the SOAP header, and reading it from within the service.

None of those methods is nice and clean. The first one forces to extend the interface of the service which is the least transparent option. The second one is not transparent for the end user, either, and additionally is service-specific. The third one is only possible if access to SOAP header is feasible, and it depends on the implementation of the web service container. An additional option is to use the state provided by a service, but this only works for services that support state, e.g. WSRF-based ones.

However, the problem of context-aware web services has been recognized. A Web Service Context Specification has been proposed [19] whose aim is to deal with Web Service context in a standardized way. This specification supports passing the WS context in the SOAP header and defines standardized XML structures to do it.

Therefore, it is natural for us to also follow the SOAP header approach. The workflow enactment engine used in our case – GWES (Grid Workflow Execution Service) [14] inserts additional information to SOAP headers which is accessed in the services (from instrumentation code) to obtain workflow and activity identifiers.

For the legacy jobs, ECID is passed as command-line parameters. However, in the case of MPI applications command line parameters might be passed only to the master process. There are a few possibilities in this case. For example, the master process can be instrumented to broadcast the correlation identifier as the first operation after `MPI_Init`. This is the solution we have chosen to employ.

6 Monitoring Example – Coordinated Traffic Management Workflow

To demonstrate workflow monitoring, we have chosen the Coordinated Traffic Management (CTM) workflow constructed from application services provided by Softeco Sismat within the K-WF Grid Project. This application targets the computation of the emission of traffic air pollutants in an urban area and has been developed in tight collaboration with the Urban Mobility Department of the Municipality of Genoa, which provided a monolithic implementation of the model for the pollutant emission calculations, the urban topology network and real urban traffic data. The CTM application workflow has been divided into several different steps in order to allow the semi-automatic composition of services and the definition of a set of ontologies which describe the CTM domain

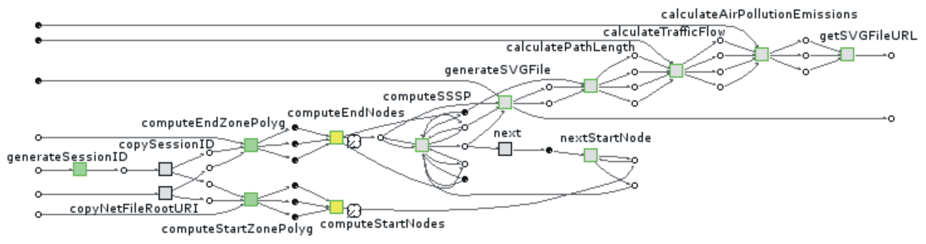


Fig. 5: CTM workflow.

and feed the system with the information needed for the proper selection and execution of services [10].

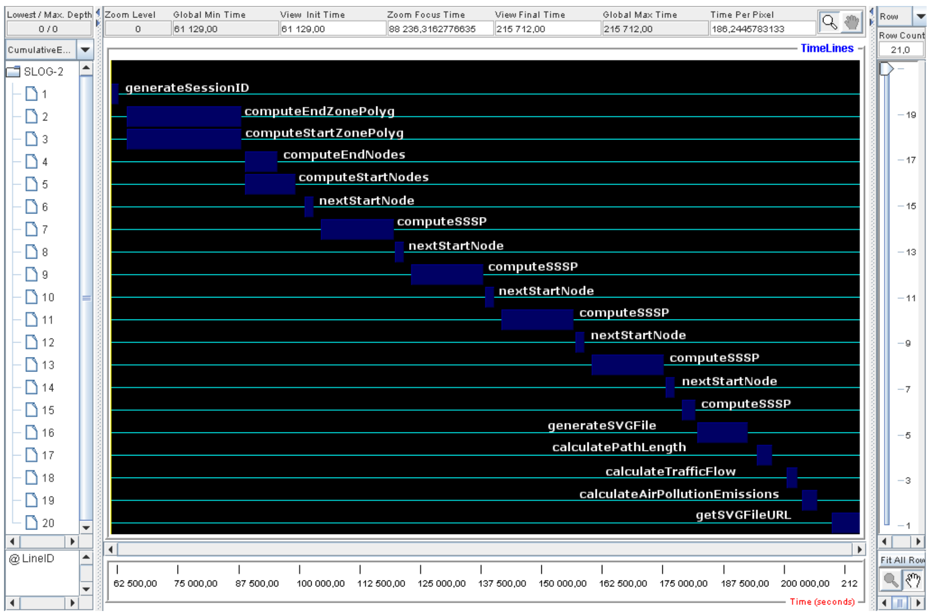
The main CTM application functionalities are best route, traffic flow and air pollutant emissions calculations. Data graphical representation in SVG format is also supported. For monitoring, a complex use case was used. It consisted of several executable transitions and three control transitions (Fig. 5).

The first activity in the workflow is the generation of a session ID. Next, there are two activities done in parallel – the computation of start and end zone district polygons. Subsequently, node coordinates for start and end are computed, also in parallel. After that, a set of calculations is done for nodes computed in earlier activities. Finally, computations responsible for calculating path length, traffic flow and air pollutants emission follow. Results are also written to the SVG format file, which is done in one of the activities.

During running of the above scenario monitoring data were acquired by instrumentation of the worklow enactment engine and workflow activities. For each activity, several events were produced: when it was initialized, created, went to the active state, at start and end of the actual running phase, and when it has completed. Each of these events has a precise time of occurence. Activities such as initialization, creation, activation and completion should be treated as a point in time. The running state is treated as a period of time.

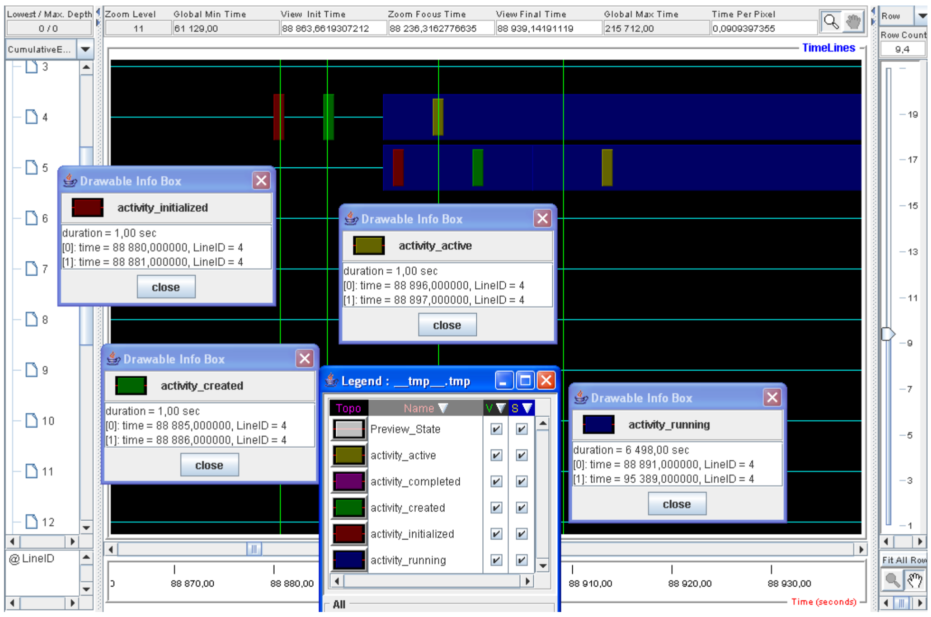
For visualization of the monitoring results, we have used the Jumpshot tool[4]. To this end, events collected from GEMINI were translated to Jumpshot's SLOG-2 format. We can observe how the workflow was executed, how much time each activity has taken, and when it was invoked. Fig. 6 presents a global view showing all activities. However, due to the time scale, only the running periods can be seen. Fig. 7 presents a zoom of a particular diagram section to show more details – individual events can be seen now. Additionally, windows describing individual bars (representing events and periods) are popped up.

[4]See `http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm`



Lowest / Max. Depth: 0 / 0
Zoom Level: 0
Global Min Time: 61 129,00
View Init Time: 61 129,00
Zoom Focus Time: 88 236,3162776635
View Final Time: 215 712,00
Global Max Time: 215 712,00
Time Per Pixel: 186,2445783133
Row Count: 21,0

Fig. 6: Monitoring results – global view.



Lowest / Max. Depth: 0 / 0
Zoom Level: 11
Global Min Time: 61 129,00
View Init Time: 88 863,6619307212
Zoom Focus Time: 88 236,3162776635
View Final Time: 88 939,14191119
Global Max Time: 215 712,00
Time Per Pixel: 0,0909397355
Row Count: 9,4

Drawable Info Box
activity_initialized
duration = 1,00 sec
[0]: time = 88 880,000000, LineID = 4
[1]: time = 88 881,000000, LineID = 4

Drawable Info Box
activity_active
duration = 1,00 sec
[0]: time = 88 896,000000, LineID = 4
[1]: time = 88 897,000000, LineID = 4

Drawable Info Box
activity_created
duration = 1,00 sec
[0]: time = 88 885,000000, LineID = 4
[1]: time = 88 886,000000, LineID = 4

Drawable Info Box
activity_running
duration = 6 498,00 sec
[0]: time = 88 891,000000, LineID = 4
[1]: time = 95 389,000000, LineID = 4

Legend : __tmp__.tmp

Topo	Name	V	S
⬜	Preview_State	✓	✓
🟨	activity_active	✓	✓
🟪	activity_completed	✓	✓
🟩	activity_created	✓	✓
🟥	activity_initialized	✓	✓
🟦	activity_running	✓	✓

Fig. 7: Monitoring results – detailed local view.

7 Summary

We have presented the use of the GEMINI infrastructure for monitoring of Grid workflows. We have explained the role of GEMINI in the K-Wf Grid Project. Next, we have focused on the problems of distributed data collection, and data correlation. Currently we are working on an ontology-based monitoring data model which can be used to represent a meaningful information about previously running experiments, and the usage of this information to extract knowledge about the experiments and the computing infrastructure.

Acknowledgements. The work described in this paper is supported by the European Union through the IST-2002-511385 project K-WfGrid, and the Foundation for Polish Science within the Domestic Grant for Young Scientists.

References

1. Balis, B., Bubak, M., Dziwisz, J., Truong, H.L., Fahringer, T.: Integrated Monitoring Framework for Grid Infrastructure and Applications. In: Innovation and the Knowledge Economy. Issues, Applications, Case Studies, Ljubljana, Slovenia, IOS Press (2005) 269–276
2. Balis, B., Bubak, M., Radecki, M., Szepieniec, T., Wismüller, R.: Application Monitoring in CrossGrid and Other Grid Project s. In: Grid Computing. Proc. Second European Across Grids Conference, Nicosia, Cyprus, Springer (2004) 212–219
3. B. Balis, H.-L. Truong, M. Bubak, T. Fahringer, K. Guzy, and K. Rozkwitalski. An Instrumentation Infrastructure for Grid Workflow Applications. In R. Meersman and Z. Tari et al., editors, Proc. OTM 2006 Conferences, volume 4276 of Lecture Notes in Computer Science, pages 1305-1314, Montpellier, France, November 2006. Springer.
4. Balogh Z., Gatial E., Laclavik M., Maliska M., Hluchy L.: Knowledge-based Runtime Prediction of Stateful Web Services for Optimal Workflow Construction. Proc. of 6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005, R.Wyrzykowski et.al. eds., 2006, LNCS 3911, Springer-Verlag, pp. 599-607, ISSN 0302-9743, ISBN 3-540-34141-2. Poznan, Poland.
5. G. Brown and R. Carpenter. Succesful Application of Service-Oriented Architecture Across the Enterprise and Beyond. *Intel Technology Journal*, 8(4):345–359, 2004.
6. S. Brydon and S. Kangath. Web Service Context Information. `https://bpcatalog.dev.java.net/nonav/soa/ws-context/index.html`, 2005. Accessed 20.06.2007.
7. M. Bubak, T. Fahringer, L. Hluchy, A. Hoheisel, J. Kitowski, S. Unger, G. Viano, K. Votis, and K-WfGrid Consortium: K-Wf Grid – Knowledge based Workflow system for Grid Applications. In Proc. Cracow Grid Workshop 2004, p.39, Academic Computer Centre CYFRONET AGH, ISBN 83-915141-4-5, Poland 2005
8. F. E. Bustamante, P. Widener, and K. Schwan. Scalable directory services using proactivity. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–12, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

9. A. Cooke, A. Gray et al., R-GMA: An Information Integration System for Grid Monitoring. In *Proc. Tenth International Conference on Cooperative Information Systems*, volume 2888 of *Lecture Notes in Computer Science*, pages 462–481. Springer, 2003.
10. T. Gubala, D. Harezlak, M. Bubak, M. Malawski, Semantic Composition of Scientific Workflows Based on the Petri Nets Formalism. Proc. 2nd IEEE International Conference on e-Science and Grid Computing, (Avaialble only on CD-ROM), IEEE Computer Society Press 2006.
11. D. K. Gunter, K. R. Jackson, D. E. Konerding, J. Lee, and B. Tierney. Essential Grid Workflow Monitoring Elements. In Hamid R. Arabnia and Jun Ni, editors, *Proc. 2005 International Conference on Grid Computing and Applications, GCA 2005*, pages 39–45, Las Vegas, Nevada, USA, June 2005. CSREA Press.
12. G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, October 2003.
13. B. Kryza, J. Pieczykolan J. Kitowski: Grid Organizational Memory: A Versatile Solution for Ontology Management in the Grid. In Proc. of 2nd International Conference on e-Science and Grid Computing, Dec. 4-6, 2006, Amsterdam, Netherlands, (C) IEEE Computer Society Press. ISBN 0-7695-2734-5
14. Neubauer, F., Hoheisel, A., Geiler, J.: Workflow-based Grid Applications. Future Generation Computer Systems **22** (2006) 6–15
15. N. Podhorszki, Z. Balaton, G. Gombas. Monitoring Message-Passing Parallel Applications in the Grid with GRM and Mercury Monitor. In: Grid Computing. Proc. 2nd European Across Grids Conference, Nicosia, Cyprus, January 2004, Springer.
16. Seragiotto, C., Truong, H.L., Fahringer, T., Mohr, B., Gerndt, M., Li, T.: Standardized Intermediate Representation for Fortran, Java, C and C++ Programs. Technical report, Institute for Software Science, University of Vienna (2004)
17. B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A grid monitoring architecture. Technical Report GWD-PERF-16-2, Global Grid Forum, January 2002.
18. Truong, H.L., Balis, B., Bubak, M., Dziwisz, J., Fahringer, T., Hoheisel, A.: Towards Distributed Monitoring and Performance Analysis Services in the K-WfGrid Project. In: Proc. PPAM 2005 Conference, Poznan, Poland, Springer (2006) 157–163
19. Web Services Context Specification (WS-Context) Version 1.0. `http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.pdf`, 2006. Downloaded 27.03.2007.

Grid Organizational Memory – Semantic Framework for Metadata Management in the Grid

Bartosz Kryza[1], Jan Pieczykolan[1], Marta Majewska[2], Renata Slota[2], Marian Babik[3], Adrian Toth[3], Jacek Kitowski[1,2], and Ladislav Hluchy[3]

[1] Academic Computer Centre Cyfronet AGH, Krakow, Poland
[2] Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland
[3] Insititue of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia

Abstract
The paper presents detailed description of the Grid Organizational Memory architecture, a justification of its applicability in the context of K-Wf Grid project, and gives a comparison of GOM with other existing semantic metadata solutions. Such additional components of Grid Organizational Memory as administration interface, Protege extension and semi-automatic service annotation tool are also presented.

1 Introduction

Grid Organizational Memory [1] is the central information and knowledge source in the K-Wf Grid system [2]. It manages various kinds of metadata stored in the form of OWL ontologies.

Grid Organizational Memory has been evaluated within the K-Wf Grid platform by supporting both middleware level functionality such as infrastructure monitoring as well as application level scenarios [3, 4], especially the ones involved in workflow composition process.

The main motivation for using ontologies to annotate Grid resources was to unify the metadata in the Grid environments through the use of one single formalism – in this case Web Ontology Language. This allows for much richer and complex annotation of resources which can span all aspects of the Grid (e.g. use metadata about data or hardware resources to describe services) in the same language (e.g. OWL). Whats more, unified metadata model allows application of various emerging tools especially ontology reasoners which can grately improve the processing of such information.

2 Grid Organizational Memory Architecture and Implementation

The main functionality of the GOM knowledge base is oriented towards supporting the semi-automatic workflow composition from services available in the Grid [5]. This includes management of semantic descriptions of resources, data and services as well as ontological representation of knowledge about the domains to which the Grid is applied, which enable sufficiently rich semantic annotations of data and services. The managed ontologies can be stored in persistent storage and queried by standard query languages for RDF and OWL. Apart from the knowledge base itself GOM has several accompanying tools that support several typical use cases in the K-Wf Grid environment.

2.1 Architecture

Grid Organizational Memory is a knowledge base which was developed in order to support the ontology separation scheme [6] (see Fig. 1). The first design choice was to distribute the GOM in a similar fashion to how the ontology components are separated [7]. The ontologies defining the Grid metadata can be divided into ontology components according to the ontology separation scheme. Thus we have a set of GOM Engine components, each running separately and managing some part of the ontology space (e.g. ServiceRegistry for CTM application). The access to the GOM Engines is provided through Proxy interfaces which contact Engine Manager component which acts as a router for locating proper GOM Engine instance for handling particular request. The administrators have control over the GOM distributed infrastructure through the web-based GOM Admin interface. The clients can also subscribe for specific events that are received by some GOM Engine. Regular users can browse and manipulate the ontologies through a proper Protege extension.

2.2 GOM Engine

Ontology components can be managed by a separate instance of GOM Engine component, possibly distributed, which can be configured with such options as persistent store and some DL reasoner [8]. This makes it possible to choose the best configuration for each ontology component. Each ontology component can be also published under proper URL thus giving clients easy access to the ontology. Each GOM Engine accepts two different kinds of requests, namely Events and Queries. The events describe a change or a set of changes that are to be applied to the local ontology model called the State Model. The events can include creation of some resource description (e.g. a full OWL-S service description), a change (e.g. a change of individual property value) or removal (e.g. a list of individuals to remove). The events are also stored in a log called Event Model which allows to recreate the state of the ontology component from any time from the past. The GOM Engine is based on plug-ins which allows for simple extension of its functionality in terms of handling events, handling queries creating state models with different reasoners and persistent storage systems. It also can be deployed as a standalone service (without the whole infrastructure).

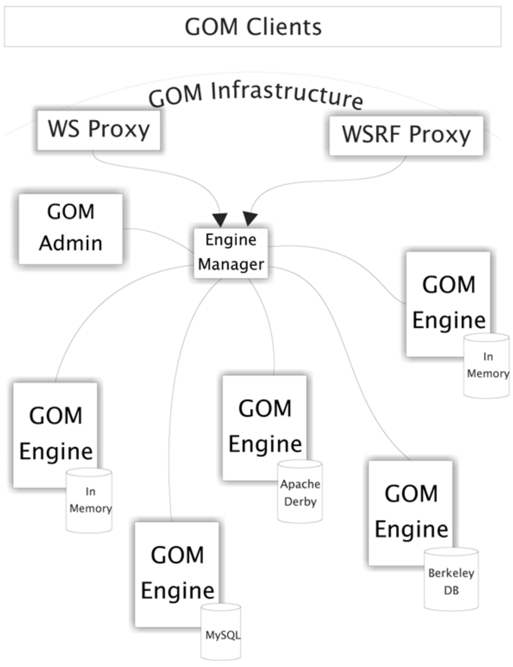


Fig. 1: Overall architecture of GOM.

2.3 Engine Manager

The Engine Manager is the main element of the GOM's architecture. It is responsible for hosting a registry of available Deployer and GOM Engine instances and provide methods facilitating looking up for references to them. It supports such functionality as GOM Engines synchronization, locking, heartbeat mechanism and event routing (see Fig. 2).

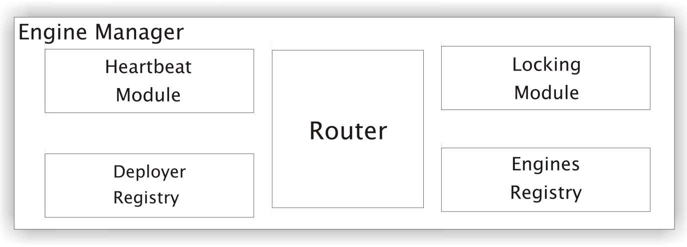


Fig. 2: Architecture of Engine Manager.

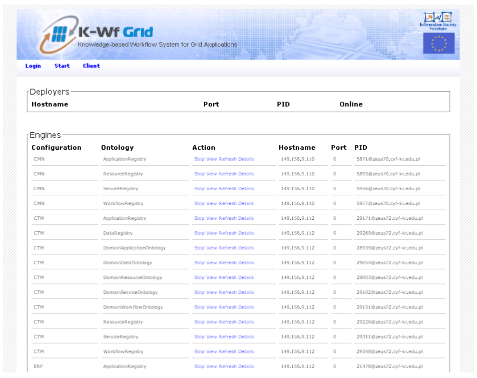
2.4 Event System

The ontologies managed by GOM can be updated by a special event system designed for fine grained modifications of OWL ontologies what makes integration of GOM with other system components easier combined with mechanism of subscription for change notifications. The system consists of 3 kinds of events:

- *Create* – which accepts sets of statements or OWL documents which are directly added to the State Model. It is useful for simple addition of chunks of knowledge (e.g. complete OWL-S service description)
- *Change* – Allows more fine grained change functionality. It is based on change ontology which describes all possible changes that can be applied to OWL ontology (e.g. ClassAddition, IndividualRemoval, IndividualPropertyValueModification, etc.)
- *Remove* – accepts list of statements that should be removed

2.5 GOM Admin

A special web based component (called GOM Admin) has been developed to manage distributed deployments of GOM (see Fig. 3). It allows administrators to start, stop and monitor status of running instances of GOM Engine components.



K-Wf Grid
Knowledge-based Workflow System for Grid Applications
Information Society Technologies
Login
Start
Client

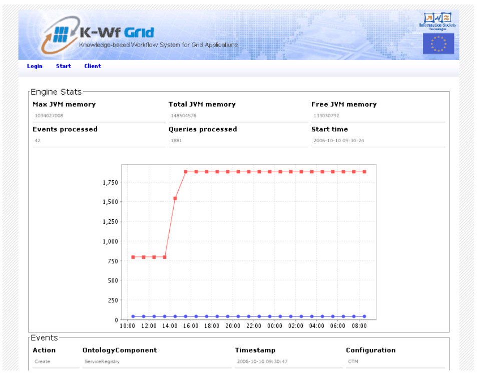
Deployers

Hostname	Port	PID	Online

Engines

Configuration	Ontology	Action	Hostname	Port	PID
CMN	ApplicationRegistry	Stop View Refresh Details	149.156.9.110	0	5871@zeus70.cyf-kr.edu.pl
CMN	ResourceRegistry	Stop View Refresh Details	149.156.9.110	0	5893@zeus70.cyf-kr.edu.pl
CMN	ServiceRegistry	Stop View Refresh Details	149.156.9.110	0	5956@zeus70.cyf-kr.edu.pl
CMN	WorkflowRegistry	Stop View Refresh Details	149.156.9.110	0	5977@zeus70.cyf-kr.edu.pl
CTM	ApplicationRegistry	Stop View Refresh Details	149.156.9.112	0	29171@zeus72.cyf-kr.edu.pl
CTM	DataRegistry	Stop View Refresh Details	149.156.9.112	0	29269@zeus72.cyf-kr.edu.pl
CTM	DomainApplicationOntology	Stop View Refresh Details	149.156.9.112	0	28939@zeus72.cyf-kr.edu.pl
CTM	DomainDataOntology	Stop View Refresh Details	149.156.9.112	0	29054@zeus72.cyf-kr.edu.pl
CTM	DomainResourceOntology	Stop View Refresh Details	149.156.9.112	0	29003@zeus72.cyf-kr.edu.pl
CTM	DomainServiceOntology	Stop View Refresh Details	149.156.9.112	0	29102@zeus72.cyf-kr.edu.pl
CTM	DomainWorkflowOntology	Stop View Refresh Details	149.156.9.112	0	29151@zeus72.cyf-kr.edu.pl
CTM	ResourceRegistry	Stop View Refresh Details	149.156.9.112	0	29226@zeus72.cyf-kr.edu.pl
CTM	ServiceRegistry	Stop View Refresh Details	149.156.9.112	0	29311@zeus72.cyf-kr.edu.pl
CTM	WorkflowRegistry	Stop View Refresh Details	149.156.9.112	0	29348@zeus72.cyf-kr.edu.pl
ERP	ApplicationRegistry	Stop View Refresh Details	149.156.9.112	0	21478@zeus72.cyf-kr.edu.pl

(a) GOM Engine list.



Engine Stats

Max JVM memory	Total JVM memory	Free JVM memory
1034027008	148504576	133030792
Events processed	**Queries processed**	**Start time**
42	1881	2006-10-10 09:30:24

Events

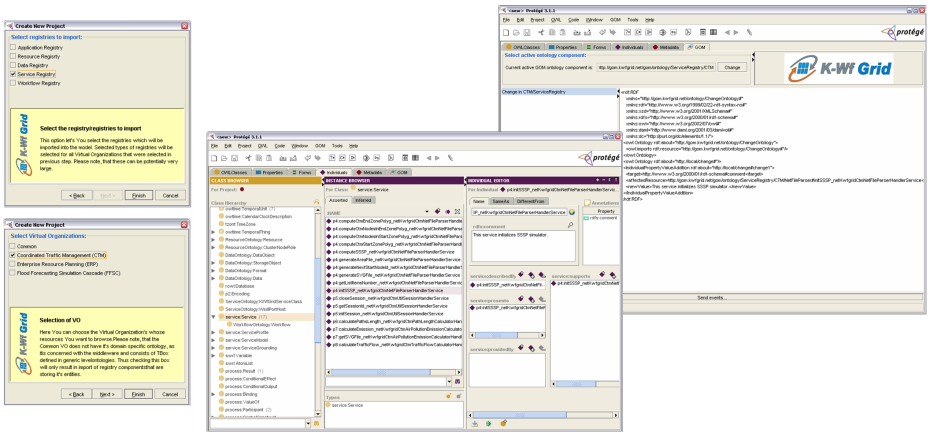
Action	OntologyComponent	Timestamp	Configuration
Create	ServiceRegistry	2006-10-10 09:30:47	CTM

(b) GOM Engine statistics.

Fig. 3: Example GOM Admin screen shots.

2.6 GOM Tab

Another tool is GOM Tab plugin (see Fig. 4) for Protege environment which enables manipulation of ontologies stored in GOM using Protege's advanced user interface and all its plugins for visualization of ontologies. A tool which translates from Common Information Model schema to OWL ontologies enables migration from legacy information systems. The prototype of the tool enables users to generate OWL ontology of CIM schema and convert instances from a CIM repository.


Create New Project

Select registries to import:

- [ ] Application Registry
- [ ] Resource Regisrty
- [ ] Data Registry
- [✓] Service Registry
- [ ] Workflow Registry

Select the registry/registries to import

This option let's You select the registries which will be imported into the model. Selected types of registries will be selected for all Virtual Organizations that were selected in previous step. Please note, that these can be potentially very large.

Create New Project

Select Virtual Organizations:

- [ ] Common
- [✓] Coordinated Traffic Management (CTM)
- [ ] Enterprise Resource Planning (ERP)
- [ ] Flood Forecasting Simulation Cascade (FFSC)

Selection of VO

Here You can choose the Virtual Organization's whose resources You want to browse.Please note, that the Common VO does not have it's domain specific ontology, as itis concerned with the middleware and consists of TBox defined in generic levelontologies. Thus checking this box will only result in import of registry componentsthat are storing it's entities.

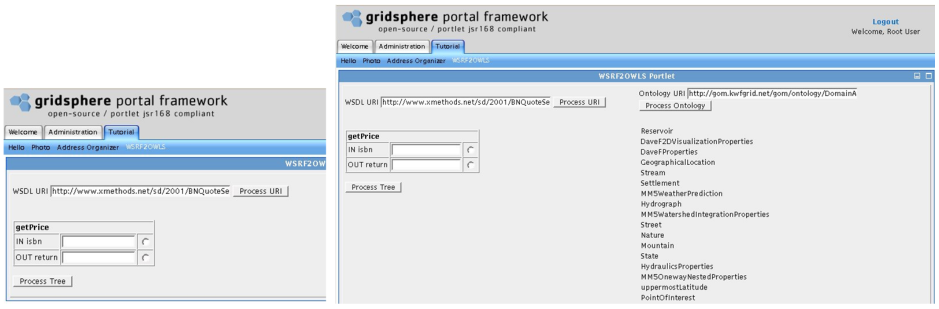
Fig. 4: GOMTab Protege interface.

2.7 WSRF2OWL-S

Several tools that support users in interacting with GOM have been developed. First one, called WSRF2OWL-S [9, 10], has been developed to enable semi-automatic generation of semantic descriptions of services. This tool supports users by generating an OWL-S description of a service from its WSDL definition and a mapping from WSDL to OWL which can be specified in a file or using a GUI integrated into K-Wf Grid GridSphere portal (see Fig. 5).

3 Related Work

Some semantic Grid knowledge bases already exists. Tupelo 2 from NCSA [11], is oriented towards archiving large scale metadata about scientific data. DRAGO [12] provides advanced functionality of reasoning over P2P distributed knowledge bases using C-OWL standard. OMII provides Grimoires [13] which is also a modular knowledge base, oriented towards managing semantic service descriptions in a Grid setting. instanceStore [14] is optimized for storing massive amounts of ontology individuals at the price of limited expressivenes and reasoning functionality. PeerThing [15] is another P2P based semantic metadata repository focusing on hardware and software annotation in the Grid based on OWL and RACER reasoner.



WSDL URI http://www.xmethods.net/sd/2001/BNQuoteSe

getPrice	
IN isbn	
OUT return	

Ontology URI http://gom.kwfgrid.net/gom/ontology/DomainA

Reservoir
DaveF2DVisualizationProperties
DaveFProperties
GeographicalLocation
Stream
Settlement
MM5WeatherPrediction
Hydrograph
MM5WatershedIntegrationProperties
Street
Nature
Mountain
State
HydraulicsProperties
MM5OnewayNestedProperties
uppermostLatitude
PointOfInterest

(a) Service selection. (b) Annotation of concepts from WSDL.

Fig. 5: Example WSRF2OWL-S screen shots.

4 Conclusions and Future Work

The main achievements of this work include the definition of unified semantic metadata model of the Grid called ontology separation scheme, design and development of generic distributed knowledge base with an event system enabling updating of managed ontologies and recovery of the state of the knowledge base from any given time in the past from serialized history of incoming events and development of Protege plug-in that enable easy interaction with the knowledge base.

The main scientific innovations include the architecture of flexible knowledge base framework, that can be optimally configured for a given kind of ontology, development of WSRF2OWL-S tool which enhances current state of the art by handling stateful Grid services and prototype implementation of CIM2OWL translation tool and development of special protocol for interacting with OWL ontologies, which allows the knowledge base to return a OWL document with a consise subset of the ontolog that contains a semantic neighborhood of results of a query.

Future work will include evaluation of different distribution models such as P2P, improvement of performance of the system as well as integration of ontology similarity algorithm for querying and updating the ontologies [16, 17, 18].

Acknowledgements. This reasearch has been done in the framework of EU IST-2004-511385 K-Wf Grid project. The authors would like to thank the whole project consortium for remarks and feedback during the development process. AGH University of Science and Technology grant is also acknowledged.

References

1. Gom website – `http://gom.kwfgrid.net`.
2. K-wfgrid project – `http://www.kwfgrid.eu`.

3. Dutka, L., Kryza, B., Krawczyk, K., Majewska, M., Slota, R., Hluchy, L., and Kitowski, J., Component-expert architecture for supporting grid workflow construction based on knowledge, In *Cunningham P. Cunningham M. (Eds): Innovation and the Knowledge Economy. Issues, Applications, Case Studies. pp. 239-246, Vol. 2, IOS Press 2005.*
4. Babik, M., Habala, O., Hluchy, L., Laclavik, M., and Maliska, M., Semantic grid services in k-wf grid, In *To appear in Proc. of the 2nd International Conference on Semantics, Knowledge and Grid (SKG2006), IEEE Computer Society, Guilin, China, 2006.*
5. Krawczyk, K., Slota, R., Majewska, M., Kryza, B., and Kitowski, J., Grid organization memory for knowledge management for grid environment, In *Bubak, M., Turala, M., Wiatr, K. (eds): Proceedings of the Cracow Grid Workshop '04, pp. 109-115, ACC CYFRONET AGH, Krakow, Poland, 2005.*
6. Kryza, B., Majewska, M., Slota, R., and Kitowski, J., Unifying grid metadata representations through ontologies, In *R. Wyrzykowski et.al. (eds) Proc. of 6th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005, Poznan, Poland; LNCS 3911, Springer-Verlag, pp.683-690, 2006.*
7. Kryza, B., Pieczykolan, J., Babik, M., Majewska, M., Slota, R., Hluchy, L., and Kitowski, J., Managing semantic metadata in k-wf grid with grid organizational memory, In *Bubak, M., Turala, M., Wiatr, K. (eds): Proceedings of the Cracow Grid Workshop '05, pp. 66-73, ACC CYFRONET AGH, Krakow, Poland, 2006.*
8. Kryza, B., Pieczykolan, J., and Kitowski, J., Grid organizational memory: A versatile solution for ontology management in the grid, In *Proceedings of 2nd International Conference on e-Science and Grid Computing, Dec. 4-6, 2006, Amsterdam, Netherlands, (C) IEEE Computer Society Press.*
9. Babik, M., Hluchy, L., Kitowski, J., and Kryza, B., Wsrf2owl-s: A framework for generating semantic descriptions of web and grid services, In *Bubak, M., Turala, M., Wiatr, K. (eds): Proc. of the Cracow Grid Workshop '05, pp. 49-46, ACC CYFRONET AGH, Krakow, Poland, 2006.*
10. Babik, M., Hluchy, L., Kitowski, J., and Kryza, B., Generating semantic descriptions of web and grid services, In *To appear in Proc. of the 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems, Springer, Innsbruck, Austria, 2006.*
11. Tupelo 2 website, `http://dlt.ncsa.uiuc.edu/wiki/index.php/tupelo_2`.
12. Serafini, L. and Tamilin, A., Drago: Distributed reasoning architecture for the semantic web., In Gómez-Pérez, A. and Euzenat, J. (Eds), *ESWC*, *Lecture Notes in Computer Science*, vol. 3532, pp. 361–376, Springer, 2005.
13. Miles, S., Papay, J., Payne, T.R., Decker, K., and Moreau, L., Towards a protocol for the attachment of semantic descriptions to grid services., In Dikaiakos, M. D. (Ed.), *European Across Grids Conference*, *Lecture Notes in Computer Science*, vol. 3165, pp. 230–239, Springer, 2004.
14. Horrocks, I., Li, L., Turi, D., and Bechhofer, S., The instance store: Dl reasoning with large numbers of individuals., In Haarslev, V. and Möller, R. (Eds), *Description Logics*, *CEUR Workshop Proceedings*, vol. 104, CEUR-WS.org, 2004.
15. Heine, F., Hovestadt, M., and Kao, O., Towards ontology-driven p2p grid resource discovery., In Buyya, R. (Ed.), *GRID*, pp. 76–83, IEEE Computer Society, 2004.
16. Pieczykolan, J., Kryza, B., and Kitowski, J. Semi-automatic creation of adapters for legacy application migration, In *Proceedings of International Conference on Computational Science 2006, pp. 252-259, vol. 4, Springer, The University of Reading, UK, Springer, 2006.*

17. Kryza, B., Majewska, M., Pieczykolan, J., Slota, R., and Kitowski, J., Grid organizational memory – provision of a high-level grid abstraction layer supported by ontology alignment, *The International Journal of FGCS, Grid Computing: Theory, methods & Applications, Elsevier*, 23(3):348–358, March 2007.
18. Slota, R., Zieba, J., Kryza, B., and Kitowski, J., Knowledge evolution supporting automatic workflow composition, In *Proceedings of 2nd International Conference on e-Science and Grid Computing, Dec. 4-6, 2006, Amsterdam, Netherlands, (C) IEEE Computer Society Press.*

On Translation Common Information Model to OWL Ontology

Marta Majewska[1], Bartosz Kryza[2], and Jacek Kitowski[1,2]

[1] AGH-UST, al. Mickiewicza 30, 30-059 Krakow, Poland
[2] Academic Computer Centre Cyfronet AGH, Nawojki 11, 30-950 Krakow, Poland
emails: {mmajew, bkryza, kito}@agh.edu.pl

Abstract
This paper presents our work on definition and development of Grid resource ontology based on Common Information Model. The paper presents issues of mapping the CIM model to Web Ontology Language (OWL) standard. The paper contains also comparison of existing approaches to conversion of CIM to OWL. A tool is also mentioned that performs the conversion of CIM schema and allows conversion of CIM instances to OWL individuals.

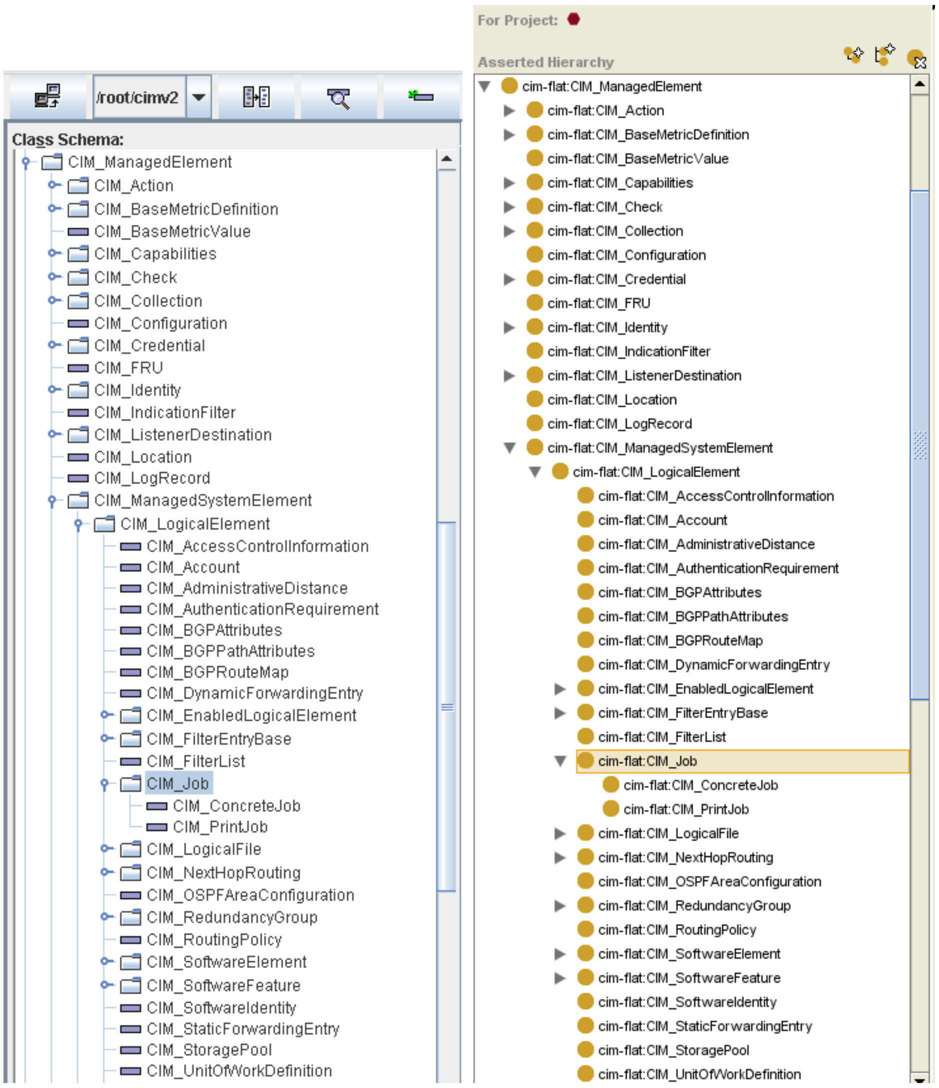
1 Introduction

The need of unified semantic description of the various Grid aspects appeared during the work on the knowledge based system for composing workflows for the grid environment in the EU IST K-Wf Grid project [1], where OWL ontologies were chosen for metadata descriptions of the Grid. The set of ontologies describing generic and domain specific features of the grid were gathered and integrated for thematic areas of workflows, grid applications, services, data and resources. However, the overall ontology for grid resource description was still an open issue. At the field of modeling computer software and hardware resources already exists a recognized standard – DMTF Common Information Model (CIM) [2]. As OWL [3] was a representation choice for the project ontologies, the problem of the interoperability appeared. For providing ontology interoperability a mapping from MOF [4] (CIM native representation language) to OWL was considered, having in mind that the full reconciliation of the two formalisms originating from the various backgrounds is challenging. Both CIM and OWL originate from different research and development communities. CIM was designed and is progressively developed by DMTF mainly for the purpose of hardware and software resources description in the distributed environments. The unified CIM information model was often used in computer network resource management systems and large scale monitoring applications. CIM received substantial popularity at this field and is considered as a standard. CIM is a hybrid approach, inspired by the object oriented modeling and database information modeling. As it introduces the meta-data for annotating model classes and individuals, it is partially not compliant with the UML methodology. MOF is the IDL platform independent language for the CIM model representation. The CIM Schema consists of particular Core, Common Models and developed by users Extension

Schemas. Shortly speaking, OWL is W3C recommended ontology language for the Semantic Web. It exploits many of the strengths of Description Logics, including well defined semantics and practical reasoning techniques. OWL offers greater expressiveness of information content description then that supported by XML, RDF, and RDF Schema, by providing additional vocabulary along with a formal semantics. OWL has three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full. However, the reconciliation of two formalisms originating from the various backgrounds is challenging. Some features of CIM originating from object oriented or database designing are difficult for translation to OWL. Likewise, some OWL characteristics, coming from the Description Logic background do not found their equivalents in CIM or cause some mapping attempts being hard.

2 Related Work

In this section, research papers and projects in the field of CIM ontology alignment with Semantic Web ontology standards are discussed. The authors of [5] propose to use XML-based ontology language to define resource management information for network management systems. They study advantages, which such an ontology can provide to this area and postulate integration of the existing management information specifications in MOF, GDMO, SMI formats using management facets written in OWL for ontology-based management systems. The mapping proposition for describing data restriction and cardinality, documentation, versioning, object distinction, model reference, element redefinition and access is described. The paper presents interesting approach, however the translation software – announced as a plug-in for loading CIM Schemas represented in MOF to Protege – is not available. It is survey of mapping possibilities for various resource meta-description formats. In [6] is presented the idea of developing the semantic descriptions of the web services based on the CIM specifications. The example of the OWL-S [7] service definition is constructed using the CIM XML format specification for service data and methods. The CIM semantic descriptions are transformed to OWL. The paper is not focused on CIM mapping issues. The CIM to OWL translation is done for basic constructs and in the limited scope, intending to be rather a proof of concepts. In [8] the mapping of CIM UML notation to RDF/S, and the extension for OWL, are presented. The paper contains first the mapping for the basic syntax constructs to RDF/S and then a supplement for OWL. While the mapping for RDF is elaborated in detail, the OWL part is rather a set of mapping propositions, not formalized or comprehensive. (e.g. the idea of using owl:allValuesFrom and owl:inverseOf for associations, owl:TransitiveProperty for inheritance in not clear). It is important and stressed in the paper that the lack of the entire semantic correspondence between CIM, UML and OWL obstruct full and unambiguous mapping. The loss of the expressivity of mapping is caused by two factors, i.e. using the UML representation of CIM and UML to OWL translation process. The presented above in our paper mapping from MOF to OWL allows protecting more CIM


/root/cimv2
Class Schema:
CIM_ManagedElement
CIM_Action
CIM_BaseMetricDefinition
CIM_BaseMetricValue
CIM_Capabilities
CIM_Check
CIM_Collection
CIM_Configuration
CIM_Credential
CIM_FRU
CIM_Identity
CIM_IndicationFilter
CIM_ListenerDestination
CIM_Location
CIM_LogRecord
CIM_ManagedSystemElement
CIM_LogicalElement
CIM_AccessControlInformation
CIM_Account
CIM_AdministrativeDistance
CIM_AuthenticationRequirement
CIM_BGPAttributes
CIM_BGPPathAttributes
CIM_BGPRouteMap
CIM_DynamicForwardingEntry
CIM_EnabledLogicalElement
CIM_FilterEntryBase
CIM_FilterList
CIM_Job
CIM_ConcreteJob
CIM_PrintJob
CIM_LogicalFile
CIM_NextHopRouting
CIM_OSPFAreaConfiguration
CIM_RedundancyGroup
CIM_RoutingPolicy
CIM_SoftwareElement
CIM_SoftwareFeature
CIM_SoftwareIdentity
CIM_StaticForwardingEntry
CIM_StoragePool
CIM_UnitOfWorkDefinition
For Project:
Asserted Hierarchy
cim-flat:CIM_ManagedElement
cim-flat:CIM_Action
cim-flat:CIM_BaseMetricDefinition
cim-flat:CIM_BaseMetricValue
cim-flat:CIM_Capabilities
cim-flat:CIM_Check
cim-flat:CIM_Collection
cim-flat:CIM_Configuration
cim-flat:CIM_Credential
cim-flat:CIM_FRU
cim-flat:CIM_Identity
cim-flat:CIM_IndicationFilter
cim-flat:CIM_ListenerDestination
cim-flat:CIM_Location
cim-flat:CIM_LogRecord
cim-flat:CIM_ManagedSystemElement
cim-flat:CIM_LogicalElement
cim-flat:CIM_AccessControlInformation
cim-flat:CIM_Account
cim-flat:CIM_AdministrativeDistance
cim-flat:CIM_AuthenticationRequirement
cim-flat:CIM_BGPAttributes
cim-flat:CIM_BGPPathAttributes
cim-flat:CIM_BGPRouteMap
cim-flat:CIM_DynamicForwardingEntry
cim-flat:CIM_EnabledLogicalElement
cim-flat:CIM_FilterEntryBase
cim-flat:CIM_FilterList
cim-flat:CIM_Job
cim-flat:CIM_ConcreteJob
cim-flat:CIM_PrintJob
cim-flat:CIM_LogicalFile
cim-flat:CIM_NextHopRouting
cim-flat:CIM_OSPFAreaConfiguration
cim-flat:CIM_RedundancyGroup
cim-flat:CIM_RoutingPolicy
cim-flat:CIM_SoftwareElement
cim-flat:CIM_SoftwareFeature
cim-flat:CIM_SoftwareIdentity
cim-flat:CIM_StaticForwardingEntry
cim-flat:CIM_StoragePool
cim-flat:CIM_UnitOfWorkDefinition

(a) CIMOM browser (b) Protege OWL browser

Fig. 1: ManagedElement hierarchy.

model original semantics. The author of the [9] focuses on the troublesome CIM to OWL translation issues. As a reason for difficulties of a translation the author postulates among others the diverse character of the namespace or a lack of the data container representation in the OWL. The author considers possible solutions as only partly correct ones (e.g. for vector reprentation using rdf:List constructs together with owl:allValuesFrom for element type restriction or alternatively creation of multi-valued properties with the owl:maxCardinality restriction, but without the control of the element sequence). In other place the author rightly observes, the only possibility to express the default values meaning in OWL seems to be annotating properties and for methods breakneck becomes thinking of the name's overloading and method inheritance rules. The author concludes pessimistically that OWL, in spite of being developed for the semantic integration purpose, finds limited applicability outside the semantic community. There is no information about the implementation of concrete tools for translation and their availability. The currently available translation tool is CIMTool [10], however with a poor documentation. It is dedicated to support management and integration of semantic models of various power management systems as well as the communication based on the semantic descriptions in CIM/XML and OWL formats. The solution works on CIM/XML language for representing power system models, which works with a subset of the RDF syntax and an RDF schema, adapted for the EPRI CIM (the EPRI CIM in UML form is maintained by IEC TC 57). The subset of RDF Syntax is designed to simplify serializers and interpreters while retaining compatibility with existing RDF tools. By the way, the CIM RDF schema is derived from the CIM Rose Model. The limited scope of constructions used in the translation is the drawback of that solution in our view. The CIMTool provides possibility to read and merge CIM and local UML models in the XMI form, browse models and check inconsistencies, generate equivalent OWL ontologies, create and edit message definitions based on the ontology and use those to generate XML schemas for messages. The common integrated CIM model allows communication between systems of various backgrounds, e.g. by generating the messages according diverse formats.

3 Mapping Approach

Since detailed discussion of the full mapping from CIM to OWL is beyond the scope of this paper, we present here an example of the mapping covering some aspects of the problem. In Figure 1a we can see a subset (hierarchy of CIM_ManagedElement concept) of the CIM Schema in CIMOM browser and in Figure 1b we can see the resulting CIM OWL ontology in the Protege ontology editor.

As a first example, lets consider simple mapping of part of `CIM_Job` class to OWL as shown in Table 1. To comment this example let us mention the mapping of a qualifier Description and data types. The Description qualifier is dedicated for the documentation, conveying semantic information to the user in the natural language form. The best suitable mapping for this qualifier is rdfs:comment. Then, between datatypes in MOF and OWL the equivalents could be find nearly for all types. The exception is the datatime MOF type, because both languages have different internal representation of date and time. In this case, the transformation of the format has to be done by the mapping

```
// CIM_Job
[Description ("A Job is a LogicalElement that ...")]
class CIM_Job : CIM_LogicalElement {
  [Description ("A free-form string that ...")]
  string JobStatus;
  //...
};
```

```
<!-- CIM_Job -->
<owl:Class rdf:about="#CIM_Job">
  <rdfs:subClassOf rdf:resource="#CIM_LogicalElement"/>
  <rdfs:comment>A Job is a LogicalElement ...</rdf:comment>
</owl:Class>
<owl:DatatypeProperty rdf:ID="CIM_Job__JobStatus">
  <rdfs:domain rdf:resource="#CIM_Job" />
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:comment>A free-form string that ...</rdf:comment>
</owl:DatatypeProperty>
```

Tab. 1: An example of a simple class definition.

tool. This rule applies also some other mapping cases discussed later.

Now lets consider in more detail translation of `CIM_Job` concept. In Table 2 we see the original CIM definition (top-left corner) and the resulting OWL ontology definition (bottom). In the top-right corner we see a subset of CIM meta OWL ontology. The usage of the cim-meta ontology allows mapping of the MOF constructs, which are difficult to express with the help of means offered by the OWL language in precise way. Elements modeled in the meta ontology are referred in the target ontology using e.g. annotations (e.g. default value, qualifiers Version, Abstract) or inheritance (e.g. `CIM_Association`, `CIM_Association__Ref`). The benefits are important, the usage of meta ontology allows elements categorization or reasoning. The lack of the fully equivalent constructions or rules for expressing some MOF constructs in OWL causes that the partial, simplified mappings have to be admitted. These unambiguous mappings concern mainly data restriction (e.g. qualifier Value, ValueMaps), distinction (e.g. qualifier Key, Propagated, Weak), redefinition (e.g. qualifier Override), access (e.g. qualifier Read, Write), versioning (qualifier Version), default values, abstracting (qualifier Abstract), and dynamics (e.g. procedures, qualifier IN, OUT). Unfortunately, many issues of the not fully semantically equivalent part schema mapping are out of the scope of this paper. To achieve optimal mappings a few approaches were used:

- the usage of various semantic constructs with an approximate and uncontradictory meaning (e.g. for qualifier Override),
- the development of the additional meta ontology for a definition of missing CIM vocabulary called cim-meta (e.g. for qualifiers Abstract, ValueMap, Values, default values),
- the usage of comments and annotation properties (e.g. for qualifier Units).

The Table 2 shows also the mapping of some popular MOF constructs with the help of mentioned approaches. These are qualifiers Abstract, Version, Write, default value, qualifiers ValueMap, Values, Units.

```
[Abstract, Version ( "2.10.0" )]
class CIM_Job : CIM_LogicalElement {
  [Write]
  uint32 JobRunTimes = 1;
  [ValueMap { "1", "2" },
  Values { "Local Time", "UTC Time" }]
  uint16 LocalOrUtcTime;
  [Units ( "Percent" )]
  uint16 PercentComplete;
//
};
```

```
// cim-meta ontology
<owl:Class rdf:ID="CIM_Value"/>
<owl:DatatypeProperty rdf:ID="value">
  <rdfs:domain rdf:resource="#CIM_Value"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="valueMap">
  <rdfs:domain rdf:resource="#CIM_Value"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:AnnotationProperty rdf:ID="defaultValue"/>
<owl:AnnotationProperty rdf:ID="cimVersion"/>
<owl:AnnotationProperty rdf:ID="isAbstract"/>
```

```
<owl:Class rdf:ID="CIM_Job">
  <cim-meta:cimVersion>2.10.0</cim-meta:cimVersion>
  <cim-meta:isAbstract>true</cim-meta:isAbstract>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <cim-meta:CIM_Value rdf:ID="CIM_Job__LocalOrUtcTime_LocalTime ">
               <cim-meta:value>Local Time</cim-meta:value>
               <cim-meta:valueMap>1</cim-meta:valueMap>
            </cim-meta:CIM_Value>
            <cim-meta:CIM_Value rdf:ID="CIM_Job__LocalOrUtcTime_UCTTime ">
               <cim-meta:value>UCT Time</cim-meta:value>
               <cim-meta:valueMap>2</cim-meta:valueMap>
            </cim-meta:CIM_Value>
          </owl:oneOf>
        </owl:Class>
      </owl:allValuesFrom>
      <owl:onProperty>
          <owl:ObjectProperty rdf:ID="CIM_Job__LocalOrUtcTime"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="CIM_Job__JobRunTimes">
  <rdfs:domain rdf:resource="#CIM_Job" />
  <rdfs:range rdf:resource="&xsd;unsignedShort"/>
  <cim-meta:Writeable/>
  <cim-meta:defaultValue>1</cim-meta:defaultValue>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="CIM_Job__LocalOrUtcTime">
  <rdfs:domain rdf:resource="#CIM_Job" />
  <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="CIM_Job__ PercentComplete">
  <rdfs:domain rdf:resource="#CIM_Job" />
  <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
  <rdfs:comment>Units ( "Percent" )</rdfs:comment>
</owl:DatatypeProperty>
```

Tab. 2: An example of a simple class definition.

Our CIM2OWL tool allows conversion of Common Information Model schema and individual to OWL. The CIM2OWL tool converts CIM schema, i.e. generate OWL ontologies from CIM schema and convert CIM instances, i.e. convert proprietary CIM instances to OWL individuals. The conversion is based on the WBEM Services API [11] and Jena Semantic Framework [12] and uses their internal model representations and verification capabilities.

4 Conclusions and Future Work

As a result they were created: the OWL resource ontology, the mapping schema and the CIM2OWL tool, which advantage is the ability to translate not only CIM schema (i.e. CIM Core and Common Models) but also CIM instances (i.e. CIM Extensions). The large scope of the expressiveness and functionality was transformed from CIM to OWL. It was possible to achieve good functional value ontology with minor expressiveness loss. The ontology strength is based on the recognized and constantly developed standard. The further changes, like minor modification or extensions in the mapping, are relatively easy to apply. The ontology and the CIM2OWL tool can be found on the Grid Organizational Memory [13, 14, 15] website: [16]. Current version of the ontology `http://gom.kwfgrid.net/ontology/cim/2006/10/02/Flat` contains 651 concepts, 3626 properties and 4365 individuals.

The future work will include further evaluation of the resulting ontologies, resolving of remaining mapping issues and further development of the CIM2OWL tool.

Acknowledgements. This reasearch has been done in the framework of EU IST-2004-511385 K-Wf Grid project. The authors would like to thank the whole project consortium for remarks and feedback during the development process. AGH University of Science and Technology grant is also acknowledged.

References

1. K-Wf Grid project website, `http://www.kwfgrid.net/`
2. DMTF, Common Information Model (CIM) Standards, `http://www.dmtf.org/standards/cim/`
3. W3C, Web Ontology Language (OWL), `http://www.w3.org/2004/OWL/`
4. DMTF, Managed Object Format (MOF), `http://www.dmtf.org/education/mof/`
5. Lopez de Vergare, J.E., Villagra, V.A., Asensio, J.I., Berrocal, J.; Application of the Web Ontology Language to define management information specifications. Proc. of the HP Openview University Association Workshop, France, 2004.
6. Keeney, J., Carey, K., Lewis, D., O'Sullivan, D., Wade, V.; Ontology-based Semantics for Composable Autonomic Elements. Workshop of AI in Autonomic Communications at the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, UK, 2005.
7. OWL-S website, `http://www.daml.org/services/owl-s/1.0/`
8. Quirolgico, S., Assis, P., Westerinen, A., Baskey, M., Stokes, E.; Toward a Formal Common Information Model Ontology. Proc. of Workshop on Intelligent Net-

worked and Mobile Systems, WISE04, Australia, 2004, Springer, LNCS vol.3307, pp. 11-21.

9. Heimbigner, D.; DMTFCIM to OWL: A Case Study in Ontology Conversion. Proc. of the Conference on Software Engineering and Knowledge Engineering (SEKE2004), Canada, 2004, ISBN 1-891706-14-4, pp. 470-474.
10. CIMTool project website, `http://cimtool.org/`
11. WBEM Services project webpage, `http://wbemservices.sourceforge.net/`
12. Jena Semantic Framework website, `http://jena.sourceforge.net/`
13. Kryza, B., Pieczykolan, J., Babik, M., Majewska, M., Slota, R., Hluchy, L. and Kitowski, J.; Managing Semantic Metadata in K-Wf Grid with Grid Organizational Memory. In: Bubak, M., Turala, M., Wiatr, K. (editors), Proceedings of the 5th Cracow Grid Workshop (CGW05), Krakow, Poland, ACC-CYFRONET UST, pp. 66-73.
14. Kryza, B., Slota, R., Majewska, M., Pieczykolan, J., Kitowski, J.; Grid Organizational Memory – Provision of a High-Level Grid Abstraction Layer Supported by Ontology Alignment. Future Generation Computer Systems Volume 23, Issue 3, March 2007, Pages 348-358, Elsevier.
15. Kryza B., Pieczykolan J., Kitowski J.; Grid Organizational Memory: A Versatile Solution for Ontology Management in the Grid. In Proc. of 2nd International Conference on e-Science and Grid Computing, Dec. 4-6, 2006, Amsterdam, Netherlands, (C) IEEE Computer Society Press. ISBN 0-7695-2734-5
16. Grid Organizational Memory website: `http://gom.kwfgrid.net`

Peer-to-Peer Distribution Model for Grid Organizational Memory Knowledge Base

Bartosz Kryza[1], Jan Pieczykolan[1], Jakub Wach[2], Mikołaj Zuzek[2], and Jacek Kitowski[1,2]

[1] Academic Computer Centre CYFRONET AGH, Krakow, Poland
[2] Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland

Abstract

In this paper an architecture of distributing an ontological Grid knowledge base over peer to peer network is presented. The discussion is based on Grid Organizational Memory knowledge base developed within the framework of K-Wf Grid EU-IST project. Authors discuss several existing approaches to P2P such as structured and unstructured and draw conclusions on which of these would be most applicable for the problem of managing heterogenous and distributed resources in a Grid environment.

1 Introduction

Latest trends in application of knowledge bases in distributed settings are being oriented toward scalable and fault-tolerant solutions based on Peer-to-Peer communication model. Currently, several semantic knowledge base solutions which are based on the mentioned model exist. However they are either very generic, with low performance and very basic functionality or are application specific oriented, making them awkward to use as semantic metadata management solutions for the Grid.

Grid Organizational Memory is a distributed knowledge base, developed within the framework of EU-IST K-Wf Grid project [1], for managing semantic metadata of Grid resources, such as hardware, data and software, as well as domain specific application information, using OWL-DL language as an ontology formalism. It is based on a notion of ontology separation scheme [2], where a global ontology is divided into interrelated ontology components, each managed by separate and possibly distributed GOM Engine element. A single ontology component is bound to a unique URI, and can be rendered and published as an OWL document. Each GOM Engine [3] can be configured in different way, providing various capabilities for ontology storage and reasoning. Currently, GOM contains rather simple distribution model, assuming that each ontology component is treated as one whole and should be managed by a single GOM Engine element. In order to extend its functionality for very large metadata sets, authors propose a P2P model for distributing an ontology component into a network of peers that will together provide a scalable semantic metadata store for some part of the metadata, for example data registry.

Due to the nature of the Grid, the authors discuss that the unstructured model of a P2P network is more natural for such environment, since it keeps metadata close to the described resources. Also, the ontology separation scheme imposes grouping of peers by super-peer nodes, which manage the particular ontology component. This model assures natural clustering of metadata into consistent sets of ontological individuals, since metadata is stored 'locally' with respect to annotated resources. Unstructured P2P network model imposes less burden on administrators and users in terms of depoying new instances of ontology components. However, use of the mentioned model have some serious drawbacks, the most important is a high cost of query answering due to the need of flooding the network in order to retrieve all potential answers. This paper discusses possible solutions for that problem including distributed indexes such as DHT and subscription mechanism which should substantially decrease the network traffic necessary for answering most query patterns. Another issue discussed in the paper is the semantic reasoning over distributed ontologies. Although the assumption is that local models will be consistent, thus some reasoning over each local model will be possible, not always all possible entailments will be inferred in this way. The applicability of several possible approaches to this problem is analysed. Since the proposed P2P model must also support the internal Grid Organizational Memory event system, which is used to update the semantic metadata on fine grained level (e.g. values of relations), a section is provided that shows how to extend that event system for a P2P network of nodes, assuring consistency of information.

2 Peer to Peer Knowledge Management in the Grid

Peer to peer networks have been given much attention in the research communities in the past years. This resulted in several models and theoretical results, some of them could be particularly appropriate to the problem of metadata distribution and resource management in distributed environments. In [4] authors examine using of the super-peer model in a multi-organizational Grid. The model is applied to membership management and resource discovery. What's more, a simulation analysis evaluates the performance of proposed model in discovering resources. From P2P-researcher point of view, paper is interesting because of the novelty – super-peer model, which aims at deploying a P2P information service in Grids. hierarchical P2P systems are discussed in [5]. Authors closely consider the specific case of two-tier hierarchy, in which peers are organized into groups with autonomous intra-group overlay and lookup. Groups are organized in a top-level tier with own overlay network. They also evaluate proposed model using Chord for top-level tier and shows that it could significantly reduce the expected number of hops in Chord, boosting system's performance. In [6] authors concentrate on using the de Bruijn routing in DHTs. Specifically, they prove that it fails to meet conflicting objectives of storage load balancing as well as search efficiency, when applied to systems with uneven key distribution. In [7], authors describes adaptive and bandwidth-efficient solution of the problem of sharing structured data in an unstructured P2P system. Their method allows high quality answers to be obtained by using efficient query routing and clustering of peers. Results of simulations are also presented, which show that presented solution is very good performer in terms of bandwidth-efficiency and effectiveness in a variety of workloads and networks. Authors of [8] explore various alternatives to standard flooding query algorithm, as found in Gnutella system. As a result an algorithm is proposed, that is based on multiple random walks. Author shows, that while being as quick in resolving queries as Gnutella's algorithm, in many cases it reduces the network traffic by two orders of magnitude. What's more, Authors also find that uniform random graphs is the best network topology performance-wise. In [9] authors perform a very detailed comparison study on structured and unstructured P2P overlay networks. In contrary to prevalent opinions Authors prove, that structured overlays could achieve low maintenance overhead and exploit heterogeneity effectively. Further research is carried on using techniques from unstructured networks, as floods and random walks in structured overlays. Exploiting structural constrains, these techniques allows to support complex queries with even better performance than unstructured overlays.

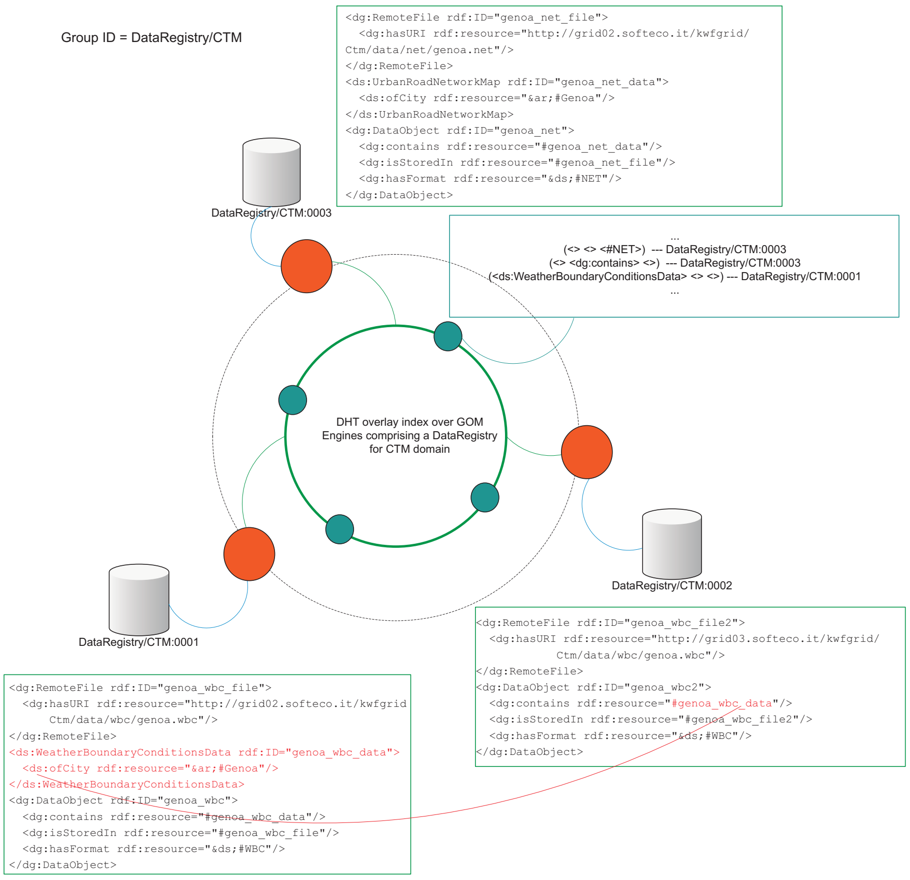
Apart general P2P research some investigations and attempts to develop P2P based knowledge bases has been performed. In [10] semantic data retrieval from knowledge base distributed over peer-to-peer network is considered. After study of example scenario from tourism domain set of requirements is defined and peer architecture is proposed. Semantic Web and P2P (SWAP) metadata model is presented in [11]. Since no shared understanding of the domain can be assumed authors propose methods for building views over the knowledge repository of a peer and rating their content. SeRQL language is described and used to define views. It allows referencing the results with views serialized in the queries. In the end issues connected with view visualisation are discussed. Authors of [12] present scalable grid ontology repository system named OntStore that provides self-organizing and content addressable network. Efficiency of distributed resource description storing and quering is improved by distributed hash table (DHT) based on Pastry architecture. In [13] ontology-based approach to the data interoperability problem in a heterogeneous P2P network is presented. Giving illustrative examples authors discuss the issue of query processing and propose query rewriting algorithm which takes into account integrity constraints of local data. Simple mapping language PML is defined and used to specify ontology mappings. Another scalable peer-to-peer RDF repository named RDFPeers is described in [14] authors describe. Peers organize themselves into multiattribute addressable network (MAAN) that is built over Chord overlay. Using MAAN as the underlying network layer RDFPeers extends it with RDF-specific storage, retrieval, subscription and load-balancing techniques. Presented framework allows multi-attribute and range RDQL queries to be routed efficiently to nodes that store matching triples, altough not all query types are supported. Hierarchical Knowledge Editor named Shared-HiKE is presented as an example of application for which the distributed RDF repository is well suited for. Theoretical approach to distributed knowledge bases can be found in [15]. Authors precisely define general framework for P2P network systems, general querying algorithm adequate for incremental search and related concepts. Algebraic models and modal logic are exploited to derive theorems which make theoretical foundations of P2P systems design. Getting certain answers from peers with partial information in knowledge systems with integrity constraints is the problem to which the solutions are proposed. In [16] authors present unstructured peer-to-peer architecture called KEEx which store knowledge in document repositories and describes resources in contexts maintained by context repositories. Peers are organized in groups called K-federations to reflect their content relations and social structures. Query resolution involves lexical search and semantic resolution based on context matching algorithm. Detailed description of peer-to-peer platform called IMAGINE (Integrated Multidisciplinary Autonomous Global Innovation Networking Environment) is presented in [17]. Above object layer managed by Chord DHT overlay semantic layer is built to support complex queries such as wildcard queries and range queries on key strings. Tree-like structured index is used to achieve platform scalability and provide efficient query processing. To increase services quality plaftorm design includes sophisticated features like decentralized load-balancing and semantic overlay replication. Authors of [18] propose peer-to-peer architecture that supports ontology-driven resource discovery in distributed knowledge base. Detailed description of algorithm for distributed resource matching is given. System utilizes RACER, an interface to OWL language, as its deductive component.

3 Vision of the Proposed Solution

The first major assumption is to keep the peer-to-peer overlay unstructured, which means the resource descriptions should be kept close to the actual resources. The user should also have the possibility to freely choose the GOM Engine to which resource description is submited. This will allow descriptions to be stored locally to the publisher in arbitrary sense (e.g. physical proximity, organizational proximity). Descriptions should be available to all users, possibly according to some authorization constraints. This assumption should enforce clustering of the triples in a way that will enable reasoning in local engines. While specifying the query user should be able to favor local matches. Many definitions of locality can be introduced. Some of them may be based on network radius (e.g. in terms of network hops) while others may utilize semantical similarity of explored resources. To constrain network flooding and improve searching performance queries should be optimized by maintaining a DHT index of resources stored in the network. Well designed indexing should be able to significantly decrease the number of peers targeted by the queryies

The system will be oriented on distributing the instance registries – not the domain ontologies (schemas), because the registries in Grid applications tend to grow much faster then schemas of the ontologies. Peers will be organized in groups, where each group will be bound to a domain ontology and identified by it's name (e.g. DataRegistry for Coordinated Traffic Management VO). Peers in a group together provide the required functionality which is answering queries, modification of resources and reasoning.

With respect to GOM as a distributed knowledge base, the most important use case is when an actor adds a resource description, written in OWL language. Architecture of GOM P2P overlay assumes that this submission is complete. It means that a description of a resource, which is a set of named, and possibly anonymous OWL individuals stored in one GOM Engine instance (P2P node) is clustered. This constraint enables GOM P2P to merge results of queries, performed on each node, in simple and elegant way. Modified scenario for this use case enables an actor to reuse existing named OWL individuals from other registries. This could be applicable in cooperation with replication of the data, e.g. when a replica of a file has been created in the local file system, the metadata instance about this file could also be replicated. Of course this is considered as an option in GOM-P2P, because synchronization issues might be very difficult. The next basic functionality of GOM-P2P knowledge base is removing of individuals. This scenario assumes that information being removed is a set of individuals that comprise some resource description. Modification of already stored metadata will also be supported by our system. This can include modification of data or object property of an OWL individual, which is a part of some resource's description. In all described above use cases an GOM Engine is modified. It changes the knowledge it contains, so new reasoning should be performed on it before answering a query. This will assure that the answer will be consistent. Firstly query is executed on a local GOM Engine. After obtaining local results, selected node propagates the query to other peers in order to find the RDF triples that match this query. Most probably propagation will be optimized by using the DHT index. The resulting information – RDF triples that match specified query – are merged by the node that initiated whole process. After merging, complete information is returned to the client that queried GOM-P2P. In GOM P2P, resolution for those questions combines with it's role as a knowledge base. In particular, when a GOM Engine is being deployed, it will connect to the existing group of nodes comprising of particular registry, for example data registry of CTM domain. In the second case, when GOM Engine in being stopped, it will try to notify it's peer group that it will not be able to provide further service. The first problem is dealing with the indexes in the DHT, that refer to the node being removed. Obvious solution is to remove them in some way. The second issue is what to do, if other nodes refer to some individuals, that are described in the node being detached. Those are questions, that remain to be answered after further research.


Group ID = DataRegistry/CTM
DataRegistry/CTM:0003
DHT overlay index over GOM Engines comprising a DataRegistry for CTM domain
DataRegistry/CTM:0002
DataRegistry/CTM:0001

```
<dg:RemoteFile rdf:ID="genoa_net_file">
  <dg:hasURI rdf:resource="http://grid02.softeco.it/kwfgrid/
Ctm/data/net/genoa.net"/>
</dg:RemoteFile>
<ds:UrbanRoadNetworkMap rdf:ID="genoa_net_data">
  <ds:ofCity rdf:resource="&ar;#Genoa"/>
</ds:UrbanRoadNetworkMap>
<dg:DataObject rdf:ID="genoa_net">
  <dg:contains rdf:resource="#genoa_net_data"/>
  <dg:isStoredIn rdf:resource="#genoa_net_file"/>
  <dg:hasFormat rdf:resource="&ds;#NET"/>
</dg:DataObject>
```

```
...
(<> <> <#NET>)  --- DataRegistry/CTM:0003
(<> <dg:contains> <>)  --- DataRegistry/CTM:0003
(<ds:WeatherBoundaryConditionsData> <> <>) --- DataRegistry/CTM:0001
...
```

```
<dg:RemoteFile rdf:ID="genoa_wbc_file2">
  <dg:hasURI rdf:resource="http://grid03.softeco.it/kwfgrid/
          Ctm/data/wbc/genoa.wbc"/>
</dg:RemoteFile>
<dg:DataObject rdf:ID="genoa_wbc2">
  <dg:contains rdf:resource="#genoa_wbc_data"/>
  <dg:isStoredIn rdf:resource="#genoa_wbc_file2"/>
  <dg:hasFormat rdf:resource="&ds;#WBC"/>
</dg:DataObject>
```

```
<dg:RemoteFile rdf:ID="genoa_wbc_file">
  <dg:hasURI rdf:resource="http://grid02.softeco.it/kwfgrid
      Ctm/data/wbc/genoa.wbc"/>
</dg:RemoteFile>
<ds:WeatherBoundaryConditionsData rdf:ID="genoa_wbc_data">
  <ds:ofCity rdf:resource="&ar;#Genoa"/>
</ds:WeatherBoundaryConditionsData>
<dg:DataObject rdf:ID="genoa_wbc">
  <dg:contains rdf:resource="#genoa_wbc_data"/>
  <dg:isStoredIn rdf:resource="#genoa_wbc_file"/>
  <dg:hasFormat rdf:resource="&ds;#WBC"/>
</dg:DataObject>
```

Fig. 1: Proposed architecture of Grid Organizational Memory Peer to Peer overlay.

4 Integration with Grid Organizational Memory Knowledge Base

Our GOM-P2P architecture is essentialy unstructured due to the nature of Grid resoure descriptions which should belong with the resources. Peer-to-peer infrastructure will be connected with GOM Engines by specific adapter component. Adapter will provide a bridge between GOM Engines and query resolver. This part of P2P infrastructure is responsible for analysing the incoming query, and possibly translating it into a set of basic RDF queries, and optimizing them in order to query minimal possible number of nodes. For instance, if the query is about a particular resource, than it can be simply resolved against DHT index which GOM Engines contain or could contain such information. However some queries won't be that simple, for instance range queries, when the actual value of the ID is not know. The query resolution module is also responsible for merging the query results from the peers that hold the information about the resource queried and to perform a join operation on the obtained information. It can be interesting to analyse the queries that could be partitioned and passed along the answer path, that means the part of the query that should be resolved first is passed according to the DHT to the next node, and that node resolves what it can, and passes the remaining query further. The last node can for instance return the answer to the first node. It is important to notice, that DHT nodes will not hold local triples, but the ones from the global space that are associated with it's index through some hashing function. The DHT thus provides a mapping between resource ID from RDF model of the registry and the ID of the GOM Engine which contains *some* information on that resource. The motivation for distribution of the index itself lies in the fact that its global size can be rather big. In GOM-P2P architecture, that is currently developed, the model can be stored in memory or in database, and can have a reasoner attached to it. The reasoner can work on local model only. In the future evalution some more advanced techniques as distributed reasoning could be tested (e.g. using e-connections from Pellet).

5 Conclusions and Future Work

The paper presented an early work vision and architecture of a distributed knowledge base for the Grid. The vision involves an already working knowledge base component called Grid Organizational Memory developed within K-Wf Grid project, which due to its architecture can be relatively easily integrated with a Peer to Peer overaly network and thus made more distributed and scalable. The future work will include careful design and implementation of the system using most appropriate solution available in the area of P2P.

Acknowledgements. This reasearch has been done in the framework of EU IST-2004-511385 K-Wf Grid project. AGH University of Science and Technology grant is also acknowledged.

References

1. K-Wf Grid project homepage, `http://www.kwfgrid.net`.
2. Kryza, B., Majewska, M., Slota, R., and Kitowski, J., Unifying grid metadata representations through ontologies, In Wyrzykowski, R., Dongarra, J., Meyer, N., and Wasniewski, J. (Eds), *PPAM*, *Lecture Notes in Computer Science*, vol. 3911, pp. 683–690, Springer, 2005.
3. Kryza, B., Pieczykolan, J., and Kitowski, J., Grid organizational memory: A versatile solution for ontology management in the grid, In *2nd International Conference on e-Science and Grid Computing, Dec. 4-6, 2006, Amsterdam, Netherlands, (C) IEEE Computer Society Press*, In print.
4. Mastroianni, C., Talia, D., and Verta, O., A super-peer model for building resource discovery services in grids: Design and simulation analysis., In Sloot et al.

[19], pp. 132–143.

5. Garcs-Erice, L., Biersack, E.W., Ross, K.W., Felber, P., and Urvoy-Keller, G., Hierarchical peer-to-peer systems, *Parallel Processing Letters*, 13(4):643-657, 2003.
6. Datta, A., Girdzijauskas, S., and Aberer, K., On de bruijn routing in distributed hash tables: There and back again., In *Peer-to-Peer Computing* [20], pp. 159–166.
7. Kantere, V., Tsoumakos, D., and Roussopoulos, N., Querying structured data in an unstructured p2p system, In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pp. 64–71, New York, NY, USA, 2004, ACM Press.
8. Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S., Search and replication in unstructured peer-to-peer networks, In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pp. 84–95, New York, NY, USA, 2002, ACM Press.
9. M. Castro, M.C. and Rowstron, A., Debunking some myths about structured and unstructured overlays, In *Proc. of NSDI'05, Boston, MA, USA, May 2005.*
10. Ding, H., Sølvberg, I., and Lin, Y., A vision on semantic retrieval in p2p network., In *AINA (1)* [21], pp. 177–182.
11. and, J. B., A metadata model for semantics-based peer-to-peer systems, In et al., K.A. (Ed.), *Semantics in Peer-to-Peer and Grid Computing – SemPGRID, collocated with the 2003 WWW Conference*, Budapest, 2003.
12. Babik, M. and Hluchy, L., Towards a scalable grid ontology repository, In Bubak, M., Turala, M., and Wiatr, K. (Eds), *Proc. of 4th Cracow Grid Workshop*, pp. 101–108, Cracow, Poland, December 2004, ACK Cyfronet-AGH.
13. Xiao, H. and Cruz, I.F., Ontology-based query rewriting in peer-to-peer networks, In *In Proc. of the 2nd Int. Conf. on Knowledge Engineering and Decision Support*, pp. 11–18, 2006, TO READ.
14. Cai, M., Frank, M.R., Yan, B., and MacGregor, R.M., A subscribable peer-to-peer rdf repository for distributed metadata management., *J. Web Sem.*, 2(2):109–130, 2004.
15. Majkic, Z., Weakly-coupled ontology integration of p2p database systems., In Zaihrayeu and Bonifacio [22].
16. Bonifacio, M., Bouquet, P., Busetta, P., Danieli, A., Don'a, A., Mameli, G., and Nori, M., Keex: A peer-to-peer solution for distributed knowledge management., In Zaihrayeu and Bonifacio [22].
17. Zhuge, H., Sun, X., Liu, J., Yao, E., and Chen, X., A scalable p2p platform for the knowledge grid., *IEEE Trans. Knowl. Data Eng.*, 17(12):1721–1736, 2005.
18. Heine, F., Hovestadt, M., and Kao, O., Towards ontology-driven p2p grid resource discovery., In Buyya [23], pp. 76–83.
19. Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., and Bubak, M. (Eds), *Lecture Notes in Computer Science*, vol. 3470, Springer, 2005.
20. *4th International Conference on Peer-to-Peer Computing (P2P 2004), 15-17 August 2004, Zurich, Switzerland*, IEEE Computer Society, 2004.
21. IEEE Computer Society, 2004.
22. Zaihrayeu, I. and Bonifacio, M. (Eds), *Proceedings of the MobiQuitous'04 Workshop on Peer-to-Peer Knowledge Management (P2PKM 2004), Boston, MA, USA, Aug 22, 2004.*, *CEUR Workshop Proceedings*, vol. 108, CEUR-WS.org, 2004.
23. Buyya, R. (Ed.), *5th International Workshop on Grid Computing (GRID 2004), 8 November 2004, Pittsburgh, PA, USA, Proceedings*, IEEE Computer Society, 2004.

TeToN – a Jena-Based Tool for Text-to-Ontology Approach

Marcin Kuta[1], Stanisław Polak[1], Barbara Palacz[1], Tomasz Miłoś[1], Renata Słota[1], and Jacek Kitowski[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Kraków, Poland
[2] Academic Computer Centre CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków, Poland
emails: {mkuta, polak, rena, kito}@agh.edu.pl

Abstract

The article presents Text to oNtology (TeToN) tool created to deal with a problem of knowledge acquisition from natural language short text notes.

This kind of knowledge is used for extension of existing knowledge base which is used for coordinating workflow execution in grid environment. The short text notes provide additional feedback from the user, which does not have to understand grid internals. This feedback may be helpful for improvement of grid operations.

Knowledge provided by user feedback (in form of short user notes) is extracted by means of natural language analysis tools. Then it is integrated with background knowledge (stored in ontologies) according to our statistical measure.

1 Introduction

Current development of grid environments and grid computing are focused on extensive use of knowledge for efficient exploitation of both hardware and software grid resources, with ontological formalism for knowledge representation applied typically. Good examples in this field are EU IST projects, like OntoGrid [19], Inteligrid [10] and K-WfGrid [13]. In the last mentioned project the application workflows are composed dynamically with services selected automatically according to a user request. The workflows are subject of opinion of users. One way of expressing the evaluation score given to the user is to use notes in natural language, which later on can be presented as a tip to another one being in the same context of work.

In this paper we present a spin-off of the research performed within K-Wf Grid, which consists of making knowledge of the tips more useful formally. For this purpose we propose to study creation and extension of existing ontologies by understanding natural language sentences. Since input sentences provide additional feedback from the user, it may be helpful for improvement of operations like workflow composition.

The TeToN (Text To oNtology) tool is intended to acquire new knowledge by analysing only short pieces of information in form of sentences in English, in the future devoted to the problems addressed by projects like K-WfGrid, making use of their existing system knowledge.

The rest of the paper is organized as follows. Section 2 presents state of the art in the field of learning ontologies from text. Section 3 introduces architecture of the TeToN system and presents process of knowledge acquisition. A suitable example of ontology extension is supplied in Section 4. The last section concludes the paper.

2 State of the Art

The studies on natural language text processing to ontology focus on methods that require large text corpora [14, 16] and extract dependencies by means of regular expressions [7, 9], association rules [17], conceptual clustering [5], ontology pruning [12] or concept learning [8].

Regular expressions method relies on remark that relations can be identified by fixed patterns, e.g. pattern "such NP as {NP,}* {(or | and)} NP" indicates generalization. Extracted dependencies are lexical relations like generalization [9] or meronymy [2].

Association rules method help to discover non-taxonomic relation between concepts. Using a concept hierarchy as a background knowledge, new relations are created basing on statistics of co-occurrences of terms in texts.

In conceptual clustering method concepts taken from input are grouped according to the semantic distance between them. Two concepts are considered to belong to the same group if their semantic distance is below predefined threshold.

Ontology pruning method uses core ontology (e.g. Wordnet) and corpus of text. New concepts are identified over text using natural language analysis tools. The resulting ontology is then pruned, basing on a statistical methods. This method is leveraged by Text2Onto from KAON tool suite [3].

Concept learning method, developed by Maedche, uses techniques already mentioned (pattern based extraction, conceptual clustering, etc.). Due to them new concepts are extracted and existing ontology is incrementally updated.

Due to few amount of input data the above methods are not directly relevant for short text notes processing. We propose our original approach, originating from regular expressions technique combined with ontology pruning method for amelioration of results.

3 TeToN Architecture

Understanding language means knowing what concepts a word or phrase stands for and knowing relations to link those concepts together in a meaningful way.

In order to perform this task in TeToN an input sentence is converted to ontology elements. If the sentence contain some new facts, these ontology elements can then be integrated with existing knowledge base. To achieve this several transformations are performed: syntactic and semantic analysis, concept

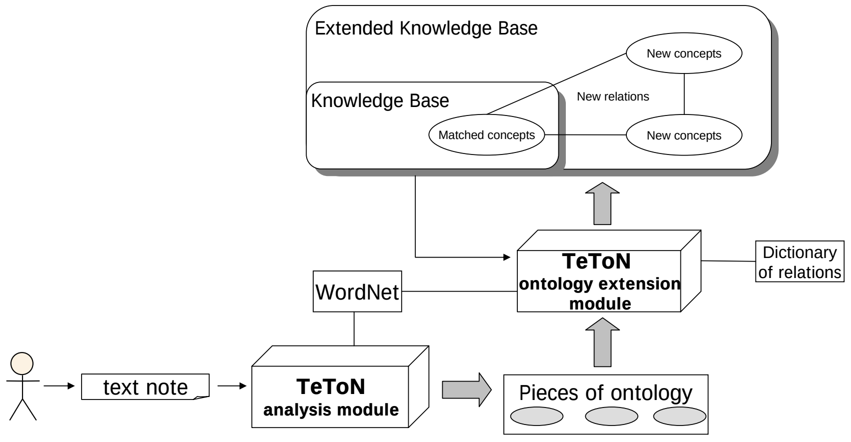


Fig. 1: General architecture of TeToN.

matching, relation matching and knowledge evolution, which are described below in detail.

Figure 1 presents general architecture of TeToN. The TeToN analysis module converts text notes to pieces of ontology. Some of these pieces are matched with concepts and relations in the Knowledge Base by an extension module. The rest of pieces can be entered to Knowledge Base as new concepts and relations.

In the implementation the Jena library [11] was used to manipulate the Knowledge Base stored in OWL [20].

3.1 Syntactic and Semantic Analysis

The result of syntactic analysis is a conversion of user note to a parse tree, whose nodes are marked by part of speech (POS) tags in Penn TreeBank notation [22]. To perform this task statistical parser trained on English treebank [4] has been used. Sentences in other languages can also be processed. In this case the user has to provide however a suitable treebank in the desired language for parser training process.

The main task of semantic analysis is word sense disambiguation. The sense of each word from the input sentence is learned according to WordNet taxonomy [6]. This task is done with help of the SenseRelate library [21] provided with semantic relatedness measure. TeToN uses by default Lesk measure [1] provided by WordNet Similarity Perl module [23] with the size of the context window set to 3 (which means that only one preceding word and one succeeding word is taken into consideration during the word disambiguation process). Additionally words' base forms from the WordNet's lexical database are retrieved in order to facilitate further processing.

The next step relays on the previously created parse tree structure. Each

Concept name	**WordNet keywords**
ModernSupersonicAirplane	**modern#a#1**, **supersonic#a#1**, **airplane#n#1**, aeroplane#n#1, plane#n#1
...	...

Fig. 2: The exemplary line of index.

noun or adjective phrase from the parse tree has to result with a piece of ontology containing at least one concept (subject or object in RDF terminology), while each verb phrase has to result with a piece that contains at least one relation (property in RDF terminology). All potential concepts and relations are identified in bottom-up manner. At each node of the parse tree a new piece of ontology is constructed by merging the ontology pieces created in its descendants.

3.2 Concept and Relation Matching and Knowledge Evolution

Having obtained pieces of ontology (i.e. candidates for concepts and relations), we extend with them Knowledge Base. Because concepts can be represented by different names there is a danger of entering redundancy to Knowledge Base. To avoid it, we try to match concept names from the created pieces of ontology with the concept names from Knowledge Base. Each concept from the created pieces of ontology is assigned a measure, M, indicating the accuracy of the match to concept from Knowledge Base (the higher measure the more accurate the match). Knowledge Base is enriched with the new concept if M is below a certain threshold. Otherwise we do not have to enter new name to Knowledge Base.

To perform this, first an index from existing knowledge-base ontology is created with help of Lucene library ([15]). The index associates concepts from this ontology with synsets – lists of WordNet keywords:

The measure M is defined according to the following formula:

$$M = n_e W_e + n_s W_s \tag{1}$$

where:

n_e – number of exactly matching Wordnet keywords (i.e. bolded keywords in figure 2),

n_s – number of synonymously matching Wordnet keywords (i.e keywords selected in normal font in figure 2),

W_e – weight of an exact match,

W_s – weight of a synonymous match.

W_e and W_s are parameters, and should satisfy condition $W_e \geq W_s$. Between variables n_e and n_s inequality $n_e + n_s \leq \min(n_k, n_t)$ holds, where:

n_k – number of words in piece of ontology we try to match to a concept from

Relation name	**Pattern**	**Target type**	**Relation range**
propertyOf	be#v#1	Adjective phrase	Feature
isA	be#v#1	Noun phrase	Vehicle
partOf	include#v#1, contain#v#1, comprise#v#1	Noun phrase	-

Tab. 1: Dictionary of relations.

Knowledge Base,
n_t – number of words matched concept name is build of.

We also have to indicate the threshold of acceptable matches. Each match has to satisfy the following inequality in order to be accepted:

$$M > T \cdot \min(n_k, n_t) \cdot W_e \tag{2}$$

where:
T – threshold value,
$\min(n_k, n_t) \cdot W_e$ – maximum possible metric value for this match

Joining equations (1) and (2) we get the acceptance condition:

$$\frac{n_e W_e + n_s W_s}{\min(n_k, n_t) \cdot W_e} > T \tag{3}$$

Our assumptions guarantee that the left side of the formula will be in range $[0, 1]$.

In the next stage proper relations for considered concepts (i.e. matched and newly created) must be selected. In order to choose the proper relation for considered concept one has to take into account the following data from input sentence:

1. words from verb phrase (to have the idea what relation is about)
2. target type, i.e. type of phrase following verb phrase (to take into account type of arguments relation can take)

Because creation of a good set of relations which would describe well modelled reality and creation of good relation names are not trivial tasks, we suppose that a special dictionary is already supplied (by the user or administrator). This dictionary describes relations by theirs names, patterns, type of arguments they can take, as shown below:

Ontology pieces created from verb phrases are compared with patterns from dictionary. The target type is also considered and the appropriate relation name is selected. If there are more matching relations in dictionary, all of the possibilities are presented to the user for acceptation. Otherwise, if there is no matching relation then a new relation name is created and proposed to the user. New relation name is compound of all the significant words in a verb phrase.

Having selected concepts and matching relations the ontology extensions with triples (subject, predicate, object) are proposed to user.

4 Example of Knowledge Acquisition

Let's consider the following example. Suppose the below Knowledge Base is given:

```
<owl:Class rdf:Id="Vehicle">
<owl:Class rdf:Id="Feature">
<Feature rdf:Id="Comfortable">
<owl:Class rdf:Id="Airplane">
        <owl:subClassOf rdf:resource="#Vehicle"/>
</owl:Class>
<owl:Class rdf:Id="ModernSupersonicAirplane">
        <owl:subClassOf rdf:resource="#Airplane"/>
</owl:Class>
<owl:Class rdf:Id="Automobile">
        <owl:subClassOf rdf:resource="#Vehicle"/>
        <owl:Restriction>
                <owl:onProperty>
                        <owl:Property rdf:about="#propertyOf"/>
                </owl:onProperty>
                <owl:hasValue rdf:resource="#Comfortable"/>
        </owl:Restriction>
</owl:Class>
<owl:Property rdf:Id="propertyOf">
        <owl:range rdf:resource="Feature"/>
</owl:Property>
```

and let's consider the following sentence as a user note:

Racing cars and supersonic planes are very fast and quite expensive.

In the syntax analysis phase the following parse tree is created:

From the parse tree we obtain ontology pieces: `(racing#n#1 car#n#1)`, `(supersonic#a#1 plane#n#1)`, `(very#r#1 fast#a#1)`, `(quite#r#1 expensive#a#1)` and `(be#v#1)`. Assuming threshold value $T = 0,6$ ontology piece `(supersonic#a#1 plane#n#1)` is matched with concept `ModernSupersonicAirplane`, thus is not entered to Knowledge Base. Using dictionary from figure 3 we find that ontology piece `(be#v#1)` should be represented by relation `partOf`.
As an outcome the tool proposes new triples to extend the knowledge-base ontology with:

```
(propertyOf, RacingCar, VeryFast)
(propertyOf, RacingCar, QuiteExpensive)
(propertyOf, ModernSupersonicAirplane, VeryFast)
(propertyOf, ModernSupersonicAirplane, QuiteExpensive)
```

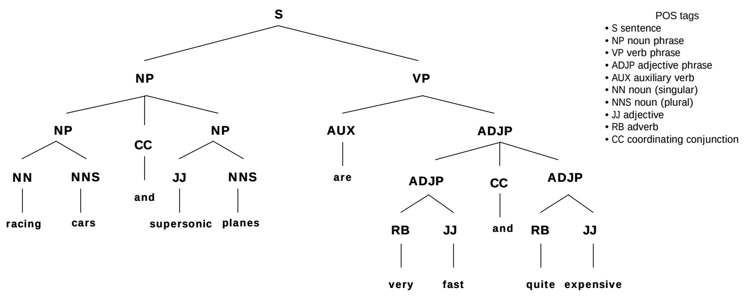


Fig. 3: Parse tree of exemplary input sentence.

5 Conclusions and Future Work

Acquiring knowledge from end user's text notes we must be sure than we can trust them completely. To make efficient use of theirs opinions it is suggested to take account of opinions appearing sufficiently often. The other solution to this problem would be to confer ontology evolution task to knowledge engineer for final acceptance. It is also hard to propose good vocabulary (concepts or relations may be represented by plethora of names). Due to these difficulties our tool needs engineer assistance for final acceptance of knowledge evolution.

In future we plan to perform tests on Grid Organizational Memory of K-Wf Grid and analyse relevance, consistency and usefulness of acquired knowledge. We would like to improve process of syntactic and semantic analysis by identifying phrasal verbs in aim to provide more accurate propositions of knowledge extensions. Analysis of N-best parse tree, in case the most probable one is not appropriate is also planned. We want to continue our research in order to relax from user intervention and make ontology extension process fully automatic.

Acknowledgements. This research has been supported by Department of Computer Science, AGH, University of Science and Technology, Cracow and partially by European Commission Project IST-2004-511385 K-Wf Grid.

References

1. Banerjee, A., Pedersen, T., An Adapted Lesk Algorithm for Word Sense Disambiguation using WordNet, In the Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, February 17-23, 2002, Mexico City.
2. Berland, M., Charniak, E., Finding Parts in Very Large Corpora, In Proceedings of the 37th Annual Meeting of the ACL, pp 57-64, 1999.
3. Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., and Zacharias V., Kaon – towards a large scale

semantic web, In Bauknecht, K., Tjoa, A. M., and Quirchmayr, G. (Eds), Proc. of E-Commerce and Web Technologies 3rd Intl. Conf., EC-Web 2002, Aix-en-Provence, France, LNCS, vol. 2455, pp. 304-313, Springer, 2002.

4. Charniak, E., A Maximum-Entropy-Inspired Parser, Proceedings of North American Chapter of the Association for Computational Linguistics, 2000
5. Faure D., Poibeau T., First Experiments of using semantic knowledge learned by ASIUM for information extraction task using INITEX, In: Staab S., Maedche A., Nedellec C., Wiemer-Hastings P. (eds.) Ontology Learning Workshop ECAI-2000, pp 7-12.
6. Fellbaum, C., WordNet An Electronic Lexical Database, Computational Linguistics, 25, 2 (1998), 292-296, `http://wordnet.princeton.edu`
7. Hahn U., Schnattinger K., Towards Text Knowledge Engineering, In AAAI'98/IAAI'98 Proceedings of the 15th National Conference on Artificial Intelligence and the 10th Conference on Innovative Applications of Artificial Intelligence, 1998.
8. Hahn U., Schulz S., Towards Very Large Terminological Knowledge Bases: A Case Stude from Medicine, Hamilton HJ. (ed.), Advances in Artificial Intelligence, 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI 2000), Montreal, Quebec, Canada. Lecture Notes in Computer Science LNCS 1822, Springer-Verlag, Berlin, Germany, pp. 176-186.
9. Hearst, M.A., Automatic acquisition of hyponyms from large text corpora, In Proceedings of the 14th International Conference on Computational Linguistics, pp. 539-545, 1992.
10. Inteligrid website: `http://www.inteligrid.com/`
11. Jena - A Semantic Web Framework for Java, `http://jena.sourceforge.net/`
12. Kietz, J.U., Maedche A., Volz R., A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet, In: Aussenac-Gilles, N., Biebow, B., Szulman, S., (eds.), EKAW 2000 Workshop on Ontologies and Texts. Juan-Les-Pins, France. CEUR Workshop Proceedings 51:4.1-4.14, Amsterdam, The Netherlands.
13. K-Wf Grid website: `http://www.kwfgrid.eu/`
14. Liu, W., Weichselbraun, A., Semi-Automatic Ontology Extension Using Spreading Activation, Journal of Universal Knowledge Management, vol. 0, no. 1 (2005), pp. 50-58.
15. Lucene text search engine, `http://lucene.apache.org/`
16. Maedche, A., Staab, S., Discovering Conceptual Relations from Text, In: W.Horn (ed.): ECAI 2000. Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, August 21-25, 2000. IOS Press, Amsterdam, 2000.
17. Maedche, A., Staab, S., Mining Ontologies from Texts, In: R. Dieng, O. Corby (eds.): EKAW 2000. 12th International Conference in Knowledge Engineering and Knowledge Management, Juan-Les-Pins, France. Lecture Notes in Artificial Inteligence LNAI 1937, Springer-Verlag, Berlin, Germany, pp. 189-202.
18. Michelizzi, J., Semantic Relatedness Applied to All Words Sense Disambiguation, Master of Science Thesis, Department of Computer Science, University of Minnesota, Duluth, July, 2005.
19. OntoGrid website: `http://www.ontogrid.net/`
20. OWL Web Ontology Language Overview, `http://www.w3.org/TR/owl-features/`
21. SenseRelate library, `http://senserelate.sourceforge.net`
22. The Penn Treebank website: `http://www.cis.upenn.edu/~treebank/home.html`
23. WordNet Similarity library, `http://sourceforge.net/projects/wn-similarity`

OnTal – the Tool for Ontology Extension in Knowledge-Based Grid Systems

Renata Slota[1], Joanna Zieba[1], Maciej Janusz[1], and Jacek Kitowski[1,2]

[1] Institute of Computer Science AGH-UST, Mickiewicza 30, 30-059 Krakow, Poland
[2] Academic Computer Centre CYFRONET AGH,
Nawojki 11, 30-950 Krakow, Poland

Abstract
This paper presents an approach to knowledge evolution developed in the K-WfGrid project and OnTal – the tool which implements it. The tool is designed to support the process of Grid services' semantic registration with help of ontology alignment methods applied to the problem of local ontology schema extension. The paper describes OnTal's functionalities along with its theoretical background and implementation details.

1 Introduction

Semantic grids are a field of intensive research nowadays. The goal of Semantic Grid community is to enhance 'classical' Grid with explicit, machine-processable knowledge, thus enabling semantic interoperability and seamless cooperation in dynamically created Virtual Organizations [1]. Several Grid projects pursue this aim.

The business-oriented InteliGrid [2] is designed to provide a grid-based interoperability infrastructure for industrial projects, which rely on cooperation of multiple business partners. Another project, OntoGrid [3], focuses on establishing a methodology for creating knowledge-enabled Grid systems and creating a reference architecture of a Semantic Grid.

Finally, K-WfGrid project focuses on creation of a knowledge-based framework which enables automated workflow applications' composition, execution and monitoring in Grid environment [4]. Knowledge is stored in Grid Organization Memory (GOM), which is a distributed knowledge base containing multiple general and domain specific ontologies, as well as individual registries, concerning five main areas of functioning of a Grid system. Semantic descriptions of Grid services, which are contained in GOM's Service Registry, play an important role in the functioning of the system because workflow composition relies on automated analysis of services' characteristics. Growth and evolution of knowledge stored in Grid Organization Memory improves the possibilities of workflow creation. Because of this, the K-WfGrid Project has developed various tools for knowledge acquisition – among them the Knowledge Assimilation Agent (KAA), which extracts semantic information from the history of previous workflow executions [5], and its part, OnTal – the tool for ontology extension presented in this paper.

2 Tool's Overview and Implementation

2.1 Overview

OnTal is a semi-automatic tool supporting the process of Grid services' registration in K-WfGrid system. It can be used by Virtual Organizations, whose members cooperate and share various resources, particularly services, which may be assembled into workflow applications with the help of semantic technologies. Consequently, all services in K-WfGrid system must be represented by their semantic descriptions (using the OWL-S schema), stored in the Service Registry. New service's description, particularly if reused from outside of the K-WfGrid users' community, may refer to concepts from external ontologies. All such concepts need to be included in Grid Organization Memory to enable workflow construction which entails reasoning over the knowledge about services. On the other hand, GOM knowledge base must stay consistent so eventual changes of its ontologies should be limited and controlled.

This situation creates requirements for OnTal: the tool should identify concepts from external ontologies in the OWL-S description and propose the way in which they may be added to GOM. More specifically, a concept can be either added to GOM together with its neighbourhood from the source ontology, or mapped onto an existing GOM concept. Ontology similarity methods (described in the following subsection) are used to determine how a certain concept should be handled and generate mapping propositions which are then assessed by the user. OnTal's user is a VO member, with a basic understanding of ontologies, responsible for providing new services. To enforce control over semantic resources, eventual changes should be also accepted by the GOM's administrator before introducing them to the ontologies. The diagram in Fig. 1 summarizes OnTal's use cases.

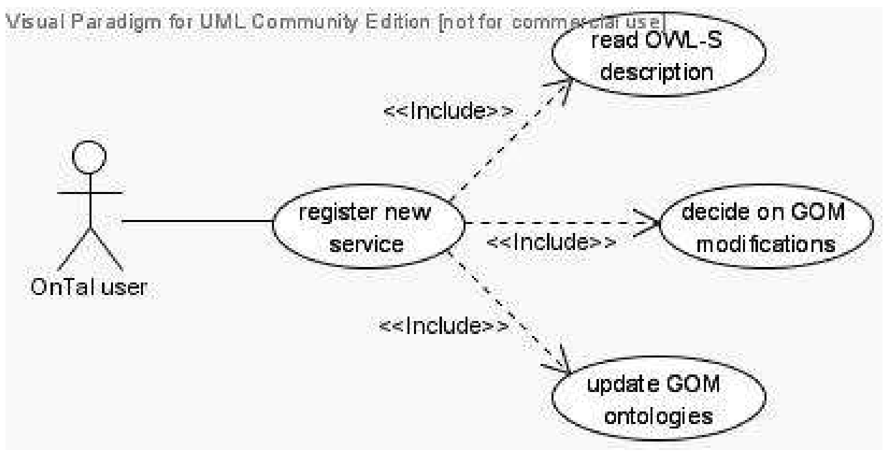


Fig. 1: OnTal's use-case diagram.

2.2 Ontology Alignment Methods

OnTal uses ontology alignment methods to generate propositions regarding the extension of one or more GOM ontologies with new concepts. Aligning two ontologies means finding correspondences between their concepts, relations and instances. Alignment methods, which base on different similarity measures[6], are used to determine if a new concept already has an equivalent in GOM. In such a case, the external concept can be mapped onto its counterpart; otherwise, it has to be added to one of GOM's domain ontologies together with its neighbourhood. The alignment algorithm used by OnTal has been described in a detailed way in [7, 8].

OnTal uses several ontology similarity measures. First, a combination of two lexical measures is applied to the product of the set of external concepts and the set of all concepts from GOM. These measures identify if two words have any tokens in common (token similarity measure) and if they are generally similar to each other (edit similarity). When the lexical similarity value is higher than the minimum (which can be configured by the user), structural similarity is determined for the given pair of concepts.

Structural similarity is computed with use of the cosine weight, which takes into account the relevance of attributes, relations and neighboring nodes of two ontological concepts. Pairs of concepts, which have both similarity values higher than the minimal value, are presented to the user as 'mapping candidate pairs'.

Introducing new entities into the Grid Organization Memory must be done in a controlled and limited way. However, in order to preserve their original semantics, they must be added to GOM together with some surrounding. In the presented approach the surrounding consists of all superclasses, subclasses, neighboring nodes and instances of the processed concept. When such subgraph of the external ontology is isolated, it is also searched for further similarities and possible mappings. However, in contradiction to global approaches like *Similarity Flooding* [9], this action is not recurrent and resulting changes are local.

2.3 Implementation

OnTal has been implemented as a Java applet and deployed inside the K-WfGrid Portal. The portal gathers multiple tools created in the K-WfGrid framework for the purpose of workflow composition, execution, monitoring and assimilation of knowledge. However, as the tool is only loosely dependent on Grid Organization Memory's ontologies, it can be used as a stand-alone applet, opened in a local Web browser or the JDK's appletviewer tool.

2.3.1 Ontology Processing

Ontology processing is facilitated by the Jena Ontology API [10]. Jena is a Semantic-Web framework and provides broad functionality, including RDF, RDFS and OWL language support, RDQL query processing and a rule-based inference engine. In the OnTal tool, Jena's facilities for OWL processing have been widely used.

Ontologies are represented as *OntModel* objects which can be created on the basis of an ontology file or URL. Many operations can be performed on these models, e.g. listing of all classes, relations and particular instances. In the process of comparing ontology entities and deducing about their similarity, Jena's representation of ontological concepts as *OntClass* objects has been used. The information about the neighbors, attributes, subclasses, superclasses etc. of such object can be easily retrieved and has been used for computing structural similarity, as well as for extracting neighbourhood graphs to be visualized. In addition to Jena, OWL-S API [11] has been used for initial analysis of service descriptions provided by the user. This library focuses on processing of OWL-S and provides a simple way to parse service descriptions and extract information from them.

2.3.2 Ontology Visualization

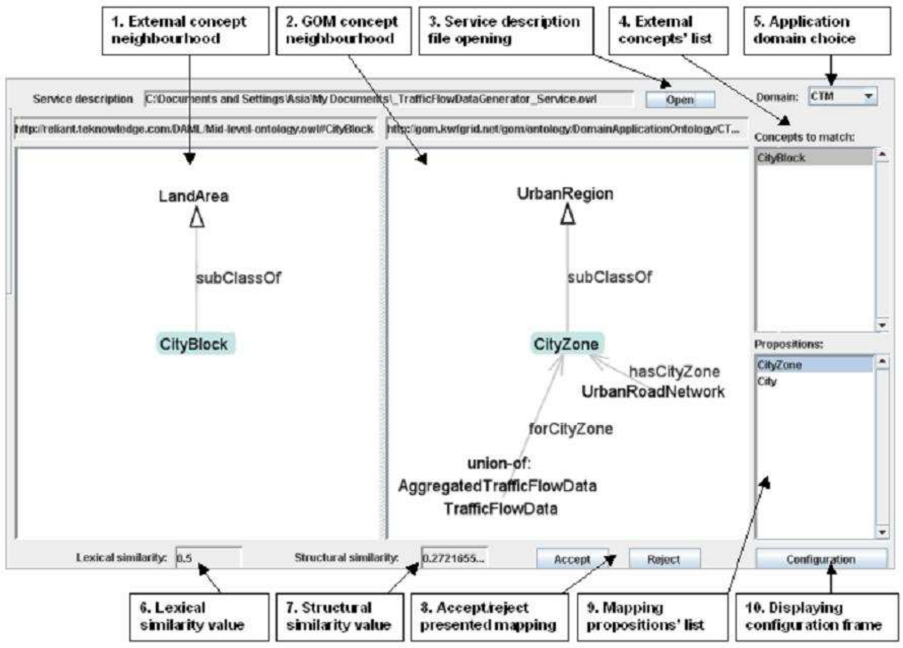
Visualization of ontologies is equivalent to visualization of large graphs (concepts/instances presented as nodes and properties/relationships as edges of a graph) [12]. Most ontology tools have their own visualization modules (e.g. Protege which uses OntoViz), which in most cases present the ontology as an plain graph. For small ontologies standard UML modelling tools can be used. Finally, multiple libraries for graphs visualization are available.

The most complicated problem of ontology visualization is placing ontology elements (objects, properties, relations) on a 2D surface and enable viewing the ontology on different detail levels. Displaying large ontologies in a clear way can also be difficult. One solution to the size problem is clusterization [12], which means that relatively strong connected nodes and their edges are represented as one node, which decreases the complexity of visualization. Only when the user wants to focus on a particular cluster, its content may be visualized as well. Another solution is to visualize ontology as a set of concentric spheres (the nodes are put on the sphere surface and the edges can cross the sphere in any direction) as in the OntoSphere project. The presented methods may be used jointly on different levels of detail.

In the OnTal tool, the Prefuse toolkit [13] was used. It was chosen because of its flexibility and variety of graph layouts. As the visualization is necessary for showing only the neighbourhood of given concepts, the graphs are not big; however, the user may extend the visible area of the ontology with the neighbourhood of any already visible node. Additionally, the graphs can be scrolled and magnified/minimized. The input for visualization is stored in the form of Jena *OntModel* objects; concepts and relations are extracted from these models and respective Prefuse *Node* and *Edge* objects are created in the graph.

3 Details of OnTal's Operation

This section presents functioning of OnTal tool from the user-centric perspective. Main phases are described with reference to the graphical user interface in Fig. 2; the activity diagram in Fig. 3 presents the algorithm of the tool's operation.


Service description: C:\Documents and Settings\Asia\My Documents_TrafficFlowDataGenerator_Service.owl

Domain: CTM

http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl#CityBlock

http://gom.kwfgrid.net/gom/ontology/DomainApplicationOntology/CT...

Concepts to match: CityBlock

Propositions: CityZone, City

Lexical similarity: 0.5

Structural similarity: 0.2721655...

1. External concept neighbourhood
2. GOM concept neighbourhood
3. Service description file opening
4. External concepts' list
5. Application domain choice
Open
LandArea
subClassOf
CityBlock
UrbanRegion
subClassOf
CityZone
hasCityZone
UrbanRoadNetwork
forCityZone
union-of:
AggregatedTrafficFlowData
TrafficFlowData
Accept
Reject
Configuration
6. Lexical similarity value
7. Structural similarity value
8. Accept/reject presented mapping
9. Mapping propositions' list
10. Displaying configuration frame

Fig. 2: OnTal's user interface.

Configuration of ontology alignment. Before the start of ontology alignment process, the user should choose the domain of the newly registered service (button 5) – now it can be one of K-WfGrid pilot applications – Coordinated Traffic Management (CTM), Flood Forecasting and Simulation (FFSC) or Enterprise Resource Planning (ERP). Moreover, s/he can configure the similarity measures after clicking on 'Configuration' (button 10) – for all lexical similarity measures, the weights and minimal values can be set. By default, similarity measures have equal weight (0.5) and their threshold value is 0.2.

OWL-S service description processing. OWL-S description of some service is processed by the OnTal tool after it is read in from the file indicated by

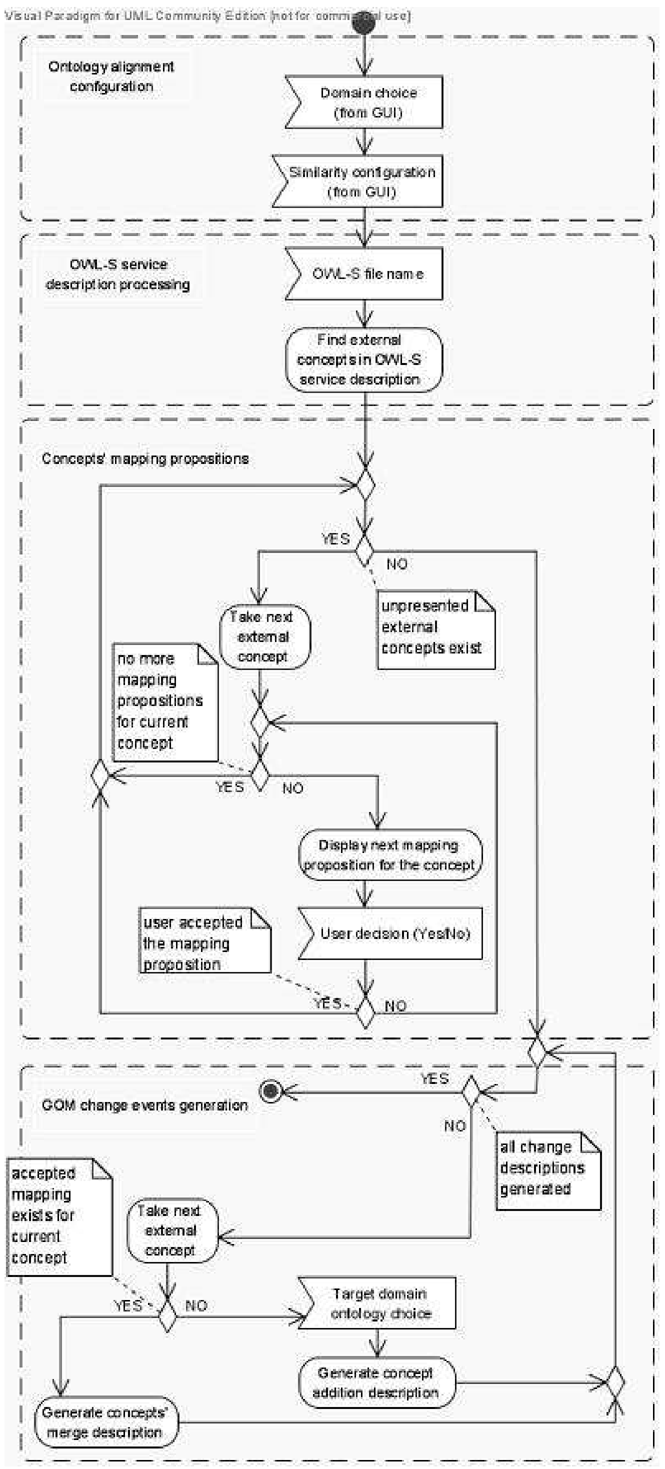

Visual Paradigm for UML Community Edition [not for commercial use]
Ontology alignment configuration
Domain choice (from GUI)
Similarity configuration (from GUI)
OWL-S service description processing
OWL-S file name
Find external concepts in OWL-S service description
Concepts' mapping propositions
YES
NO
Take next external concept
unpresented external concepts exist
no more mapping propositions for current concept
YES
NO
Display next mapping proposition for the concept
user accepted the mapping proposition
User decision (Yes/No)
YES
NO
YES
GOM change events generation
NO
all change descriptions generated
accepted mapping exists for current concept
Take next external concept
YES
NO
Target domain ontology choice
Generate concept addition description
Generate concepts' merge description

Fig. 3: Activity diagram of OnTal's functioning.

the user (button 3). The loaded OWL-S file is scanned to identify concepts from external ontologies. This activity may take about a minute due to the HTTP connection to remote ontologies and the necessary search for same/similar entities in Grid Organization Memory. A progress dialog is displayed to show the process status.

Concepts' mapping propositions. When the service's OWL-S description is already analyzed, the concept mapping propositions generated by OnTal can be displayed. The list of external concepts is shown (panel 4), together with a list of mapping propositions for the currently highlighted external concept (panel 9). The user is served with visualized mapping propositions (panels 1 and 2) and information on concepts' similarity values (fields 6 and 7). The mapping can be either accepted or rejected by clicking the relevant buttons (panel 8). After mapping acceptation, the external concept is highlighted in green; isolated concepts are marked with red.

GOM ontology change events generation. After finishing the ontology mapping process, the tool generates ontology change events. These events are described by the Change Ontology, which contains concepts describing possible ontology modifications e.g. class addition or property change. Consequently, for all external concepts, an ontological description of necessary changes is generated and displayed to the user who can save it in a file. Later, if the GOM administrator accepts such a modification, it can be introduced to the knowledge base through the Grid Organization Memory's Event Dispatcher [14].

4 Summary

Knowledge extension is an important issue in Semantic Grid systems. In the K-WfGrid project it is facilitated by the Knowledge Assimilation Agent and OnTal tool, which has been presented in this paper. OnTal relies on the ontology alignment methods and uses information about the concept neighbourhoods to assist the user in extending Grid Organizational Memory's domain ontologies. The tool is easy to use because of its semi-automation and produces limited changes to the original ontologies, which can be easily controlled by the GOM's administrator. It has been designed for a specific task and does not aim at present to perform integration of complex ontologies or provide interoperability between them. However, this assumption allowed to adapt the ontology alignment algorithm to the specific of the use-case which results in successful functioning of OnTal tool.

Acknowledgements. This work has been supported by European Commission, Project IST-2004-511385 K-Wf Grid and Institute of Computer Science, AGH University of Science and Technology, Cracow.

References

1. D. De Roure, N.R. Jennings, N.R. Shadbolt; The Semantic Grid: A future e-Science infrastructure, in Grid Computing – Making the Global Infrastructure a Reality, F. Berman, G. Fox, and A.J.G. Hey, Eds.: John Wiley and Sons Ltd., 2003, pp. 437-470.
2. A. Gehre, P. Katranuschkov, R.J. Scherer; Management and integration of virtual enterprise information on grid spaces through semantic web ontologies, in: Proc. of ECPPM2006, Valencia, Spain, 2006.
3. C. Goble, S. Bechhofer; OntoGrid: A Semantic Grid Reference Architecture, CTWatch Quarterly, Volume 1, Number 4, November 2005.
4. K-Wf Grid project (FP6-511385) website, `http://www.kwfgrid.net`.
5. Z. Balogh, E. Gatial, M. Laclavk, M. Maliska and L. Hluchy: Knowledge-based Runtime Prediction of Stateful Web Services for Optimal Workflow Construction, in: Proc. of 6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM 2005, R. Wyrzykowski et. al. eds., 2005, LNCS 3911, Springer-Verlag, pp. 599-607, ISSN 0302-9743, ISBN 3-540-34141-2. Poznan, Poland.
6. A. Maedche and S. Staab; Measuring similarity between ontologies, in: Proc. of the European Conference on Knowledge Acquisition and Management – EKAW-2002, LNCS 2473, Springer-Verlag, pp. 251-263, Madrid, Spain,
7. R. Slota, J. Zieba, M. Majewska, B. Kryza, J. Kitowski; Ontology Alignment and Ontology Similarity for Extension of Service Ontology in Grid Environment, in: Proceedings of Cracow Grid Workshop – CGW'05, M. Bubak, M. Turala, K. Wiatr (Eds.), November 20-23 2005, ACC Cyfronet UST, 2006, Krakow, pp. 41-48.
8. R. Slota, J. Zieba, B. Kryza, J. Kitowski; Knowledge Evolution Supporting Automatic Workflow Composition, accepted for 2nd IEEE Conference on e-Science and Grid Computing, Amsterdam 2006.
9. S. Melnik, H. Garcia-Molina, E. Rahm; Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching, in: ICDE '02: Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2002, p. 117.
10. Jena homepage, `http://jena.sourceforge.net`.
11. OWL-S API homepage, `www.mindswap.org/2004/owl-s/api/`.
12. C. Fluit, M. Sabou, F. van Harmelen; Ontology-based Information Visualisation, in Visualising the Semantic Web, Vladimir Geroimenko ed., Springer Verlag, 2002.
13. Prefuse toolkit homepage, `http://prefuse.sourceforge.net`.
14. B. Kryza, R.Slota, M. Majewska, J. Pieczykolan, J. Kitowski; Grid organizational memoryprovision of a high-level Grid abstraction layer supported by ontology alignment, The International Journal of FGCS, Grid Computing: Theory, Methods and Applications, vol. 23, issue 3, Mar 2007, Elsevier, 2007, pp. 348-358.

Knowledge Assimilation Agent – Component for Workflow-Based Grid Service Performance Prediction, Workflow Result Estimation and Semi-Autonomic Service Discovery

Zoltán Balogh[1], Martin Šeleng[1], Martin Mališka[1], Ladislav Hluchý[1], Renata Slota[2], Joanna Zieba[2], and Jacek Kitowski[2,3]

[1] Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia
[2] Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland
[3] Academic Computer Centre CYFRONET AGH, Krakow, Poland
email: `balogh@savba.sk`

Abstract

Knowledge Assimilation Agent (KAA) is a knowledge-based component for Grid service workflows, which comprises three basic functionalities: assimilates run-time information about Grid services from different sources and produces performance estimations of future Grid service invocations (the KAA-Web Service), performs past workflow analysis and produces workflow result estimations (the KAA-WXA tool) and discovers new potential services through interactive semi-automatic ontology alignment (the OnTal tool). Individual tools responsible for each of three functionalities are described in the article. Use of KAA functionalities in scope of the K-Wf Grid project is described.

1 Introduction

The K-Wf Grid is a 6th FP IST [1] project which aims at developing a novel knowledge-based approach to service-oriented Grid workflow composition and execution. Knowledge Assimilation Agent (KAA) is one of the main knowledge-producing components in scope of the K-Wf Grid project. KAA comprises functionalities which enable knowledge-supported construction or execution of Grid service-based workflows such as prediction of Grid service performances and estimations of Grid workflow results. KAA also comprises semi-autonomic ontology alignment, which enables discovery of potential services suitable for use in Grid workflows. Tools which implement the above mentioned three most important functionalities of KAA are described further in the article in separate chapters.

2 Performance Estimation of Services for Grid Workflows

Performance estimations about Grid services are needed by several components in the K-Wf Grid platform such as Automatic Application Builder (AAB) [2] and the Scheduler [3]. Based on information provided by KAA, AAB selects most stable hosts out of the all available ones. AAB provides a list of hosts and KAA ranks individual hosts according to their availability and operation stability. Historical information about services are used. The results are passed from AAB to Scheduler, which is another consumer of KAA estimations. Scheduler needs to select which deployment of a service is the best to invoke in a given context. Estimation of service performance is dependent on the invocation context, i.e. what are the invocation parameters and input resources of a service. KAA needs to know the expected invocation context before making the estimation. The central part of KAA is the KAA-Web Service (KAA-WS), which is a stateless web service implementation through which third party components request and retrieve required knowledge. KAA-WS implements the core learning algorithm concretely an extended and improved instance-based learning technique [6-10]. There is an internal ontology model of Grid service performances, resources and used invocation parameters which is called Knowledge Assimilation Schema (KAS). A single service invocation is modeled as a Case (see Fig. 1). Each Case has a Context and Result. Context can be related with Resource properties or invocation parameters. The Result concept models resulting effects of the service invocation. Another concept modeled by KAS is the Profile and Features which enable customization of case retrieval (Fig. 1). KAS extends OWL [15] ontology.

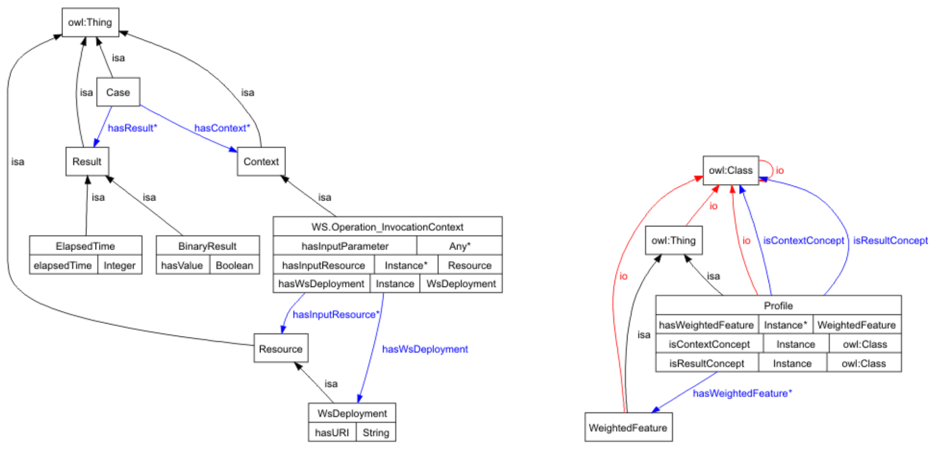


Fig. 1: The Parts of the Knowledge Assimilation Schema (KAS).

KAA consumes information generated by third party components such as Gemini (a monitoring infrastructure) and the Workflow Composition Tool (WCT) and transform them into KAS representation. New information are added upon completion of any grid workflow, which is detected by a component called KAA-WXA. Using a retrieved workflow identifier, a component called KAA-Gemini retrieves information about Grid services invocations and transforms them into KAS representation.

The main scientific innovations of KAA are extension of instance based learning technique of WS performance prediction by enabling retrieval and inclusion of semantic resource properties into the feature vector, enabling specification of WS performance prediction profiles which specify feature vector and result to be estimated and proposition of a novel ontology for IBL-based WS performance prediction together with methodology extending classical Case-Based Reasoning approach.

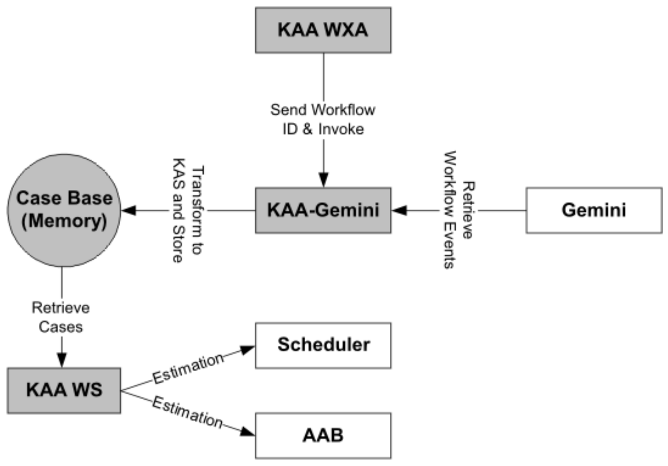


Fig. 2: Data Flow between KAA-WS and other relevant components.

KAA-WS enables estimation of various Grid service performance measures, such as run-time, availability, accessibility or queue wait times. The prediction is based on knowledge from the past such as invocation parameters and metadata about used resources. The extension of the tool is viable through simple extension of the KAS ontology. Further details about the approach can be found in [6-10].

3 Analysis of Past Grid Workflows

User Assistant Agent (UAA) [4] is another component using knowledge assimilated by KAA. Workflows from the past are compared with the current workflow using a Workflow XML Analyzer (KAA-WXA).

If identical workflow is found, the result of such workflows is offered to the current user for potential reuse through UAA interface or user can used this composed and fully filled workflow with data to submit it for computing. KAA-WXA checks an XML database, produced by a Grid Workflow Execution Service (GWES) [5], for new workflows, determines the context from the workflow (using XPath expressions) and sends the results to the UAA in a format of a note. If UAA has the same context and input data, the old note is overwritten by the new one, thus ensuring information update. Workflow XML Analyzer is

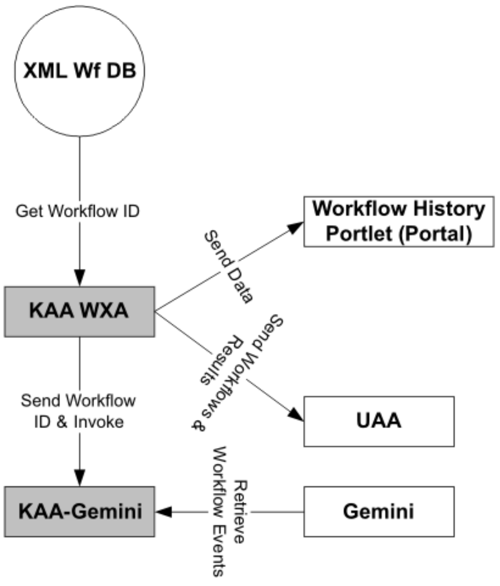


Fig. 3: KAA-WXA Component Architecture.

based on XPath language [16] and all XPath queries are stored in configuration file. KAA-WXA can be reused in all application which are based on XML files parsing. When changes are made in a XML Schema (commonly user must build entire project) only queries in configuration files must be changed. This means that KAA-WXA is a generic tool reusable for parsing XML files. There are other approach like DOM, SAX, Xerces or Castor. DOM and SAX tool have a problem that they must know what is searched, so the tool is not generic or is too robust. Xerces and Castor have a problem with XML Schemas: when a XML Schema is changed entire project must be rebuild. In K-Wf Grid project there are three applications: CTM, ERP, MM5. Every application in K-Wf Grid project has own queries for input and output data like:

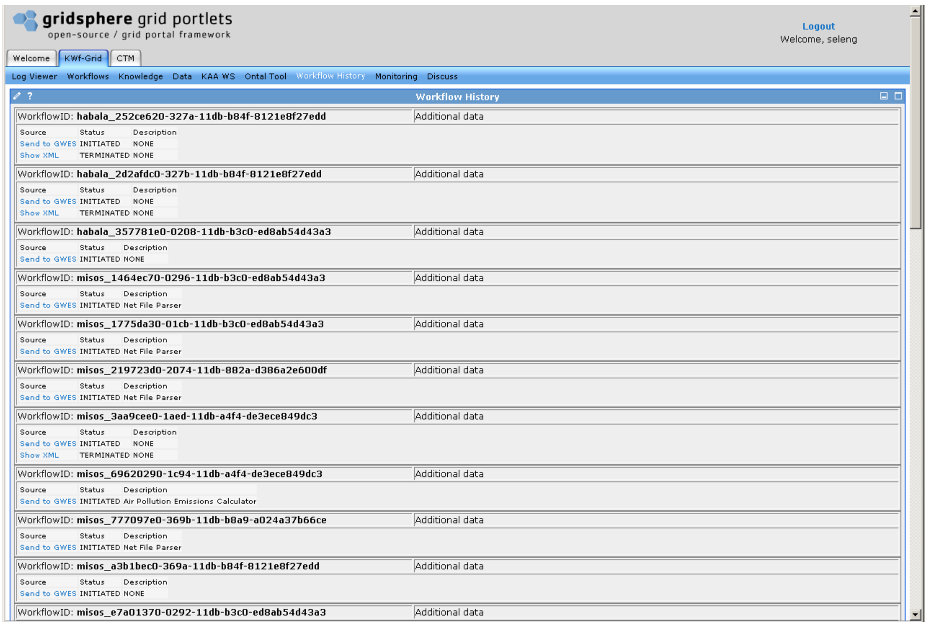
```
STATUS_PATTERN=/*/*[@name='status']/text()
STATUS_TEXT=!
```

The above expressions detect an element with name “status”, whose value will be received by KAA-WXA and will generate a text with this value.

```
START_ZONE_PATTERN=/*/*[@ID='startZoneId']/*/*/*/text()
START_ZONE_TEXT=Start Zone!
```

This query is used in the CTM application and retrieves the ID of the Start Zone.

KAA-WXA responsibility is also to forward input parameters, results and other relevant information to the Workflow History portlet (see Fig. 4).



```
WorkflowID: habala_252ce620-327a-11db-b84f-8121e8f27edd    Additional data
Source        Status     Description
Send to GWES  INITIATED  NONE
Show XML      TERMINATED NONE
WorkflowID: habala_2d2afdc0-327b-11db-b84f-8121e8f27edd    Additional data
Source        Status     Description
Send to GWES  INITIATED  NONE
Show XML      TERMINATED NONE
WorkflowID: habala_357781e0-0208-11db-b3c0-ed8ab54d43a3    Additional data
Source        Status    Description
Send to GWES  INITIATED NONE
WorkflowID: misos_1464ec70-0296-11db-b3c0-ed8ab54d43a3     Additional data
Source        Status    Description
Send to GWES  INITIATED Net File Parser
WorkflowID: misos_1775da30-01cb-11db-b3c0-ed8ab54d43a3     Additional data
Source        Status    Description
Send to GWES  INITIATED Net File Parser
WorkflowID: misos_219723d0-2074-11db-882a-d386a2e600df     Additional data
Source        Status    Description
Send to GWES  INITIATED Net File Parser
WorkflowID: misos_3aa9cee0-1aed-11db-a4f4-de3ece849dc3     Additional data
Source        Status     Description
Send to GWES  INITIATED  NONE
Show XML      TERMINATED NONE
WorkflowID: misos_69620290-1c94-11db-a4f4-de3ece849dc3     Additional data
Source        Status    Description
Send to GWES  INITIATED Air Pollution Emissions Calculator
WorkflowID: misos_777097e0-369b-11db-b8a9-a024a37b66ce     Additional data
Source        Status    Description
Send to GWES  INITIATED Net File Parser
WorkflowID: misos_a3b1bec0-369a-11db-b84f-8121e8f27edd     Additional data
Source        Status    Description
Send to GWES  INITIATED NONE
WorkflowID: misos_e7a01370-0292-11db-b3c0-ed8ab54d43a3     Additional data
```

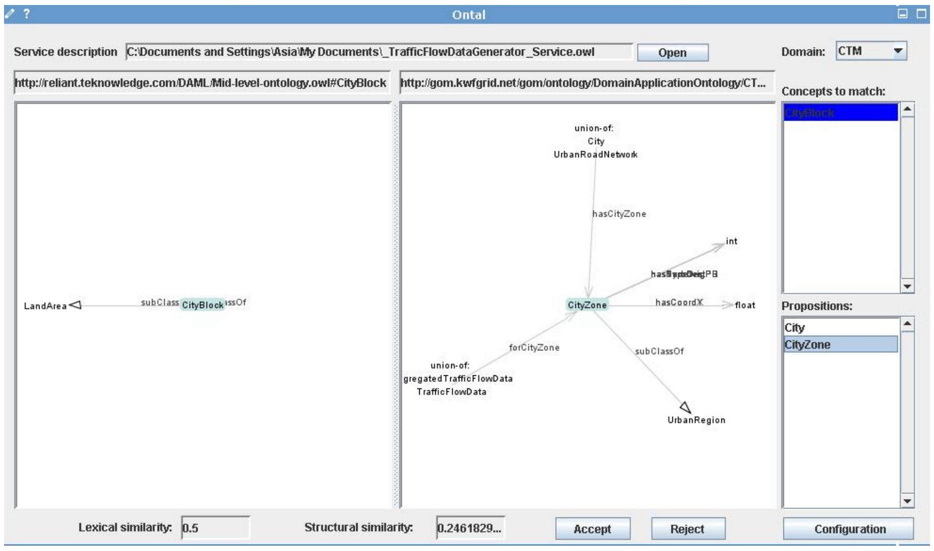
Fig. 4: Snapshot of the Workflow History Portlet embedded in the K-Wf Grid portal.

4 Interactive Semi-Automatic Ontology Alignment

To extend knowledge gathered in the system during the process of new Grid services' registration which is a part of service discovery the OnTal tool has been designed [13, 14]. It processes an OWL-S service description, maps its external ontological concepts onto concepts already existing in the system knowledge base (Grid Organizational Memory) and proposes a set of ontology extensions validated further by a knowledge administrator.

OnTal uses the methods of ontology alignment [11] for mapping of concepts from external ontologies onto corresponding entities in the system knowledge base. In order to assess different aspects of concepts' similarity, both lexical and structural methods are used. The tool tries to find Grid Organizational Memory equivalents (identical or very similar concepts) for all external concepts found in the service description and visualizes pairs of mapped concepts to gather user's feedback. When some external concept doesn't have any equivalent, it is added to one of GOM's ontologies together with its module (extended neighborhood in the original ontology), so that its semantics is preserved. The methods described above use the fact that OWL-S service descriptions are strongly structured and only contain single concepts from external ontologies, which allows taking a local approach to extending the system knowledge base (Grid Organization Memory) with new concepts. Moreover, the characteristics of Grid Organization Memory's ontologies have also been taken into account during the algorithm's design phase.

OnTal tool aims to support a knowledge engineer in his/her work during discovery and registration of new Grid services for the system. The engineer can view visualized concept mapping propositions along with their similarity values and assess these propositions by accepting or rejecting them (see Fig. 5). Consequently, the tool generates description of necessary ontology changes, which can be then submitted to Grid Organization Memory. A more detailed description of the tool's functioning can be found in [14].



Service description: C:\Documents and Settings\Asia\My Documents_TrafficFlowDataGenerator_Service.owl

Domain: CTM

http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl#CityBlock

http://gom.kwfgrid.net/gom/ontology/DomainApplicationOntology/CT...

Concepts to match: CityBlock

Propositions: City, CityZone

Lexical similarity: 0.5

Structural similarity: 0.2461829...

Fig. 5: OnTal User Interface.

5 Conclusion

In the article we have described a set of tools which provide the core functionality of a component called Knowledge Assimilation Agent in scope of the K-Wf Grid project. All the tools described in this article were implemented and are available for use in other service-oriented Grid workflow environments were the reuse of knowledge about services is required. Details about the K-Wf Grid project as well about the described tools can be found at the project website [1].

Acknowledgements. This work is supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, RAPORT APVT-51-024604, VEGA No. 2/6103/6, VEGA 2/7098/27.

References

1. K-WfGrid Project, `http://www.kwfgrid.eu`
2. Lukasz Dutka, Piotr Nowakowski, Jacek Kitowski; Java-Based Component Expert Framework for Automatic Application Builder. In Proceedings of the *Cracow Grid Workshop 2004*, pp.270-274, Academic Computer Centre CYFRONET AGH, ISBN 83-915141-4-5, Poland 2005.
3. Marek Wieczorek, Radu Prodan, Thomas Fahringer; Scheduling of Scientific Workflows in the ASKALON Grid Environment, ACM SIGMOD Record, 09/2005.
4. Laclavik, M., Gatial, E., Balogh, Z., Habala, O., Nguyen, G., Hluchy, L.; Experience Management Based on Text Notes (EMBET); Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunnigham and Miriam Cunnigham; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0.
5. Andres Hoheisel, Thilo Ernst, Uwe Der; A Framework for Loosely Coupled Applications on Grid Environments. In: *Grid Computing: Software Environments and Tools.* Cunha, Jose C.; Rana, Omer F. (Eds.), 2006 ISBN: 1-85233-998-5.
6. Balogh Z., Gatial E., Laclavik M., Maliska M., Hluchy L.; Knowledge-based Runtime Prediction of Stateful Web Services for Optimal Workflow Construction. Proc. of 6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005, R. Wyrzykowski et.al. eds., 2006, LNCS 3911, Springer-Verlag, pp. 599-607, ISSN 0302-9743, ISBN 3-540-34141-2. Poznan, Poland.
7. Balogh Z., Gatial, E., Laclavik, M., Hluchy, L.; Refinement of case retrieval for instance-based Grid service performance prediction through semantic description of input data. In: J. Paralic, J. Dvorsky, M. Kratky (Eds.): Znalosti 2006, Proceedings of 5th annual conference. VŠB-Techická universita Ostrava, Fakulta elektrotechniky a informatiky, 2006, pp.151-158. February 2006, Hradec Kralove, Czech Republic. ISBN 80-248-1001-8.
8. Balogh, Z., Hluchy, L.; Knowledge Assimilation for Performance Prediction of Grid Services for optimal Workflow Execution. In: Proc. of 1-st workshop Grid Computing for Complex Problems – GCCP 2005, VEDA, 2006, pp.135-144, ISBN 80-969202-1-9. November-December 2005, Bratislava, Slovakia.
9. Balogh Z., Gatial E., Laclavik M., Hluchy L.: Refinement of Case Retrieval for Grid Service Performance Prediction through Semantic Description of Input Data. In: Bubak, M., Turala, M., Wiatr, K. (editors): *Proceedings of the Cracow Grid Workshop '05*, Cracow, November 2005, pp. 59-65, published by ACC CYFRONET AGH, Poland, April 2006. ISBN 83-915141-5-3.
10. Balogh, Z., Gatial E., Laclavik, M., Maliska, M., Hluchy, L., Oravec, V., Nguyen, G.; Predicting Web Service Performance based on Web Service States and Input Data Properties. Proc. of 8th Intl. Conf. informatics, SSAKI Bratislava, 2005, pp. 113-117. ISBN 80-969-243-3-8.
11. Zieba, J., Slota, R., Majewska, M., Kryza, B., and Kitowski, J.,: Ontology Alignment and Ontology Similarity for Extension of Service Ontology in Grid Environment. In: Bubak, M., Turala, M., Wiatr, K. (eds.): *Proceedings of the Cracow Grid Workshop '05*, Cracow, November 2005, published by ACC CYFRONET AGH, Poland, April 2006. ISBN 83-915141-5-3.
12. Kryza, B., Slota, R., Majewska, M., Pieczykolan, J., Kitowski, J.; Grid Organizational Memory – An Overview of Semantic Approach to Providing High-Level Grid Abstraction Layer, FGCS, 2006, in print.

13. Maedche, A., Staab, S.; Measuring similarity between ontologies, In: Proc. of the European Conference on Knowledge Acquisition and Management – EKAW-2002, LNCS 2473, Springer-Verlag, pp. 251-263, Madrid, Spain,
14. Slota, R., ZIEBA, J., JANUSZ, M., Kitowski, J.; OnTal - the tool for ontology extension in knowledge-based grid systems, CGW'06.
15. OWL Web Ontology Language Overview, `http://www.w3.org/TR/owl-features/`
16. XML Path Language (XPath) Version 1.0, `http://www.w3.org/TR/xpath`

User Assistant Agent: Towards Collaboration and Knowledge Sharing in Grid Workflow Applications

Michal Laclavík, Martin Šeleng, and Ladislav Hluchý

Institute of Informatics, Slovak Academy of Sciences,
Dubravska cesta 9, 845 07 Bratislava, Slovakia
email: `laclavik.ui@savba.sk`
`http://ikt.ui.sav.sk`

Abstract

This paper focuses on a User Assistant Agent for collaboration and knowledge sharing in Grid Workflow applications. The theory, implementation and use of such system – EMBET are 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 a user is detected from computerized tasks performed by the user. Also intelligent components in the grid middleware such as monitoring, workflow analysis or workflow composition can provide context sensitive notes to be displayed for the user. The User Assistant Agent was created in scope of the K-Wf Grid project, in which grid services are semi-automatically composed to workflows dealing with 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. due to missing specific input data or fulfillment of a user and application specific requirements. To help a user in workflow construction, problem specification or knowledge reuse from past runs, it is appropriate to display notes and suggestions entered by users or intelligent middleware components. Thus experts can collaborate and fill up application specific knowledge base with useful knowledge, which is shown to users in the right time.

1 Introduction

The experience management [1] 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 [2]. 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 a 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 **e**xperience **m**anagement **b**as**e**d on **t**ext notes – EMBET 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 – actions provided in grid environment (portal, submitted jobs, past runs), a step in a business process or a workflow management system, used files or detection of other events performed in the computer. 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 User Assistant solution based on EMBET system[3] is to provide a simple and powerful collaboration and knowledge sharing platform based on experience management infrastructure, which can be applicable in many areas especially in grid virtual organizations. The idea is to return information, knowledge or 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 knowledge support for construction, execution and reuse of grid workflows and results. 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

The EMBET system is built on several theories and dealing with several theoretical challenges:

- Experience Management,
- Context Modeling and Context Detection,
- and Context Matching

[1]`http://www.sadiel.es/Europa/pellucid/`

[2]`http://www.kwfgrid.eu/`

[3]`http://ikt.ui.sav.sk/?page=software{\_}doc/embet.php`

2.1 Experience Management

According to Bergman [1], the experience management is simply the capability to collect lections 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). To provide appropriate lesson learned we need to fulfill the following steps:

1. User context detection from the environment which describes problem P
2. Our Model is described by ontology and Notes are stored with 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. Context matching techniques are applied to find match between knowledge and user context. As a result all applicable notes are matched and returned.
4. The lesson is left to be applied by the user by reading appropriate notes.

More on applying this theory in EMBET can be found in [3], [4].

2.2 Context Modeling and Context Detection

For context and also knowledge modeling we use semantic web approach – ontologies and the CommonKADS [5] methodology. We are able to model and detect context when the application domain is well modeled using ontologies. Our model is an ontology model based on five main elements: Resources, Actions, Actors, Context and Events. Other ontology models can be attached easily to the model where concepts from foreign ontology are specified as context if not specified otherwise. The main idea is to model environment events which provide context of the user. For more details see [6], [7].

In EMBET we need to detect user context from events transformed to the ontology model. For each application we need to specify an appropriate algorithm for user context updating based on user related events [4].

We also need to detect context of information, knowledge or experience entered by a user. In Section 4 we describe a concrete example of such detection. Detected context is only suggested to the user in form of checklist and a user confirms final knowledge /note context. A checklist is created of user current context and context detected from text of notes using automatic semantic annotation techniques of a knowledge note text. This semantic annotation is described in detail in [8], [9].

2.3 Context Matching

The role of EMBET is to assist users in relevant knowledge/suggestions, which are applicable to their current situation. EMBET needs to return experience in context where experience is relevant. Thus we need to match context of knowledge and context of a user to return appropriate knowledge to the user. In

K-Wf Grid we use a simple matching technique where ontology elements present in knowledge context have to be found in user context; otherwise knowledge notes are not displayed. However, this simple matching algorithm is not sufficient in some applications and we need to use technologies based on similarity matching [10], [11]. Another problem concerning voting on knowledge notes occurs, if similarity mechanisms are used since a vote weight depends on a similarity value and need to be determined.

Currently we are developing and testing a new context matching algorithm based on intersection of user and knowledge context.

We define 3 sets of concepts in (1):

$$U = \{u_1,\ u_2,\ ...,\ u_n\}\,, K = \{K_1,\ K_2,\ ...,\ K_m\}\,, W_i = \{k_{i1},\ k_{i2},\ ...,\ k_{il}\} \quad (1)$$

Where U is a set of user context, K is a set of knowledge notes and W_i is a set of knowledge context for knowledge K_i.

A problem occurs in context matching, because there are four different ways to match user context and knowledge context (2).

$$U = W_i, \quad U \subset W_i, \quad U \supset W_i, \quad U \not\subset W_i \wedge W_i \not\subset U \wedge U \cap W_i \neq \emptyset \quad (2)$$

It is clear that the best option is the first one because there is exact match in contexts. The question is what about the other equations? We decide to define the relevance r_i (3), (4), which sorts the detected knowledge from best to worst.

$$r_i = \frac{|U \cap W_i|}{|U \cup W_i|} \quad (3)$$

In (4) we show different approach which can compute different relevance in some cases.

$$r_i = \frac{2\,|U \cap W_i|}{|U| + |W_i|} \quad (4)$$

For all knowledge where $r_i > 0$ we create sorted list where the most relevant knowledge is on the top.

Both approaches (3), (4) give equal relevance in the two cases described in (5) and (6):

$$U = \{u_1,\ u_2\}\ W_i = \{k_{i1},\ k_{i2},\ k_{i3}\}\,,\ u_1 = k_{i2} \quad (5)$$

$$U = \{u_1,\ u_2,\ u_3\}\ W_i = \{k_{i1},\ k_{i2}\}\,,\ u_2 = k_{i1} \quad (6)$$

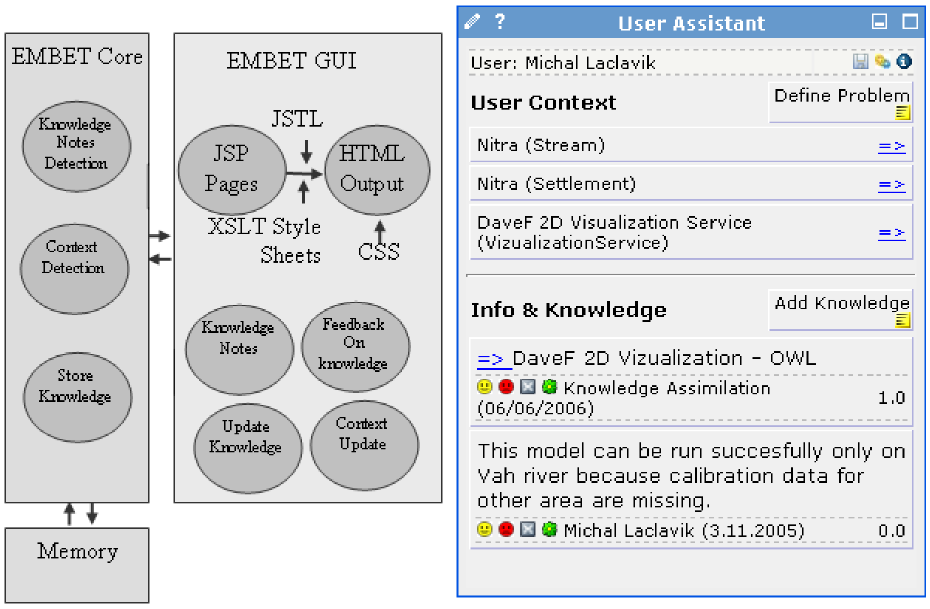
This equality is our future work. In addition, we would like to deal with voting on displayed knowledge notes. We need to calculate different voting weights based on relevance of user and knowledge context.

In our future work we will also evaluate usage of both relevance approaches (3), (4).

3 Architecture and Technology

Architecture of EMBET (Figure 1 left side) consists of 3 main elements:

- Core of the system
- GUI
- System Memory



User: Michal Laclavik

User Context — Define Problem

Nitra (Stream) =>

Nitra (Settlement) =>

DaveF 2D Visualization Service (VizualizationService) =>

Info & Knowledge — Add Knowledge

=> DaveF 2D Vizualization - OWL

Knowledge Assimilation (06/06/2006) 1.0

This model can be run succesfully only on Vah river because calibration data for other area are missing.

Michal Laclavik (3.11.2005) 0.0

Fig. 1: 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 or SOAP to EMBET GUI.

EMBET GUI Graphical User Interface (Figure 1 right side) visualizes 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 visualizes knowledge based on XML[4] transformation to HTML through XSL[5] Templates processing. Moreover EMBET GUI has an infrastructure for a note submission and context visualization. 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.

[4]`http://www.w3.org/XML/`
[5]`http://www.w3.org/TR/xslt`

EMBET Core – EMBET GUI interface is used for an XML data exchange between EMBET Core and EMBET GUI. The Interface is based on the XML-RPC[6] 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. Ontology is stored and managed in the Web Ontology Language (OWL)[7]. The Jena Semantic Web Library[8] 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[9] and implementation middleware is reasonable.

4 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.

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:

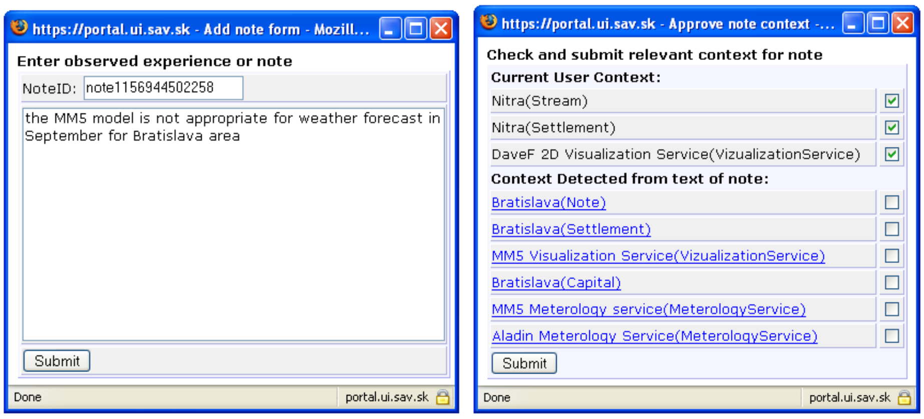
- A user enters a note through UAA, stating that "the MM5 model is not appropriate for weather forecast in September for the Bratislava area" (Fig. 2).
- From the workflow in which the user states this note, we know directly the current user context (checked items on Fig. 2)
- Some of current context 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 found in the ontology memory (GOM) (Fig. 2). In this case, by finding the text MM5 we can assume that "MM5 Meteorology Service" 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.

[6]`http://www.xmlrpc.com/`

[7]`http://www.w3.org/TR/owlfeatures/`

[8]`http://www.sf.net/`

[9]`http://jakarta.apache.org/tomcat/`


https://portal.ui.sav.sk - Add note form - Mozill...

Enter observed experience or note

NoteID: note1156944502258

the MM5 model is not appropriate for weather forecast in September for Bratislava area

Submit
Done
portal.ui.sav.sk
https://portal.ui.sav.sk - Approve note context -...

Check and submit relevant context for note

Current User Context:

Nitra(Stream)	✓
Nitra(Settlement)	✓
DaveF 2D Visualization Service(VizualizationService)	✓

Context Detected from text of note:

Bratislava(Note)	☐
Bratislava(Settlement)	☐
MM5 Visualization Service(VizualizationService)	☐
Bratislava(Capital)	☐
MM5 Meterology service(MeterologyService)	☐
Aladin Meterology Service(MeterologyService)	☐

Fig. 2: Left: Entering new Note; Right: Note Context Detection, checked items are current user context, unchecked items are elements detected from text of the note. User selects only those items which are relevant.

- After the context detection, the user is provided with a checklist (Fig. 2) where the user may select only the relevant parts of the note context.
- A user selects parts of the context, which were detected by the system as really relevant. He/she can modify the contents of the list and finally submit the note.
- Each time anyone works with the MM5 service for the Bratislava area in September, the note is displayed or it can be also displayed in similar contexts
- Each note can be evaluated 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.

5 Conclusion and Future Work

Our solution was evaluated on the K-Wf Grid IST project, focused on building grid-based workflows, where users need to collaborate and share knowledge about different applications, computational models, grid resources or data. Previously it was evaluated on a selected administration application in the Pellucid IST project, where the context or the problem of a user was detected in the Workflow Management Application. Such solution may be applied in many further areas where the user problem can be detected from computerized tasks. Usually this occurs 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.

In K-Wf Grid we help users in workflow construction and execution by displaying knowledge 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. In addition, intelligent components create and provide knowledge notes with attached context to the EMBET systems concerning past workflows runs or computed results. Thus users can reuse old workflows or results of execution, which are displayed in proper user context by EMBET system.

Acknowledgements. This work is partially supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, RAPORT APVT-51-024604, SEMCO-WS APVV-0391-06, VEGA No. 2/6103/6, VEGA 2/7098/27.

References

1. Bergmann, R.; Experience Management: Foundations, Development Methodology, and Internet-Based Applications (Lecture Notes in Artificial Intelligence), 2002
2. Davenport, T.H., Prusak, L.; Working Knowledge, ISBN 1578513014, 2000
3. Laclavik, M., Gatial, E., Balogh, Z., Habala, O., Nguyen, G., Hluchy, L.; Experience Management Based on Text Notes (EMBET); Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunnigham and Miriam Cunnigham; IOS Press, pp. 261-268. ISSN 1574-1230, ISBN 1-58603-563-0.
4. Michal Laclavik; Experience Management based on Ontology and Text Notes (The EMBET System); Work submitted for the RNDr. degree; Ustav Informatiky, Prirodovedecka fakulta, Univerzita Pavla Jozefa Safarika v Kosiciach, 2005
5. August Schreiber et al., Knowledge Engineering and Management: the CommonKADS methodology, ISBN 0262193000, 2000
6. Laclavik, M., Babik, M., Balogh, Z., Hluchy, L.; AgentOWL: Semantic Knowledge Model and Agent Architecture; In Computing and Informatics. Vol. 25, no. 5 (2006), pp. 419-437. ISSN 1335-9150
7. Michal Laclavik, Marian Babik, Zoltan Balogh, Emil Gatial, Ladislav Hluchy; Semantic Knowledge Model and Architecture for Agents in Discrete Environments; In: Frontiers in Artificial Intelligence and Applications, Vol 141, 2006. Proc. of ECAI 2006 Conference, G.Brewka et al.(Eds.), IOS Press, pp. 727-728. ISBN 1-58603-642-4 ISSN 0922-6389 2006
8. Laclavik, M., Seleng, M., Gatial, E., Balogh, Z., Hluchy, L.; Ontology based Text Annotation – OnTeA; Information Modelling and Knowledge Bases XVIII. IOS Press, Amsterdam, Marie Duzi, Hannu Jaakkola, Hannu Kangassalo, Yasushi Kiyoki (Eds.), 2007
9. Michal Laclavik, Martin Seleng, Marian Babik; OnTeA: Semi-automatic Ontology based Text Annotation Method; In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. P. Navrat et al. (Eds.), Vydavatelstvo STU, Bratislava, 2006, pp. 49-63, ISBN 80-227-2468-8.

10. Michal Laclavik; Ontology and Agent based Approach for Knowledge Management; PhD Thesis submitted for the degree philosophiae doctor; Institute of Informatics, Slovak Academy of Sciences, field: Applied Informatics, 2005
11. Balogh Z., Budinska I.; OntoSim – Ontology-based Similarity Determination of Concepts and Instances. In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. P. Navrat et al. (Eds.), Bratislava, 2006, pp. 64-70, ISBN 80-227-2468-8.
12. Hluchy L., et al.; Flood Forecasting in CrossGrid project. In: Grid Computing, 2nd European Across Grids Conference, Nicosia, Cyprus, Jan 28-30, 2004, LNCS 3165, Springer-Verlag, 2004, pp. 51-60, ISSN 0302-9743, ISBN 3-540-22888-8

K-Wf Grid Portal: a Web Interface to Semantic Workflows

Emil Gatial[1], Branislav Simo[1], Giang Nguyen[1], Michal Laclavik[1], Ladislav Hluchý[1], Tilman Linden[2], and Bartosz Kryza[3]

[1] Institute of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia
[2] Fraunhofer FIRST, Berlin, Germany
[3] Academic Computer Centre CYFRONET AGH, Krakow, Poland
email: `emil.gatial@savba.sk`

Abstract
The K-Wf Grid portal provides a central user interface, point of interaction and access to the components developed within the K-Wf Grid project. The main functionality of K-Wf Grid portal is to provide users with access to the components developed within the K-Wf Grid project and other requested resources. This covers the components for workflow specification, workflow management and visualization, components dealing with the metadata management and components for accessing the other Grid services. Moreover K-Wf Grid portal includes collaborative feature provided by lightweight discussion portlet. Architecture of K-Wf Grid portal is based on the Gridsphere portal framework that provides an open-source portlet based Web portal. GridSphere enables developers to quickly develop and package third-party portlet web applications that can be run and administered within the GridSphere portlet container. The basics features of Gridsphere include management of user accounts, groups and access to specific portlets, providing customized layout. Portlets provide "mini application" that can either display informational content or provide access to other components. The paper describes K-Wf Grid portal architecture along with the user interfaces of the key components developed within the K-Wf Grid project.

Keywords: K-Wf Grid, Workflow, Gridsphere, ontology, portlet, Java applet, AJAX.

1 Introduction

K-Wf Grid application portal provides convenient web based user interface for the tools and applications being developed in the context of the K-Wf Grid project [1] (simplified component structure is shown on the Fig. 1). The Users with valid portal account can access various portal sections focused on K-Wf Grid's applications and associated knowledge, application monitoring and grid services.

The portal is built on the Gridsphere portal framework [2]. The framework is an implementation of a portlet paradigm defined in JSR-168 Portlet API

Specification – portlets are self-contained components generating just part of the portal page, independent from each other that are pluggable into the portal. The Gridsphere framework provides the environment for running the portlets, portlet state management, session management and various other tasks. The gist of the K-Wf Grid portal development lies in the development of the portlets providing user interfaces to various K-Wf Grid software components.

Portlets can be assimilated into the portal web pages with customized layout. KWf Grid portal includes many different components (portlets) that access the components of K-Wf Grid.

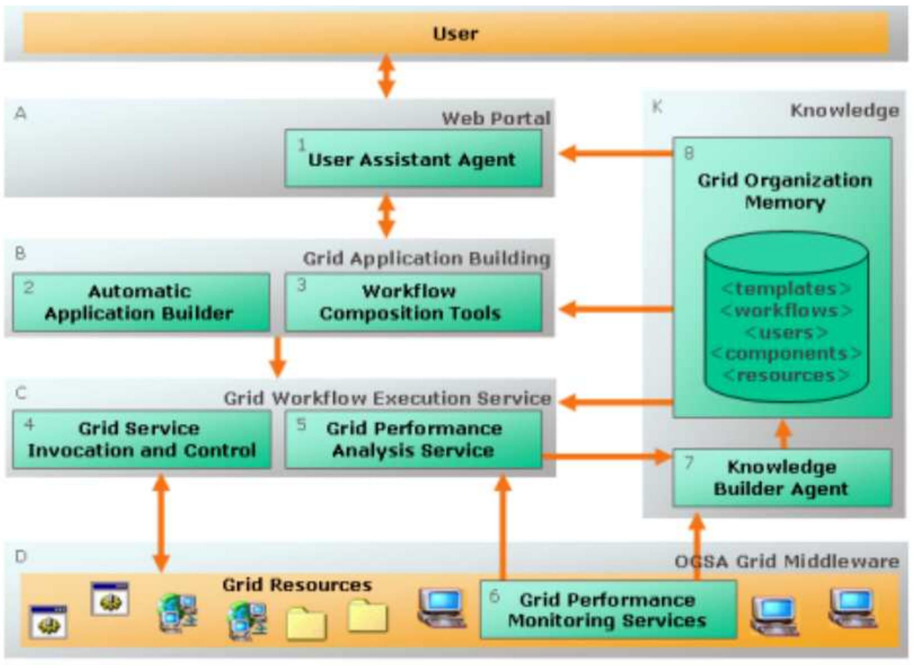

User
A
Web Portal
1 User Assistant Agent
K
Knowledge
8 Grid Organization Memory
<templates>
<workflows>
<users>
<components>
<resources>
B
Grid Application Building
2 Automatic Application Builder
3 Workflow Composition Tools
C
Grid Workflow Execution Service
4 Grid Service Invocation and Control
5 Grid Performance Analysis Service
7 Knowledge Builder Agent
D
OGSA Grid Middleware
Grid Resources
6 Grid Performance Monitoring Services

Fig. 1: Basic software architecture of K-Wf Grid project.

2 Used Technologies

K-Wf Grid portal uses various technologies to produce application logic via the user interface by connecting to background components. Most of the user interfaces are wrapped within the portlet space, which dimensions are tailored to application needs. The portlets wrapping other components show a very easy way how to integrate components developed by diverse project partners. The following list enumerates used wrapped technologies:

- Server side technologies like JSR-168 portlet, JSP, XSLT or GridSphere UI tags – implemented components must be periodically processed by server and refreshed after each action.
- Client side technologies like JavaScript and Java applets – usually loads to client browser and runs independently of the portal. While Java applets may use plenty of communication ways (using XML-RPC, web services, JMS, etc), the JavaScript client applications have the only viable way how to communicate with server is using AJAX methodology.

3 Portal Structure

The layout of K-Wf Grid portal components has structure as depicted of the schema (Fig. 2). The layout is separated into 3 main panels:

- Application panel – comprises the components regarding the workflow and user supporting applications.
- Knowledge panel – encompasses the components that processing the data and information stored in the GOM (Grid Organizational Memory). Such data are relevant during the semantic workflow composition. The most relevant example of such component is a KAA (Knowledge Assimilation Agent) [3],
- Performance monitoring and Analysis panel – contains components that access and present monitoring data.

Every panel is further divided into several panels where the particular components are located.

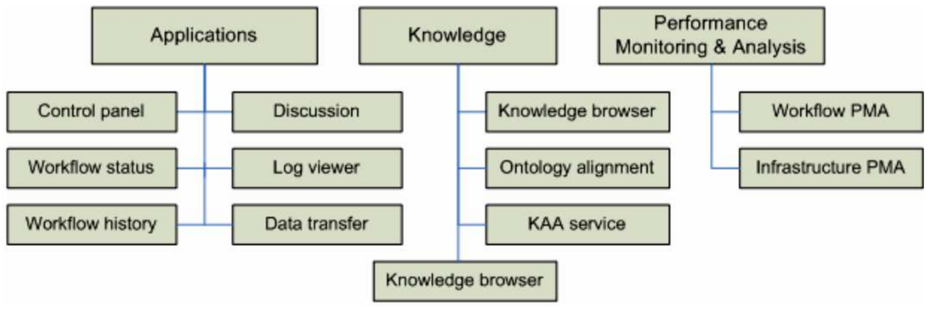


Fig. 2: K-Wf Grid portal layout structure.

4 User Interface in More Detail

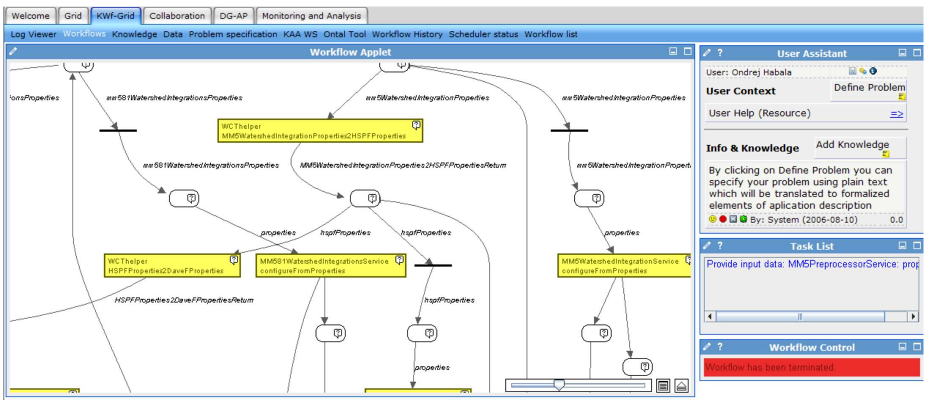
From the technological point of view there are several typical interesting components characterized by using of specific combination of technologies. Since full description of developed portal components can lead to stodgy enumeration of components, the following paragraphs will describe only some typical representatives with brief technology description.

4.1 Control Panel

The most relevant portlets reside on the Control panel, such as the Workflow portlet which provides cute interactive interface to GWES (Grid Workflow Execution Service) component [4] via included GWUI applet. This applet is tightly coupled with other two visual components Task list and Workflow control that are implemented as applets wrapped within the small portlets. This portlet allows users to manage a semantic composed workflow by the well arranged workflow graph.

Next component residing the Control panel is a User Assistant (UA) portlet which provides access to the User Assistant Agent (UAA) component [5]. This portlet allows user to specify a problem and view the notes relevant to such problem. UAA component can advise the user on already composed workflows for similar problem.

Originally the interface is implemented as a servlet along with the use of XSLT technology and it is wrapped by the UA portlet. Moreover, UA portlet implements dynamic refreshment mechanism using AJAX technology. The great advantage of using the AJAX methodology comes with sending an asynchronous requests without continuously reload the page which can otherwise look disturbing. Refresh mechanism works similarly as conventional 'iframe' refreshment [6] except that the client browser periodically checking the incoming value instead of executing the JavaScript fragments within the 'iframe' tag.


Welcome
Grid
KWf-Grid
Collaboration
DG-AP
Monitoring and Analysis
Log Viewer
Workflows
Knowledge
Data
Problem specification
KAA WS
Ontal Tool
Workflow History
Scheduler status
Workflow list
Workflow Applet
mm581WatershedIntegrationsProperties
mm5WatershedIntegrationProperties
WCThelper MM5WatershedIntegrationProperties2HSPFProperties
MM5WatershedIntegrationProperties2HSPFPropertiesReturn
properties
hspfProperties
WCThelper HSPFProperties2DaveFProperties
MM581WatershedIntegrationsService configureFromProperties
MM5WatershedIntegrationService configureFromProperties
HSPFProperties2DaveFPropertiesReturn
User Assistant
User Context
Define Problem
User Help (Resource)
Info & Knowledge
Add Knowledge
Task List
Workflow Control
User: Ondrej Habala

By clicking on Define Problem you can specify your problem using plain text which will be translated to formalized elements of aplication description

By: System (2006-08-10) 0.0

Provide input data: MM5PreprocessorService: prop

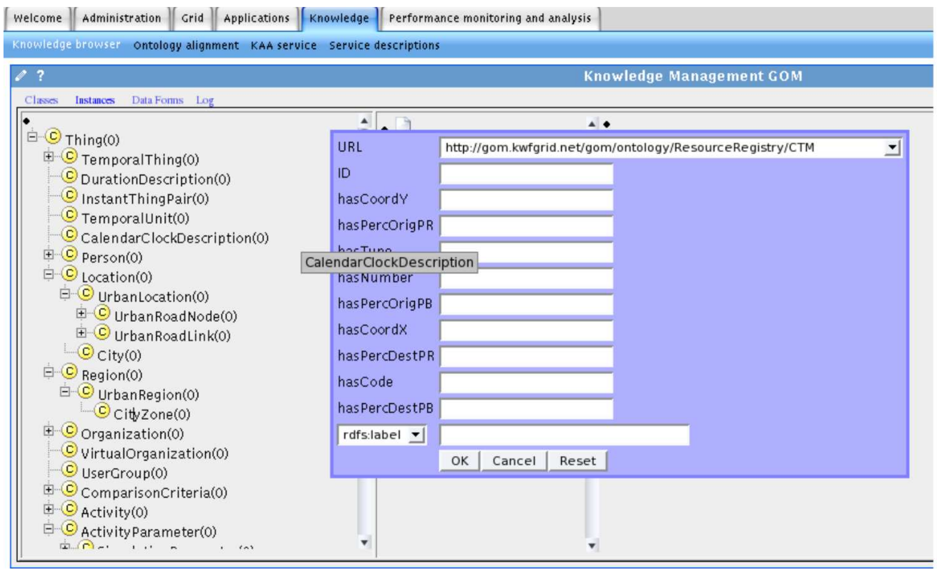
Workflow has been terminated.

Fig. 3: Main panel of K-Wf Grid portal sketching the Workflow applet, User assistant, Task list and Workflow control component.

Picture (Fig. 3) shows layout of the user interface components on the most important portal panel – Control panel.

4.2 Knowledge Browser Panel

GOM Browser portlet employs services implemented in GOM [7] and allows user to browse and manipulate the specified ontology stored within (Fig. 4). While the portlet manages access to the GOM resources the visualization of ontology is mostly based on the JavaScript (XML parser and AJAX) that moves the processing on the client browser. This speeds up the process of visualization and makes the notion of seamless application.



URL http://gom.kwfgrid.net/gom/ontology/ResourceRegistry/CTM

rdfs:label

Fig. 4: The screenshot of knowledge browser panel.

The GOM browser portlet provides following capabilities:

- Ontology browsing – such feature is the core of this component, but also the most complicated. The algorithms for recursive backward ontology processing, OWL tags parsing, dynamic DOM object creation, event handling must be implemented and well coordinated in order to provide true client-side application based purely on technologies such XML, JavaScript and AJAX.
- Update mechanism – periodically sends request for ontology updates. It is implemented like that in UA portlet (see Section 4.1). The only difference is that the page is not reloaded (in case of UA portlet), but the update response contains OWL data about updated fragments of ontology.

- Ontology modifications – features includes ontology fragment (classes or instances) addition and removal. Such features are implemented on the client side as sending the RDF fragments describing the type of action (modification, addition or removal) and modified resources. On the server side, particular service is called according to the recognized action with specific resources. The changes are almost immediately visible via described update mechanism.

4.3 Data Browser Panel

Data Browser portlet wraps servlet for accessing Grid services such as GridFTP transfer, RLS service. Using this portlet user can download or upload files from/to specified storage element. Moreover, the Data Browser can be used as a stand alone servlet accessed directly from URL request.

The user interface is dynamically created on the client-side like in the case of GOM browser. The difference is that the dynamic directory structure is made of data provided via a GridFTP service provided on the server side.

4.4 Log Viewer Panel

Logging portlet shows logging messages produced by the project components, thus making it possible to monitor inner workings of those components. It is mainly meant for component developers as a convenient way of monitoring of their software and the cooperation of all the projects components. The portlet is made of the portlet, JSP and UI tags technology.

4.5 Discussion Panel

Discussion portlet is lightweight collaborative portlet. User can create new topics, answer to existing topics and view topic's threads or topics only. This porltet is integrated with UAA to annotate text of topics. The portlet is implemented as a portlet using UI tags accessing the database via persistent services.

5 Conclusion

The main innovation of the portal is providing a uniform access to the components developed within the K-Wf Grid project with using the technologies such as AJAX for seamless communication between client and server. The interfaces are wrapped by the JSR-168 compliant portlet and deployed within the Gridsphere portal framework. Moreover, the development and testing of portal components had exposed some interesting comparisons of used technologies. For example, the user interfaces made by AJAX methodology demonstrated a viable alternative to Java applets. Both technologies provide highly interactive client-side applications, but the components using AJAX produce code that is considerably smaller (and therefore are downloaded much faster to a client browser) than components using Java applets, but on the other hand JavaScript applications are more difficult to debug.

Acknowledgements. This work is supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, RAPORT APVT-51-024604, VEGA No. 2/6103/6.

References

1. K-WfGrid Project – `http://www.kwfgrid.eu`
2. Gridsphere web page – `http://www.gridsphere.org`
3. Balogh Z., Hluchy L.: Knowledge Assimilation for Performance Prediction of Grid Services for optimal Workflow Execution. In: Proc. of 1-st workshop Grid Computing for Complex Problems – GCCP 2005, VEDA, 2006, pp.135-144, ISBN 80-969202-1-9. November-December 2005, Bratislava, Slovakia.
4. Hoheisel, A.: User Tools and Languages for Graph-based Grid Workflows. In: Special Issue of Concurrency and Computation: Practice and Experience, Wiley, 2005.
5. Laclavik M., Gatial E., Balogh Z., Habala O., Nguyen G., Hluchy L.: Experience Management Based on Text Notes (EMBET). eChallenges 2005, In: Proc. of eChallenges 2005 Conference, 19-21 October 2005, Ljubljana, Slovenia, Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunnigham and Miriam Cunnigham; IOS Press, pp.261-268. ISSN 1574-1230, ISBN 1-58603-563-0.
6. AJAX Patterns – `http://ajaxpatterns.org/Unique_URLs`.
7. Bartosz Kryza, Jan Pieczykolan, Marta Majewska, Renata Slota, Marian Babik, Adrian Toth, Jacek Kitowski, Ladislav Hluchy. Grid Organizational Memory – Semantic Framework for Metadata Management in the Grid. Accepted for Cracow Grid Workshop'06, October 15-18, 2006, Cracow, Poland.

Service-Based Flood Forecasting Simulation Cascade in K-Wf Grid

Ondrej Habala, Martin Mališka, and Ladislav Hluchý

Institute of Informatics, Slovak Academy of Sciences,
Bratislava, Slovakia
email: `Ondrej.Habala@savba.sk`

Abstract
Flood Forecasting Simulation Cascade is a flood prediction application, using series of meteorological, hydrological, and hydraulic simulation models to predict possible flood risks. This application has been developed for several years, and in the K-Wf Grid project has been implemented as a set of grid services, using the WSRF standard. In this article we describe its architecture, implementation details, usage scenarios, user interfaces, and our experiences in using WSRF standard and K-Wf Grid middleware for flood prediction.

1 Introduction

Prediction of natural disasters is an important part of every public early warning and risk assessment system. This applies also in the area of flood prediction. The process of flood prediction is of complex nature, since it requires knowledge of future weather conditions, which is a complex requirement in itself. The Flood Forecasting Simulation Cascade (FFSC) application of the K-Wf Grid [10] project is a system, which attempts such complex task, using a set of several simulations, performed in stages – weather prediction first, followed by watershed integration, hydrological prediction for the target river, and finally hydraulic simulation for the target area. The application also contains several visualization modules, which can be used to display results from all stages in several ways – for example, the final water flow simulation can be displayed as a series of 2D images, or as a 3D animation, suitable also for 3D display systems, like the CAVE environment. Each stage of the application can be performed by multiple simulation models, depending on the characteristics of the target area and on user preferences.

The second chapter describes the history of the flood prediction application. The third chapter provides a short introduction to the K-Wf Grid concept, and the rest of tha paper deals with the flood prediction application, and its operation using the K-Wf Grid middleware.

2 History of Flood Forecasting Simulation Cascade

The FFSC application has been developed since 2002, when the IST project ANFAS (Data Fusion for Flood Analysis and Decision Support) commenced.

This project tried to solve the final part of the flood simulation – the water flow in the target basin. The hydraulic simulation model FES-WMS [1] has been used; it later evolved into the now-used DaveF model [2]. As a testing area was selected part of the river Vah, and topographical data for this area has been obtained.

The Flood forecasting application has evolved during the CROSSGRID project [5] from single model application to the large cascade of models integrated by support tools. In the ANFAS project the Flood forecasting application was based on water hydraulics simulation model. But it was insufficient for real time forecasting and it has created requirement for involving the simulation of meteorological a hydrological processes as precursors to the flood forecasting process. Such integrated process is quite complex and requires a lot of computational power and sophisticated software tools. And this was task for the CROSSGRID project.

The cascade takes inputs from our national provider of meteorological analysis data – Slovak hydro-meteorological institute SHMU. This data cover central Europe and are processed in the cascade by the meteorological models (Aladin, MM5) to obtain higher resolution data for target area (which is noticeably smaller in contrast to central Europe). Outputs of the meteorological models are used in two ways: they are visualized in order to allow the user to interact the cascade execution (the user can decide to stop some part of cascade) and they are inputs for the next step in the cascade – the hydrological processing. This process is organized in a virtual organization [3].

After CROSSGRID has finished, the application has been further developed in the MEDIGRID IST project [7]. While this project targeted mainly cross-platform usage of grid systems, it has also contributed to the development of a SOA-based [4] flood forecasting cascade. The services are based on the WSRF standard implementation [8] from the Globus toolkit version 4 (GT4) [9]. It is a Java implementation of core web (grid) services with security based on X.509 certificates, notifications and other features. Each of the system components – simulation models, data providers, information services or other supporting services – are exposed as a web service. This schema is also used in the K-Wf Grid project, where it is further expanded with workflow management and knowledge management tools.

3 Knowledge-Based Grid Computing in K-Wf Grid

The idea of the project K-Wf Grid is based in the observation, that users often have to learn not only how to use the grid, but also how to best take advantage of its components, how to avoid problems caused by faulty middleware, application modules and the inherent dynamic behavior of the grid infrastructure as a whole. Additionally, with the coming era of resources virtualized as web and grid services, dynamic virtual organizations and widespread resource sharing, the variables that are to be taken into account are increasing in number. Therefore we tried to devise a user layer above the infrastructure, that would be able to handle as much of the learning and remembering as possible. This layer should be able to observe what happens during application execution, infer new knowledge from these observations and use this knowledge the next time an application is executed. This way the system would – over time – optimize its behavior and use of available resources.

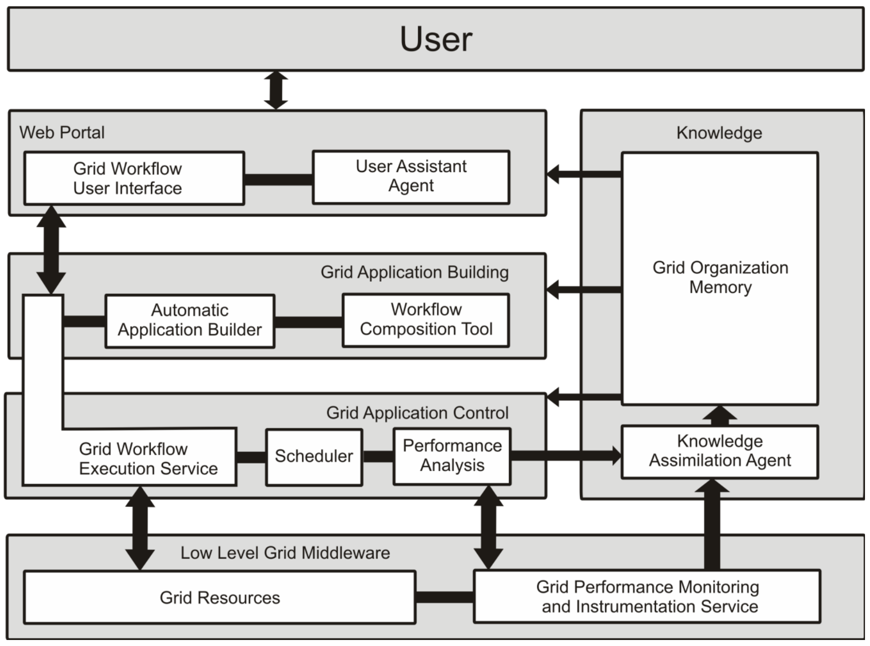

User
Web Portal
Grid Workflow User Interface
User Assistant Agent
Knowledge
Grid Application Building
Automatic Application Builder
Workflow Composition Tool
Grid Organization Memory
Grid Application Control
Grid Workflow Execution Service
Scheduler
Performance Analysis
Knowledge Assimilation Agent
Low Level Grid Middleware
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 1: K-Wf Grid architecture.

The main interaction of users with the system occurs through the Web Portal (see architecture in Fig. 1). Through it, users can access the grid, its data and services, obtain information stored in the knowledge management system, add new facts to it, construct and execute workflows. The portal consists of three main parts, the Grid Workflow User Interface (GWUI), the User Assistant Agent (UAA) interface, and the portal framework based on GridSphere [6], including collaboration tools and interfaces to other K Wf Grid modules. GWUI is a Java applet visualization of a Petri net-modeled workflow of services, in which the user can construct a workflow, execute it and monitor it. UAA is an advisor, which communicates to the user all important facts about his/her current context – the services he/she considers to use, the data he/she has or needs. Apart from automatically generated data, the displayed information contains also hints entered by other users, which may help anyone to select better data or services or avoid problems of certain workflow configurations. This way the users may collaborate together and share knowledge.

Under the Web Portal lies the Workflow Orchestration and Execution module, composed of several components. These components together are able to read a definition of an abstract workflow, expand this definition into a regular workflow of calls to service interfaces, map these calls to real service instances and execute this workflow to obtain the expected results, described in the original abstract workflow. This way the user does not need to know all the services that are present in the grid and he/she is required only to state what result is required.

To be able to abstract the grid is such a way as described in previous paragraph, the system has to know the semantics of the grid environment it operates on, and so we need to employ serious knowledge management, computer-based learning and reasoning. This is the area of the Knowledge module, which is split into the storage part – Grid Organization Memory (GOM), and the learning part – Knowledge Assimilation Agent (KAA). KAA takes observed events from the monitoring system, maps them to the context of the performed operation and extract new facts from them. These facts are then stored into GOM, as well as used in later workflow composition tasks in order to predict service performance. GOM itself stores all information about the available application services in a layered ontology and new applications may be easily added into its structure by describing their respective domains in ontology, connected to the general ontology layer developed in K-Wf Grid.

The monitoring infrastructure is integrated into the original grid middleware, with the Grid Performance Monitoring and Instrumentation Service (GPMIS) as a processing core. GPMIS receives information from a network of sensors, embedded into the middleware, application services (where it is possible to instrument the services) and into the other K-Wf Grid modules. Apart from collecting observations for the learning modules, the monitoring infrastructure is also a comprehensive tool for performance monitoring and tuning, with comfortable visual tools in the user portal.

At the bottom lies the grid itself – the application services, data storage nodes and communication lines. K-Wf Grid has three distinct and varied pilot applications, which it uses to test the developed modules.

4 Knowledge-Based Flood Prediction Application

The new architecture of what was previously called Simulation Cascade [11] has been considerably extended (Fig. 2). It is a set of loosely coupled services, with several possible execution scenarios.

Fig. 2 contains several entities, each of them having its role in our application. At the top of the figure is depicted our main data provider, the Slovak Hydrometeorological Institute (SHMI). SHMI provides us with input data for the first stage of our application, the Meteorology. In this stage, we employ two models ALADIN and MM5 for weather predictions, with the latter having three distinct operation modes (simple, one-way nested and two-way nested). The predicted weather conditions are used in the Watershed integration stage to

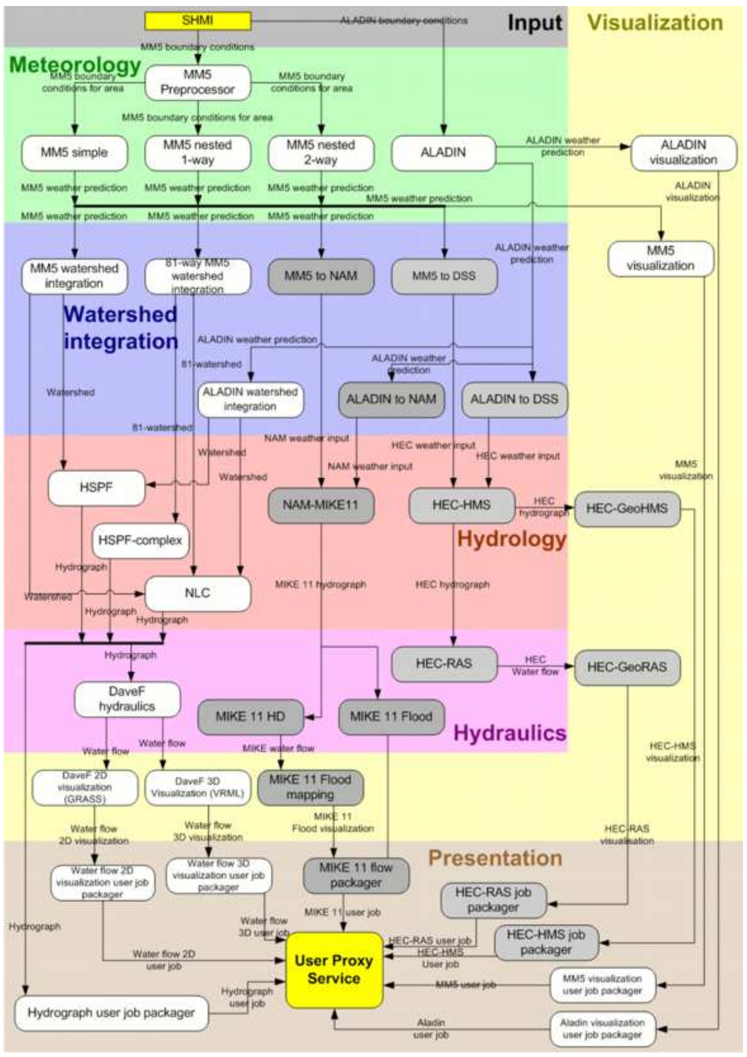


Fig. 2: Service-based flood prediction application.

compute water runoff into the target river. This result is then further processed in the Hydrology stage, where two models – HSPF and NLC compute river levels for selected geographical points. These levels are then used to model water flow in the last, Hydraulic stage of the application. Concurrently all important results are optionally visualized, packaged and displayed to the user – if he/she requires it.

Apart from the simulation models, preprocessor and associated tools, the data flow contains also several job packagers and a User Proxy Service. These services implement our approach towards interactive grid jobs and also toward in-process user collaboration. The user proxy service may receive a ZIP file (prepared by a visualization service) with a HTML sub-tree, which is displayed to a certain user.

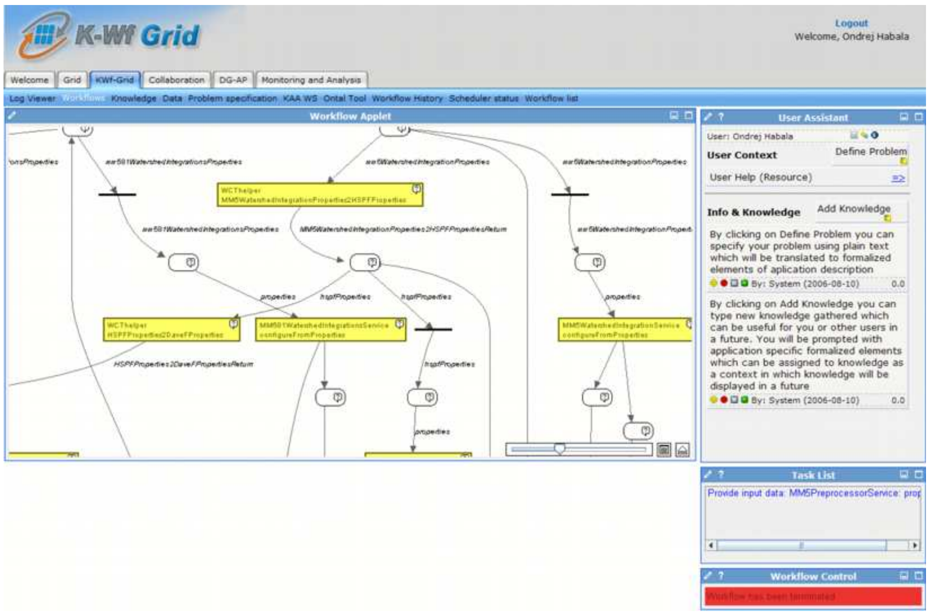
The user is then notified of his/her new task (currently by e-mail, later also ICQ notifications will be implemented) and may view the HTML sub-tree, included images, animations, tables, etc. and may even react and provide input by filling out HTML forms (which are further processed by following services in the application workflow). For each specialized task another user may be asked to provide input, thus collaborating on a bigger job, requiring expertise of several users. Apart from enabling seamless multi-user collaboration on a single workflow, the concept of the User Proxy Service enables (with its asynchronous notifications) users to leave long-running workflows unattended and return to them on request, either to view the computed results or to provide some input.

5 Example Application Scenario

When a user logs into the system, he/she starts by using the User Assistant Agent interfaces (see Fig. 3). He/she enters a text description of the problem to be computed. This description is then analyzed for known keywords and detected elements are presented to the user to confirm detected context of the problem. Free text problem definition is important, when a user starts to work with the system and wants to define the problem in a way understandable for automatic service composition.

When some elements of the problem context are confirmed by the user, they become the semantic representation of user's problem and composition of the services can start. After the problem context and semantic description is stated, the system creates a so-called abstract workflow, consisting only of unknown input, one high-level, abstract task (transition) and the defined output (the solution). This abstract workflow is then analyzed, relevant services capable of creating the solution are found, and a more concrete workflow is created, in which the abstract transition is replaced by several classes of services. These are not yet actual service instances and calls, but rather representations of certain capabilities, which are known to be found in one or more real service instances (see the yellow boxes in Fig. 3).

At this point, the system is ready to start execution of the workflow. When the user clicks the play button in the workflow visualization (lower left part of Figure 6), the system starts looking for real service instances, which are able to perform the tasks represented by the class transitions. These service instances are evaluated by the KAA based on their previous monitored behavior, and the instance believed to perform best (according to a selected metric, for example speed) is then executed. If the system is unable to find service instance for the class transition, user's attention is required. Also, the system is able to recover from a fault of the selected service instance, and to use another instance, possibly working one.



By clicking on Define Problem you can specify your problem using plain text which will be translated to formalized elements of aplication description

By: System (2006-08-10) 0.0

By clicking on Add Knowledge you can type new knowledge gathered which can be useful for you or other users in a future. You will be prompted with application specific formalized elements which can be assigned to knowledge as a context in which knowledge will be displayed in a future

By: System (2006-08-10) 0.0

Provide input data: MM5PreprocessorService: prop

Workflow has been terminated

Fig. 3: View of the K-Wf Grid portal with a flood prediction workflow initialized.

The User Assistant pane has another important role in the workflow execution process. To aid user in service selection, input data provision and general orientation in the world of web and grid services, it provides description of the workflow elements, if such description (note) has been entered previously by other user. This is yet another form of experience management, this time based on the text notes passed between users. These text notes are entered in a form of a line (or several lines) of text through a button in the User Assistant pane. After the note is entered, the context of the currently selected element of the workflow is analyzed, verified by the user and the note is bound to this semantic context. Then the User Assistant is able to find the note later, if a similar context of a workflow element is present in the workflow, and the note may be displayed and may possibly guide the user with previous user's experience. These notes may be used to describe certain special qualities (or deficiencies) of some service classes or instances, such as ability/inability to work under certain conditions or to provide quality results for some tasks.

6 Conclusion

The flood prediction application, and also by it supported virtual organization, have undergone extensive development in recent 2 years during the K-Wf Grid project. It has evolved from a single simulation module to a sophistically controlled set of services, based on state-of-the-art community standards. Also the possibilities for VO support have expanded, with collaboration tools and experience management now present in the support middleware, and all knowledge described and stored in an ontology-based repository. Now its users don't have to know anything about grid, except very basic essential principles of grid-wide data management and security. All control is done via graphical interface with domain-specific names of components; data input and results visualization is also customizable – usability of the system has been significantly increased.

The application is further evolving. New simulation models are added to it; new tools for data visualization and data representation are also under development. In the future, we hope to integrate the application with other geographical and meteorological tools and standards, which are widely adopted and appreciated in the hydrometeorological community.

Acknowledgements. This work is supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, VEGA No. 2/6103/6, VEGA 2/7098/27.

References

1. FES-WMS: Finite Element Surface Water Modeling System. U.S.G.S. `http://smig.usgs.gov/cgi-bin/SMIC/model_home_pages/model_home?selection=feswms`
2. Tran, V., Hluchy, L., Froehlich, D., Castaings, D.: Parallelizing Flood Model for Linux Clusters with MPI. LNCS vol. 3019, 2004, Springer Verlag, Berlin, pp. 521-527. ISBN 978-3-540-21946-0.
3. Hluchy, L., Tran, V.D., Habala, O., Simo, B., Gatial, E., Astalos, J., Dobrucky, M.: Flood Forecasting in CrossGrid Project. LNCV vol. 3165, 2004, Springer verlag, Berlin, pp. 51-60. I SBN 978-3-540-22888-2.
4. Web Services and Service-oriented Architectures. `http://www.service-oriented.com` (accessed December 2006).
5. CrossGrid – Development of Grid Environment for Interactive Applications, `http://www.crossgrid.org` (Accessed Sept. 2006)
6. Novotny, J., Russel, M., Wehrens, O.: GridSphere: a portal framework for building collaborations. In: Concurrency and Computation: Practice and Experience, Volume 16, Issue 5, 2004, pp. 503-513
7. The Medigrid project home page. `http://www.eu-medin.org/projectPage.php?acronym=MEDIGRID` (visited September, 2006)
8. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The Physiology of the Grid. `http://www.globus.org/alliance/publications/papers/ogsa.pdf` (Accessed September 2006)
9. Globus Toolkit Version 4: Software for Service-Oriented Systems. I. Foster. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005. `http://www.globus.org/toolkit/` (September 2006)
10. EU IST 6th FP project K-Wf Grid – Knowledge-based Workflow System for Grid Applications. `http://www.kwfgrid.eu/` (Accessed September 2006).
11. Hluchý, L., Astalos, J., Dobrucký, M., Habala, O., Simo, B., Tran, V.D.: Flood Forecasting in a Grid Computing Environment. LNCS vol. 3019, 2004, Springer, Berlin. ISBN 978-3-540-21946-0, pp. 831-839.

Application of K-Wf Grid Technology to Coordinated Traffic Management

M. Masetti, S. Bianchi, and G. Viano

Softeco Sismat S.p.A., Via De Marini 1- Torre WTC, 16149 Genova, Italy

Abstract

The Coordinated Traffic Management pilot application developed by Softeco Sismat S.p.A. over the K-Wf Grid (EU IST-2002-511385) middleware targets the computation of traffic air pollutant emission in a urban area and has been developed in tight collaboration with the Urban Mobility Department of the Municipality of Genoa, which provided a monolithic implementation of the model for the pollutant emission calculations, the urban topology network and real urban traffic data. The objective is computational-wise challenging and provided a valid testbed for the K-Wf Grid middleware, allowing to evaluate and demonstrate the benefits of the application of the project results to a real environment as well as to deliver pre-competitive application frameworks and tools to be exploited in concrete business opportunities. As required by the internal architecture of the K-Wf Grid middleware, the CTM application workflow has been divided into several different steps in order to allow the semi-automatic composition of services and the definition of a set of ontologies which describe the CTM domain and feed the system with the information needed for the proper selection and execution of services.

1 Introduction

The Knowledge-based Workflow System for Grid Applications (K-Wf Grid project IST programme 511385) enables the knowledge-based support of workflow construction and execution in a Grid computing environment. In order to achieve these objective the system developed is able to compose semi-automatically a workflow of Grid services, execute the composed workflow application in a Grid computing environment, monitor the performances of Grid infrastructure and applications, analyse the resulting monitoring information, capture the knowledge contained in the information by means of intelligent agents and reuse it to construct more efficient workflows for new Grid applications.

In order to demonstrate the exploitability of the K-Wf Grid middleware, a Coordinated Traffic Management (CTM) pilot application has been developed and tested: the main goal of this application is the computation of air pollutant emission produced by private/public transportation in a urban area.

2 Application

2.1 Functionalities

CTM algorithms apply on a city road network. A city network can be considered from two different perspectives: a topological view (districts polygonal, nodes coordinates, distances, areas,...) and a graph view (basically, a directed acyclic graph or DAG). The main CTM application functionalities implemented, deployed and tested are:

- Best Route Calculations: given two different city districts, first topological calculations are applied to compute the internal nodes of start and end zones, then for each start node the Dijkstra Single Source Shortest Path algorithm is applied;
- Traffic Flows Calculations (over the best routes, given real traffic data): main traffic flows are derived from an origin/destination (O/D) matrix: flows are then split *per path* taking into consideration path length and vehicles average speed on path links;
- Air Pollutant Emission Calculations: an existing model (PROGRESS – PROGramme for Road vehicles EmiSSions evaluation in Genoa, developed by the University of Genoa) has been ported on the grid middleware. The model can take care of a massive amount of data and handles up to 8 different vehicles categories and calculates the emissions for four most critical pollutants (CO, HC, NOx and particulate matter).
- Data Graphical Representation: graphical services are used to plot computations results in diagrams and charts. Analysis results are also represented in SVG (Scalable Vector Graphic) format for an easy access from any web browser.

Each functionality can be considered as a self-consistent scenario and can be used singularly or as an intermediate step for more complex use cases. In order to enrich the test bed of the pilot application and to offer the possibility of evaluating some benchmarks, for each aforementioned functionality implementations with different payloads have been registered into the middleware knowledge base and made available to the user.

2.2 Test-bed and Software Architecture

The application is composed by a set of SOAP based web services distributed on different nodes of the K-Wf Grid network (Genoa – 1 node, Innsbruck – several nodes, Athens – 1 node, Bratislava – 1 node, Cracow – 3 nodes).

Plain SOAP web services proved to be the good solution to provide an early functional release of services for the first prototype, helping the consortium in testing and evaluating the K-Wf Grid system. All packages have been developed in Perl 5 as Perl proved to have a short application deployment time (with respect to more advanced technologies as Java) and provided several useful base packages on the CPAN (the worldwide Perl Archive Network). Relying on a base of ready-made packages was a key factor in respecting the deployment milestones table.

The application is organized in the following main packages:

- `ctm.util`: base utility package;
- `ctm.util.topology`: topology related classes and interfaces;
- `ctm.util.graph`: graph related classes and interfaces;
- `ctm.visum`: package containing all the interfaces and classes to handle a VISUM format network topology description file;
- `net.kwfgrid.ctm`: package containing application services. The exposed application services are the following:
  - `net.kwfgrid.ctm.netFileParser`: exposes methods to deal with the network topology and the computation of the best routes
  - `net.kwfgrid.ctm.PathLengthCalculator`: exposes methods to compute roads and paths lengths
  - `net.kwfgrid.ctm.TrafficFlowCalculator`: exposes methods to compute the traffic flow
  - `net.kwfgrid.ctm.AirPollutionEmissionCalculator`: exposes methods to compute emissions of pollutants
  - `net.kwfgrid.ctm.util.Session`: exposes methods to provide a transaction layer during workflow execution.

Each package (library) has been equipped with a testing environment, whereas software test coverage is quite good and reaches almost 90% of the code.

2.3 Workflows and Use Cases

The CTM workflow, depending on the specific scenario tackled by the operator, is dynamically composed by the K-Wf Grid infrastructure: different workflows can be executed concurrently and each workflow exploits services provided by different nodes, the selection based upon monitoring data. Also application data are spread on the different nodes of the grid. Different sub-workflows can be derived from the main one and can be considered as specific application use cases:

- Best routes computation between two different city zones;
- Traffic flows computation;
- Traffic flows computation due to a traffic jam (blocked road).

Best route computation involves actually a wide range of complex computations for which different input data is needed (above all, a topology representation of the city road network). All the needed computations are performed by operations exposed by the net.kwfgrid.ctm.NetFileParser web service.

The topology describes basically a directed graph where for each element (vertex, edge, etc.) coordinates are provided. The Genoa network topology used for the CTM application provided the following data:

- city zones list: the city has been divided in more than 200 zones and the application computes the best paths between two city zones;
- zones polygonal: the polygonal represents the zone perimeter;

- network nodes (or vertex): a node represents a road junction or a square or any discontinuity in the network;
- network arcs (or links, or edges): an arc connects two nodes and is directed; a two-way road is represented by two different edges. To compute best paths and traffic flows edges have to come with some added information:
- number of lanes;
- average crossing speed for different vehicle types;
- maximum speed for different vehicle types.

The number of lanes and the average speed are used to give a *weight* to each edge, so to come to a weighted graph needed for the best path computation.

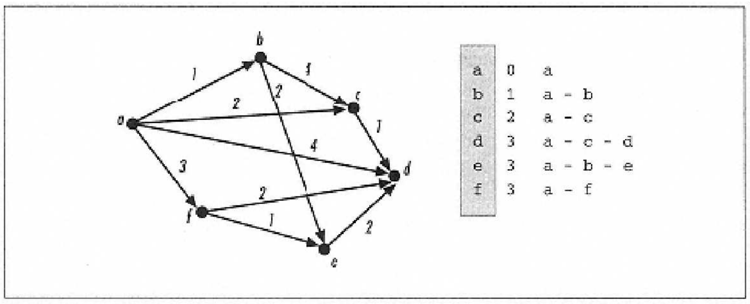
To derive the best routes sorted list from one city zone to another the network must be considered first from a topological perspective and then it is necessary to compute the list of nodes that belongs to the start and end zone. Traditional geometry inclusion algorithms are applied at this stage, in particular edge crossing. This computation can be performed in parallel for starting/ending zone and is performed by the following web services:

- `computeCtmStartZonePolyg`;
- `computeCtmStartNodes`;
- `computeCtmEndZonePolyg`;
- `computeCtmEndNodes`

After having computed the list of nodes belonging to starting/ending zones, the best route paths have to be computed. A very common task for a weighted graph is to find the shortest (lightest) possible path between vertices, applying *single-source shortest path* (SSSP) or *all-pair shortest path* (APSP) algorithms.

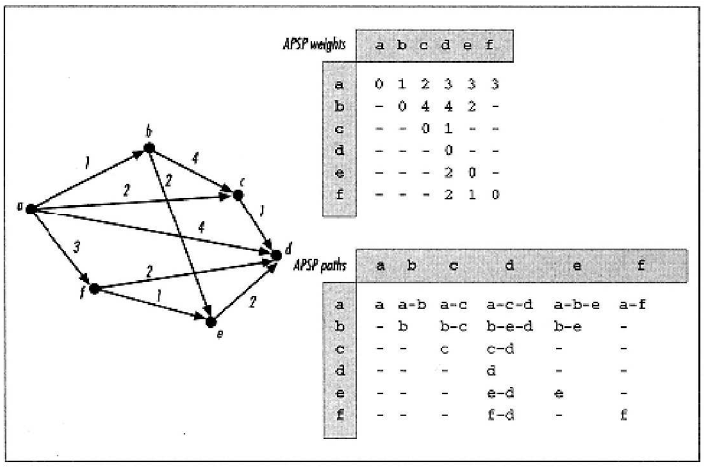
2.4 Algorithms

Given a graph and a vertex ("source"), the *single-source shortest paths* (SSSP) are the shortest possible paths to all other vertices. The *all-pairs shortest paths* (APSP) problem is the generalization of the single-source shortest paths: instead of starting always from a specific vertex and choosing always the lightest path, all possible paths are covered and all path lengths are calculated.



a	0	a
b	1	a - b
c	2	a - c
d	3	a - c - d
e	3	a - b - e
f	3	a - f

Fig. 1: A graph and its SSSP.



APSP weights	a	b	c	d	e	f
a	0	1	2	3	3	3
b	-	0	4	4	2	-
c	-	-	0	1	-	-
d	-	-	-	0	-	-
e	-	-	-	2	0	-
f	-	-	-	2	1	0

APSP paths	a	b	c	d	e	f
a	a	a-b	a-c	a-c-d	a-b-e	a-f
b	-	b	b-c	b-e-d	b-e	-
c	-	-	c	c-d	-	-
d	-	-	-	d	-	-
e	-	-	-	e-d	e	-
f	-	-	-	f-d	-	f

Fig. 2: A graph and its APSP.

There are several levels of difficulty: are there only positively weighted edges, or are there also negatively weighted edges, or even negatively weighted cycles? A negatively weighted cycle (negative cycle) is a cycle where the sum of the edge weights is negative. Negative cycles are particularly nasty because looping causes the minimum to just keep getting "better and better" and ignoring negatively weighted cycles would mean choosing an arbitrary definition of "shortest". Shortest paths are then found by repeatedly executing a process called *relaxation*: if there is a better (shorter) way to arrive to a vertex, lower the current path length minimum at that vertex. The act of processing an edge this way is called edge relaxation.

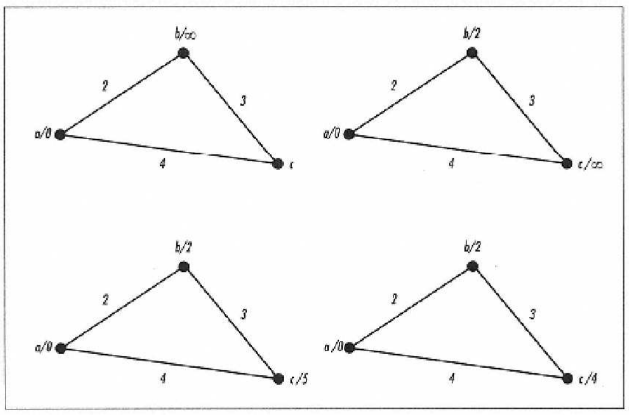
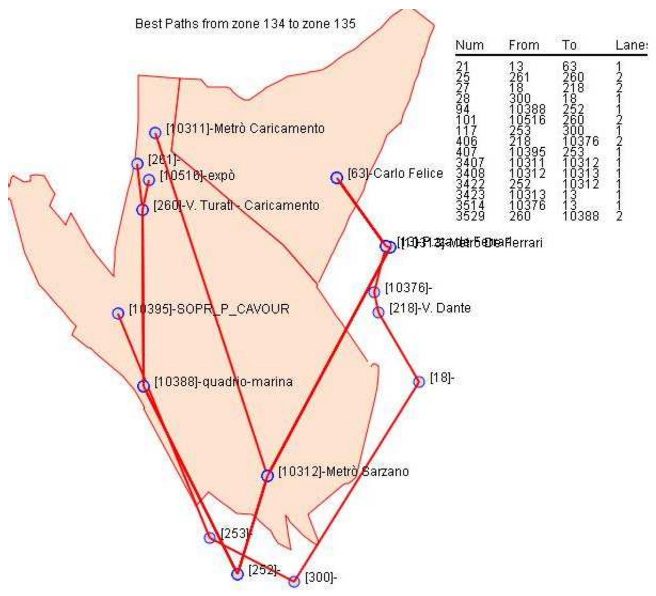


Fig. 3: Relaxing the edge a-c lowers the weight of vertex c from 5 to 4.

The *Dijkstra's single-source shortest paths* algorithm can be used only if all the edges are positively weighted and has been fully tested on the available network before being used in the project. If the graph weights do not vary, the relaxed network could be saved and reused, this is not done and relaxing computation is forced each time to stress the grid test-bed.

The best routes found are plotted on a SVG format file to be presented to the user using the K-Wf Grid user portal.



Num	From	To	Lane
21	13	63	1
25	261	260	2
27	18	218	2
28	300	18	1
94	10388	252	1
101	10516	260	2
117	253	300	1
406	218	10376	2
407	10395	253	1
3407	10311	10312	1
3408	10312	10313	1
3422	252	10312	1
3423	10313	13	1
3514	10376	13	1
3529	260	10388	2

Fig. 4: Visualization of results in SVG format: paths.

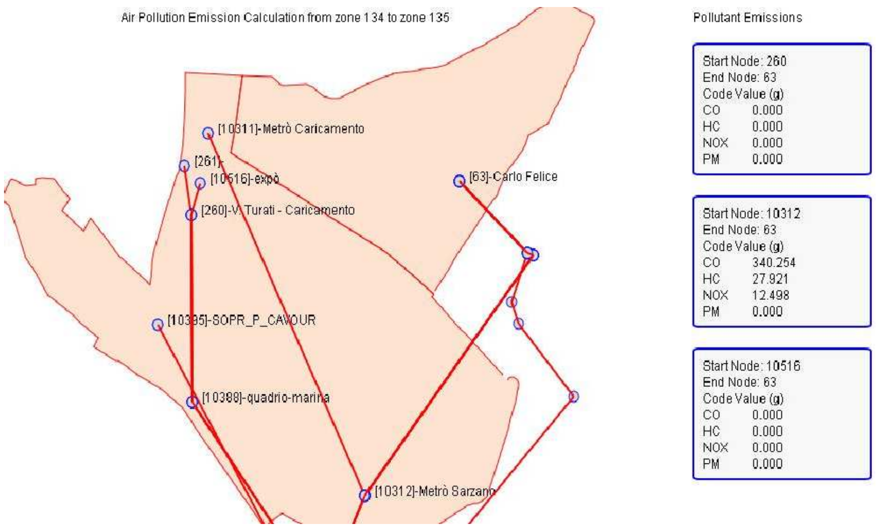
A key step in the use case implemented by the CTM application is the selection of a broken path, simulating the computation of pollutants emissions for critical situations (a road broken for maintenance or a car accident).

Once derived the best paths, the O/D (origin/destination) matrix is generally used to derive traffic flows for each path. O/D matrices represents flow data for each start/end zone in a given time shift. The flows are split among all possible paths taking into consideration the number of lanes and the average speed.

The formula to derive pollutants emission for a given time shift and a given vehicle type is the following:

$$emission = number\ of\ vehicles * distance * emission\ factor$$

where the emission factor is derived using a specific emission model. Emission factors are computed only for hot engines and are the result of a field monitoring lasted three years and conducted by the municipality of Genoa. Results are then included in SVG files for proper visualization.



```
Start Node: 260
End Node: 63
Code Value (g)
CO        0.000
HC        0.000
NOX       0.000
PM        0.000

Start Node: 10312
End Node: 63
Code Value (g)
CO        340.254
HC        27.921
NOX       12.498
PM        0.000

Start Node: 10516
End Node: 63
Code Value (g)
CO        0.000
HC        0.000
NOX       0.000
PM        0.000
```

Fig. 5: Visualization of results in SVG format: pollution emissions.

3 Achievements and Innovation

3.1 Overall Benefits

With respect to a traditional market solution, the CTM application exploits all the advantages that come from the adoption of a grid of resources instead of a classical monolithic solution:

- parallel and seamless task execution: the middleware is able to run independent tasks in parallel and on different grid nodes whenever possible, exploiting the power of the grid;
- computations potentially dispersed on different grid nodes;
- dynamic task execution depending on resource monitoring: performance monitoring and feedback information is used to constantly tune task execution for a better use of grid resources.

3.2 K-Wf Grid Value Added

Furthermore, the application benefits from the adoption of the K-WF Grid middleware, raising then from common grid applications:

- Semi-automatic application workflow construction: using the domain, service and data application knowledge, the middleware is able to compose the best workflow that fits the user needs;
- A new application development model: provided all the services to cope with each application task, the application workflow is built by the middleware; the user can anyway decide to save and reuse already composed workflows and re-compose them at some events (new services deployed,

changes in the grid of resources, new scenarios). A derived advantage with respect to monolithic market solutions is the possibility to cope with new scenarios that can arise in the traffic management domain;
- Services potentially automatically exploited in different domains: the more services are described in the middleware knowledge base, the more scenarios are reachable, as they can be reused independently and self-consistently.

3.3 Potential Application Scenario

The introduction of the aforementioned benefits in a complex domain such as the CTM application can be appreciated considering a real scenario, for example the estimation of the consequences of closing a downtown road on the daily traffic and on the air pollution. A conventional approach in fact would usually consist in performing manually several simulations in order to estimate the consequences of different scenarios and control measures. An experienced traffic manager would edit the inputs, calibrate the parameters, launch the simulation, collect the results, perform the analysis, come to conclusions and plan actions. All these activities would be usually performed sequentially, on a single PC, with manual data management and offline evaluations - a waste of time and resources. K-Wf Grid fosters and supports the transition from a work organisation modelled on workflow based on "humans" which perform tasks by means of a number of monolithic applications, toward a "virtual workflow" implemented by a middleware with the transparent orchestration of grid services.

4 Conclusions

The CTM application has been a valid test bed for the K-WF Grid middleware and helped to cope with and solve middleware weaknesses that arose in the first stages of application development.

The application has been helpful to test core middleware functionalities and the robustness of the overall system. Currently CTM services are deployed on grid nodes at Softeco Sismat and University of Innsbruck premises.

As a result of the collaboration between an industrial IT partner and a potential real end user, the application also allowed an early evaluation of the impact of the adoption of grid-based and K-Wf Grid-powered solutions in a domain which actually offers interesting business possibilities.

This research has been partially supported by the K-Wf Grid project (EU IST-2002-511385). The authors thank the Mobility Department of the Municipality of Genoa for the support in domain analysis and application conception, design and evaluation.

The Potentials of Knowledge-Based Workflows and Grid Infrastructures in an ERP Environment: The K-Wf Grid Experience

Pantelopoulos Stelios, and Kalaboukas Konstantinos

Singular Logic, Al.Panagouli & Siniosoglou, 142 34, N.Ionia, Greece
emails: {spantel, kkalam}@singular.gr

Abstract
Current enterprise environment is characterized by rapid changes and fuzzy networks of inter-enterprise correlations and dependencies. Thus, effective decision making, which is strongly related to the company competitiveness, requires computation- and data- intensive tasks due to the complexity of the supporting algorithms and the massive storage of enterprise data. To solve computation- and data- intensive tasks and to integrate back-office business processes, we propose, in this paper, the utilization of knowledge-based workflow systems and grid technologies over service oriented enterprise infrastructures. Finally, we present the experience gained from the preparation and the execution of ERP-based workflow scenarios in a grid computing environment.

1 Introduction

Most enterprises are under mounting pressure to adopt and deploy reliable, high performance Enterprise Resource Planning (ERP) integrated solutions to gain benefits concerning the strategic planning and the operational and management control [1] [2]. Such enterprises are expected to find ways to enhance their competitiveness by offering new or improved services a) to their employees, who need better and faster access to business information; and b) to their customers who demand faster response to their requirements and improved quality of service and support.

Moreover, from the technological point of view, Service Oriented Architecture (SOA) is the de-facto technological basis for implementing state of the art business applications and platforms, including ERP systems [3]. The development and deployment of Service Oriented Business Applications, which constitutes a set of independently running services communicating with each other in a loosely coupled message-based manner, as well as the publishing of Web Services, have already made headway within large organizations, the technology will start filtering down to medium and small companies (SMEs) and will expand into supply chains [4].

As current enterprise environment is characterized by rapid changes and fuzzy networks of inter-enterprise correlations and dependencies, the effective decision making, which constitutes crucial factors concerning the company positioning and competitiveness in the enterprise environment, requires *computation- and* *data- intensive tasks* due to the *complexity of the supporting algorithms* and the *massive storage of enterprise data.* In addition, the enterprise employees, who are the real ERP users, demand fast access to machine processed enterprise data so as to support efficiently and execute effectively back-office business processes, leading to the faster delivery of vital information, to the optimization of the enterprise performance and to the enhancement of customers' satisfaction.

To solve computation- and data- intensive tasks and to integrate back-office business processes, we propose the combination of knowledge-based workflow systems with grid computing over service oriented enterprise infrastructures. The utilization of grid technology leverages the notion of SOA, by providing grid enabled business services, while the introduction of knowledge-based workflows facilitates the *dynamic, semi-automated composition* and *optimized execution of business processes*, delivering and ensuring high ERP and database throughput and performance at service levels.

In this paper, we document and present the experience and the lessons learnt gained from the customization, deployment and validation of an innovative, so called K-Wf Grid platform, developed in the frame of the EC co-funded IST project entitled "Knowledge-based Workflow System for Grid Applications" (K-Wf Grid) [5], in a testing ground in the business sector. The pilot application of the K-Wf Grid project aimed to enable the knowledge-based support of ERP-oriented workflow construction and execution in a Grid computing environment.

The structure of this paper is as follows: in the following section, we present an overview and the architecture of the developed K-Wf Grid system, while Section 3 identifies data- and computational- intensive ERP workflow-based scenarios to be utilized as K-Wf Grid pilot applications. The preparation and the execution of the ERP demonstrator are described in Section 4. Finally, Section 5 presents the results and the conclusions of ERP-related experiments undertaken, as well as the potentials of the K-Wf Grid platform in business environments.

2 The K-Wf Grid Architecture

We envisioned the future Grid as a large, distributed collection of Grid services, as well as more general Web Services [5]. These services serve different purposes: from data storage, general processing, network transfer, specialized simulations, data processing services to sophisticated middleware services such as scheduling, fault tolerance, monitoring and performance analysis. Moreover, the services are partly complementary and partly redundant, with competition between similar services or their groups. The K-Wf Grid system allows customized development of Grid applications, enabling the user to control the important functionality and quality of service parameters.

Figure 1 presents the overall layered architecture of the K-WfGrid system. The architecture is composed of four horizontal layers (A, B, C, D) and the vertical knowledge layer (K). The horizontal layers are the Web Portal (A), the Grid Application Building layer (B), the Grid Application Control layer (C), and the Low Level Grid Middleware layer (D). Each horizontal layer communicates only with the proximate layers so single layers can be modified or replaced without changing the whole system. The Web Portal (A), for example, receives data from the Knowledge layer (K) and communicates interactively with the components of the Grid Application Building layer (B) but does not have direct connection to the Low Level Grid Middleware (D).

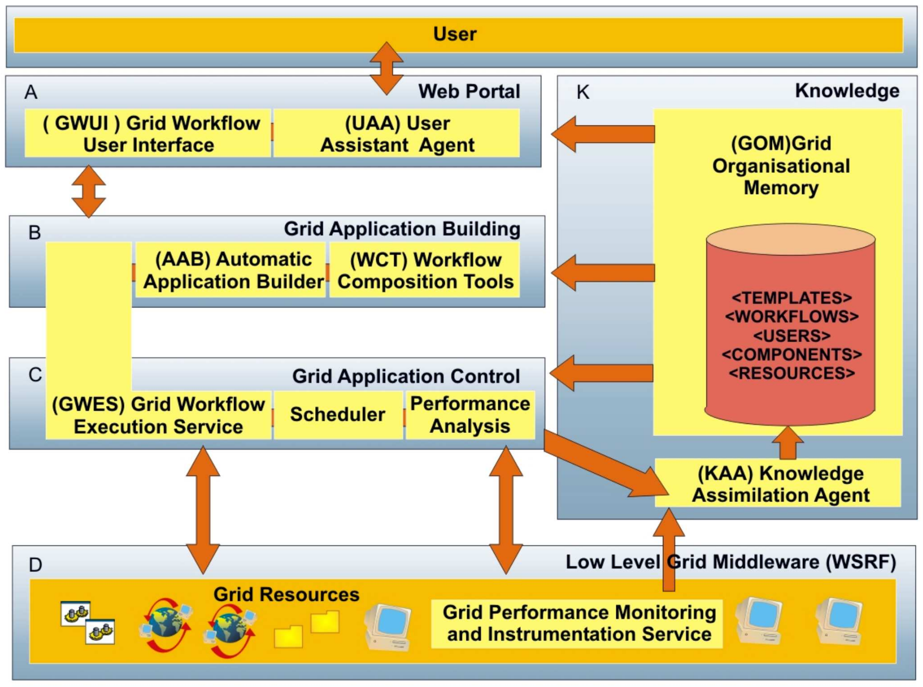

User
A
Web Portal
(GWUI) Grid Workflow User Interface
(UAA) User Assistant Agent
K
Knowledge
(GOM)Grid Organisational Memory
<TEMPLATES>
<WORKFLOWS>
<USERS>
<COMPONENTS>
<RESOURCES>
B
Grid Application Building
(AAB) Automatic Application Builder
(WCT) Workflow Composition Tools
C
Grid Application Control
(GWES) Grid Workflow Execution Service
Scheduler
Performance Analysis
(KAA) Knowledge Assimilation Agent
D
Low Level Grid Middleware (WSRF)
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 1: The K-Wf Grid System Architecture.

A user accesses the system through the Web Portal (A), which contacts the User Assistant Agent (UAA), responsible for creation of a user workspace, monitoring user actions and contacting other parts of the system when necessary. The system enables the user to interact with other users and provides support for developing workflow-based Grid applications. Such applications consist of a set of Grid services that interact with each other via standard protocols such as SOAP. An associated UAA contacts the tools located in the Grid Application Building framework (B), i.e. the Automatic Application Builder and the Workflow Composition Tool that constitute knowledge-based semi-automatic modelling services, able to propose known solutions to problems solved in the past by selecting appropriate services and interconnection patterns of interest. When a workflow description of the application is constructed, the workflow layer consecutively contacts the Grid Workflow Execution Service (C), consisting of a Grid Service Invocation and Control and a Grid Performance Analysis Service.

While Grid Service Invocation and Control also serves as an intelligent proxy for Grid resources in layer D, capable of performing observations of the resources, the Grid Performance Analysis Service works together with one or more Grid Performance Monitoring Services, located directly in layer D (thus being regular OGSA-compliant services) and gathers monitoring information derived from instrumentation inside the resources themselves.

The Knowledge layer K is comprised of two main parts, i.e. the Grid Organization Memory and the Knowledge Builder Agent. The Grid Organization Memory stores all observed, measured, directly entered or otherwise extracted data and makes it available to intelligent components of K-Wf Grid on layers A, B and C. The Knowledge Builder Agent is responsible for processing the measured and observed data for knowledge extraction and also for processing external data sources, such as UDDI registries or WSDL documents.

3 ERP-Based Workflow Scenarios

Taking into consideration the business lifecycle and the everyday activities of a typical enterprise of the rapidly growing Fast Moving Consumer Goods (FMCG) industry, encompassing a huge range of products and services in manufacturing, distribution and retailing, three crucial ERP usage scenarios have been identified regarding the competitiveness and the total performance of the enterprise.

The first scenario, so-called "Product Cost Analysis", concerns the calculation, real-time, of the cost of each product type traded by the enterprise, based on the long-term product orders made by the enterprise (Fig. 2). Imagine that the FMCG enterprise trades over than five thousand different product types, and places over ten orders per product type and per month to its suppliers – which makes us over fifty thousand orders per month – purchasing, in the frame of each order, products in different price and shipping details, e.g. cost. In case, the FMCG enterprise decides to adopt a pricing policy that implies a standard profit rate per product type (e.g. the enterprise should sell a product with 30% profit over the calculated mean cost), it is obvious that the everyday or day-by-day calculation of the final price of each product type constitutes a really data intensive workflow-based task.

The second scenario, entitled "Orders Management and Approval", concerns the approval (of the rejection) of all customers' orders submitted to the enterprise based on the financial reliability of the customers and the availability of the selected products.

Let's imagine that the FMCG enterprise receives daily over than ten thousand orders, each of them containing a set of ten product types on average, from over five thousand customers. First of all, the ERP user has to check the financial reliability of all customers having placed an order, taking into account customer's credit, the order's total cost, as well as a predefined acceptance threshold, in order to proceed in the acceptance of the order. In case, the financial reliability of the customer is proved, the ERP user, so as to finally accept the order, should check the availability of the product types and quantities listed in the specific

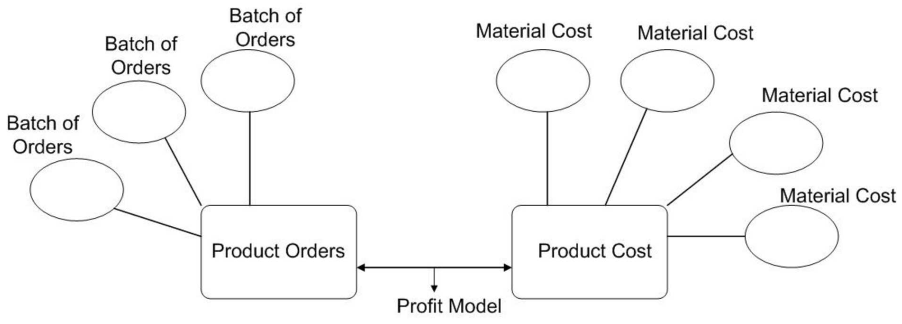


Fig. 2: The "Product Cost Analysis" problem overview.

order either in the enterprise inventory, which could be the real time aggregated result of the enterprise warehouses stock, or in the pending enterprise orders places to its suppliers, which are scheduled to be delivered in predefined time margins. It is obvious that the everyday management and final approval of each customer's order constitutes also a data intensive workflow business application.

Finally, the third ERP-based scenario (Fig. 3), entitled "Products Stock Management", being the most critical one, concerns the calculation of the required quantities for each product type, traded by the enterprise, to be ordered in a weekly/monthly basis, so as the future products stock level in the enterprise warehouses to remain above a predefined, safe level.

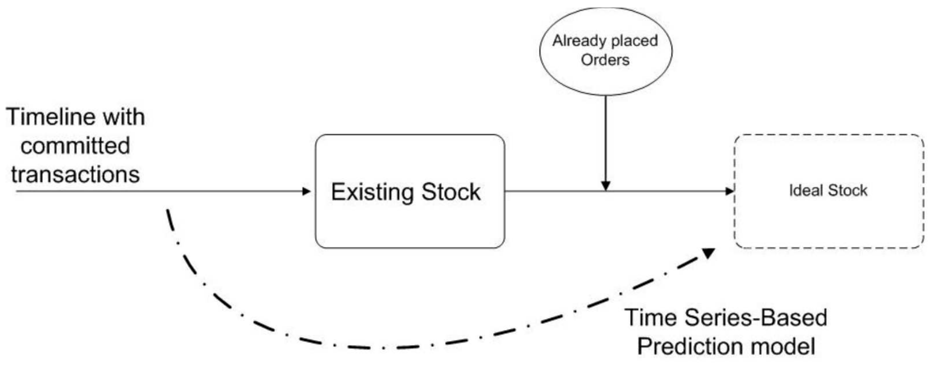

Already placed Orders
Timeline with committed transactions
Existing Stock
Ideal Stock
Time Series-Based Prediction model

Fig. 3: The "Products Stock Management" problem overview.

The "Products Stock Management" scenario utilizes several time-series analysis and prediction models (i.e. both the Single and the Triple Exponential Smoothing models [6]), to predict the forthcoming sales, and calculates the next-period orders of each product type traded by the enterprise, based on the long-term product sales and stock management algorithms. This module is practically a module of the Data Warehouse management component of the ERP. Taking into account the fact that a typical FMCG enterprise trades over than five thousand different product types, and places over ten orders per product type and per month to its suppliers, which makes over fifty thousand orders per month, as well as the computational needs of complex forecasting algorithms, the given "Products Stock Management" scenario is proved to be both a data- and a computational- intensive ERP-based service.

The process complexity characterizing all these above described ERP-based use cases require the introduction of knowledge-based workflows, to facilitate the composition and optimized execution of business processes, while grid infrastructure is required to address both data- and computational- intensive needs of the selected workflow-based applications. The K-Wf Grid platform will be deployed and validated through these three identifies use cases providing the ERP users, e.g. accountants, sales and finance managers, who has no experience in grid computing and workflow management systems, with an integrated tool and user friendly interface to execute parameterized, scenario-specific, optimized business processes.

4 Utilization of the K-Wf Grid Platform

Once the selection of either data- or computational- intensive ERP workflow-based applications, among a set of pre-identified ERP-based use cases, we have developed and exposed web service resources to the K-Wf Grid system, based on the pre-selected application scenarios, so as to facilitate the utilization of ERP applications by the K-Wf Grid system, which included a cluster of machines hosting the GT4 toolkit constituting the grid infrastructure (Fig. 4).

The utilization of the grid infrastructure in the frame of usage scenarios is summarized as follows: a) the data space of the ERP deployed in the enterprise could be bound as Grid Resource in the deployed grid infrastructure, in order to avoid continuous and computational intensive database query each time the above defined process is executed; b) the execution of the above defined use cases could be parallelized utilizing the computational resources registered in the deployed grid infrastructure (e.g. the management of 10000 customers' orders daily could be executed in 30 computers constituting the grid infrastructure, rather than in a single server); and c) the execution of the above defined workflows could be parallelized in the process space, (e.g. the check of the customer's Financial Reliability and the check of the availability in stock of the products placed in customer's order constitute non-interrelated services and could be executed in parallel).

5 Conclusions

The adoption and deployment of K-Wf Grid in enterprise environments, characterized by complex intra- and inter- organizational business processes that

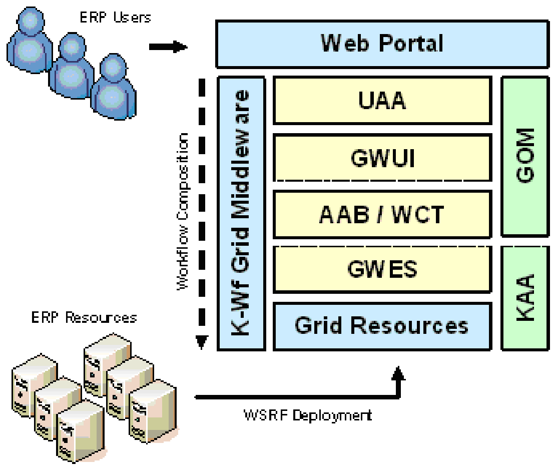


Fig. 4: The ERP Pilot Application.

orchestrate business-driven services deployed by customizable business software systems, scale existing systems horizontally using a grid-enabled service-oriented architecture.

More specifically, K-Wf Grid approach and platform a) allows semantically-assisted search, discovery and selection of appropriate tasks/services, stored in services repository that could typically contain some hundreds of services, with their desired functionality in order to compose a business-driven e-workflow and to establish connections among these tasks (control and data flow); b) allows the user to identify, at design time, the operational metrics of discovered services and grid resources, including timeliness, quality of products delivered, cost of service, and reliability, facilitating, thus, the composition of optimized grid-enabled e-workflows; c) supports the semi-automated generation, storage, and reuse of business e-workflows with a level of parameterization; d) shortens calculation time of data- and computation- intensive business-driven ERP-based processes, and improves enterprise competitiveness delivering instantly high quality computations; and e) enables simple ERP users, with no experience in grid computing and workflow management, to benefit from the emerging Grid-enabled Service Oriented Computing by offering user-friendly interfaces to a complex Grid-based environment.

References

1. E-Business Insight – ERP, CRM and Supply Chain Management. (11th March 2005). ERP Benefits – Operational Control, Management Control and Strategic Planning. Retrieved from `http://www.sysoptima.com/erp/erp_benefits.php` (15th September 2006)

2. State of Iowa, ERP Steering Committee. (2001). ERPN – Enterprise Resource Planning Newsletter. April 2001, Vol. 1, Issue 2. Retrieved (15th September 2006) from http://www.infoweb.state.ia.us/newsletter/erp/erp_apr.pdf
3. Bouras, A., Gouvas, P., Friesen, A., Pantelopoulos, S., Alexakis, S., Mentzas, G. (2006). Business Process FUSION based on Semantically-enabled Service Oriented Business Applications. In the proceedings of the 2nd Workshop on Web Services Interoperability (WSI 2006), 2nd International Conference I-ESA 2006 for Interoperability for Enterprise Software and Applications, 20-24 March, Bordeaux, France.
4. IBM Whitepaper. (2004). Service-Oriented Architecture and Web Services: Creating Flexible Enterprises for a Changing World. October 2004. Ziff Davis Media Custom Publishing.
5. Knowledge-based Workflow System for Grid Applications (K-Wf Grid IST project). Retrieved from http://www.kwfgrid.eu/ (15th September 2006).
6. National Institute of Standards and Technology (NIST) / SEMATECH, e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, (21st August 2006).

Software Engineering Aspects of K-WfGrid

Marian Bubak[1,2], Maciej Malawski[1], Piotr Nowakowski[2], and Steffen Unger[3]

[1] Institute of Computer Science, AGH Academy of Mining and Metalurgy, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] ACK Cyfronet AGH, ul. Nawojki 11, 30-950 Kraków, Poland
[3] Fraunhofer FIRST, Kekuléstraße 7, 12489 Berlin, Germany

Abstract
The aim of this paper is to present aspects of software engineering in a large-scale European Grid research project, using the K-WfGrid project as an example. This paper presents the considerations which should be taken into account when establishing, carrying out and disseminating the results of a joint scientific research enterprise.

Keywords: Software engineering, Grid, European research

1 Introduction

The paper describes in detail the quality assurance methodology and guiding rules together with processes supporting their realization in all phases of the software development process, from user requirements analysis to integration testing. K-WfGrid software engineering methodologies have been set forth in the the K-WfGrid Quality Assurance plan, which bases on both accepted standards and previous experience with drafting integrated QA procedures for CrossGrid (such as the IEEE-compliant SRS templates, Design Specifications and the deliverable review procedure), and on the experiences of other affiliated European Grid projects, most notably the DataGrid Project [2] and GridLab [3]. IEEE template 730-1989 (see [4]) was used for the creation of the K-WfGrid QAP.

2 The Aims and Scope of K-WfGrid

An important yet often neglected aspect of European and other international research projects is the proper application of software engineering methodologies, relevant for such large-scale distributed scientific collaborations. This paper aims to present some of the methods involved in successful management and control of software engineering in a Grid development project, on the example of Knowledge Workflow Grid (K-WfGrid) which is a FP6 IST undertaking.

K-WfGrid addresses the need for a better infrastructure for the future Grid environment. The Grid as a vast space of partially-cooperating, partially-competing Grid services is a very dynamic and complex environment. In order to address the complexity in using and controlling the next generation Grid, the K-WfGrid consortium adopted the approaches envisioned by semantic Web and Grid communities in a novel, generic infrastructure. The K-WfGrid system assists its users in composing powerful Grid workflows by means of a rule-based expert system. All interactions with the Grid environment are monitored and evaluated. The knowledge about the Grid itself is mined and reused in the process of workflow construction, service selection and Grid behaviour prediction. Workflows are dynamic and fault-tolerant beyond the current state of the art. The K-WfGrid system is generic by providing domain-independent system components, freeing the user from the burden of complex Grid usage and maintenance. Specific application knowledge is added in a customization phase carried out by system integrators including SMEs. These abilities are applied to three different application domains regarding scientific simulations (flood forecasting simulation) as well as industrial applications (ERP and traffic management). K-WfGrid contributes to strengthen SMEs and create progress by aggregating SME contributions with those of research organizations and global players. Two industrial SMEs – Softeco and LogicDIS – are involved in the project, enabling a strong focus on SME research and application, including transferability of results to real-world practice and strengthening of the European SME landscape. The achievements of K-WfGrid help bring the benefits of a global computation and information environment to a broader user space beyond the computer science community.

K-WfGrid was scheduled for two and a half years, officially starting in September 2004 and lasting until March 2007. The Project was divided into four distinct phases, as exemplified by its milestones:

- Initial phase, including requirement definition and merging of specifications (Months 1-6),
- Prototyping phase, centered on the development of a functional system prototype (Months 7-15),
- Refinement phase, focused on improving the prototype and implementing additional functionality, in preparation for a complete, stable release (Months 16-24),
- Testing and evaluation phase (Months 25-30).

Each phase was set to produce different artifacts, expressed in the form of deliverables, and each will require different approaches to quality assurance. In addition, each of the interim development phases will conclude with the delivery of a new *software release*. The QA procedures applicable to each phase of the Project were fully described in the Quality Assurance Plan, presented in Fig. 1.

Due to the scope and structure of the project a custom software engineering approach has been worked out by the Project consortium, basing on accepted software development methodologies [1] but also taking into account the specifics of developing software within an international collaboration and their impact on such processes as software design, prototyping, testing and quality assurance.

Software engineering in K-WfGrid covers the following aspects:

- K-WfGrid quality objectives and their role in software development,
- organizational structure of the project, including project Work Packages (WPs), project tasks (the basic organizational unit of the project) and project management, project documentation expressed in the form of deliverables and publications produced by K-WfGrid partners. This concerns

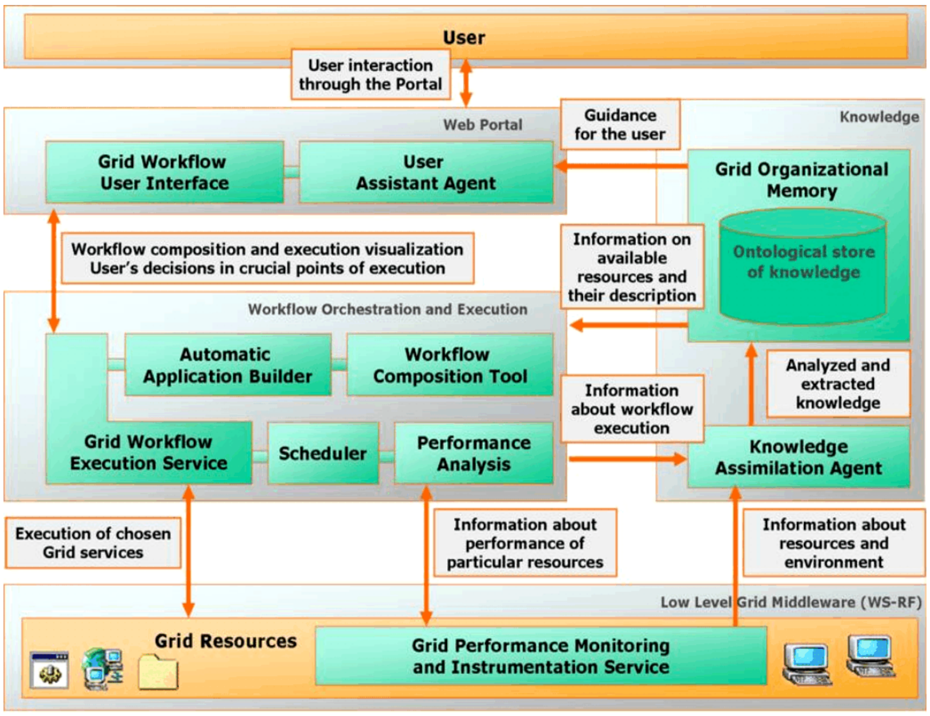

User
User interaction through the Portal
Web Portal
Guidance for the user
Knowledge
Grid Workflow User Interface
User Assistant Agent
Grid Organizational Memory
Ontological store of knowledge
Workflow composition and execution visualization
User's decisions in crucial points of execution
Information on available resources and their description
Workflow Orchestration and Execution
Automatic Application Builder
Workflow Composition Tool
Analyzed and extracted knowledge
Information about workflow execution
Grid Workflow Execution Service
Scheduler
Performance Analysis
Knowledge Assimilation Agent
Execution of chosen Grid services
Information about performance of particular resources
Information about resources and environment
Low Level Grid Middleware (WS-RF)
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 1: The Architecture of K-WfGrid, as defined during the final implementation phase of the Project.

both the physical layout of K-WfGrid deliverables and the deliverable submission and acceptance procedure, involving the Internal Review Board and Quality Assurance. An official K-WfGrid Publication Policy is also defined,

- generic conventions, standards and metrics which the Project adheres to, including: tools used within K-WfGrid (for publication as well as programming),
- coding conventions for all programming languages used within K-WfGrid,
- the Central Repository and its means of usage,
- the release preparation process,
- other standardized conventions to be followed,
- the K-WfGrid software development process and the role of QA procedures at each step of the Project's timeline. This involves both the technical aspects of the project as well as the reporting that goes with it and is divided into the following sections:
  - the software requirements review (initial phase),
  - the software design review (first development phase),

  - unit testing procedures (all development phases),
  - release and prototype reviews (all development phases),
  - testbed QA (all development phases),
  - publication reviews (all phases),
  - managerial reviews (all phases).
- the K-WfGrid corporate identity as a set of rules, captions and graphical elements to be used in official Project publications.

3 The K-WfGrid Quality Assurance Plan

In order to maintain a firm grasp of the course and aims of the K-WfGrid project, as well as exert control over how these objectives are pursued, a Quality Assurance Plan had to be developed, to provide quality support to partners, with respect to the main Project characteristics.

The plan defined the following Quality Objectives for the Project:

- All deliverables are to be handled in on time and subjected to a review process, so that their contents can be found satisfactory both by Project Management and by the European Commission which is entitled to review K-WfGrid documentation.
- All K-WfGrid Partners should follow the same set of conventions and metrics, which adhere to industry standards regarding good design and coding practices, as well as reviews and internal audits of the generated code.
- The code produced by the K-WfGrid technical tasks (i.e. Workpackages 1-5) should be handed in on time (as specified by the release workplan issued by Project Management) and perform as specified by their design (which is also subject to approval).
- All reviews are to be conducted at the scheduled times and their results are expected to be satisfactory, as defined in the objectives of each review.

3.1 Project Management

Fig. 2 depicts the central management of K-WfGrid. As shown in the picture, the main managerial bodies of the Project included the Project Coordinator, the Project Coordination Committee (PCC) and the Technical Board (TB).

The Project Coordination Committee coordinated the Project. The PCC assumes overall responsibility for liaisons between the Partners in relation to the Project, for analyzing and approving the results, for proper administration of the Project and for implementation of the provisions contained in the Consortium Agreement.The Technical Board supervised the technical and scientific progress of the Project in compliance with the Work Plan and the milestones defined. The TB also coordinated the activities within the different workpackages, analyzing emerging technical problems or time delays and report to the PCC, including the provision of decision alternatives.

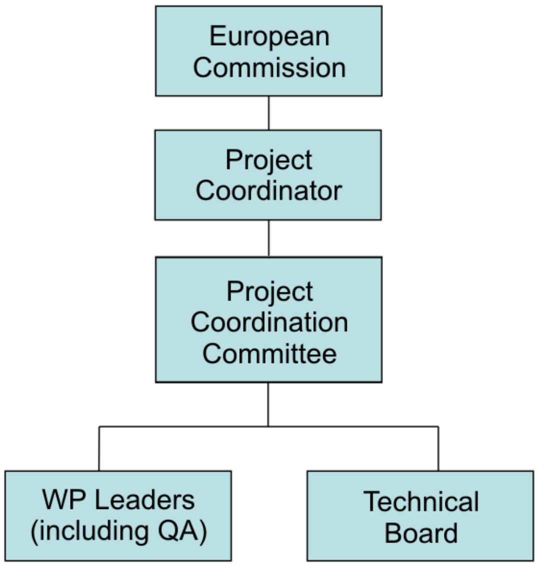


Fig. 2: The K-WfGrid Project Management hierarchy.

3.2 Quality Assurance Officer

As mentioned before, the Project involved a separate Quality Assurance workpackage. The Quality Assurance Officer of the Project reported to the Project Coordination Committee and performed the following functions with respect to the Project's QA objectives:

- established QA procedures to be followed by the entire Consortium, with respect to deliverable preparation, usage of tools, standards conformance, reporting schemes, ensuring the validity of Project code and issue identification,
- instituted a proper internal review process for Project deliverables,
- performed technical reviews of Project documentation and deliverables, with particular emphasis on conformance with this Quality Assurance Plan,
- identified Project-related risks and provided relevant information to the Project Coordination Committee in a timely and accurate manner, so that mitigation strategies could be implemented.

3.3 Project Corporate Identity

In order to assert K-WfGrid corporate identity, a set of document templates were provided and all K-WfGrid partners used the available K-WfGrid identity characteristics, namely:

- standardized document templates, both for deliverables and other documents,
- standardized transparency templates,
- standardized presentation templates (in PowerPoint),
- the K-WfGrid logo,
- uniform color schemes for printed material and presentations,
- references to the European Union as sponsoring the Project.

4 Quality Assurance – Review Procedures

Reviews are the means by which Quality Assurance is enforced in any collaboration. Hence, in order to facilitate reviews of K-WfGrid artifacts (deliverables), the following procedures were established. Figure 3 presents the QA plan applied to each technical deliverable before shipping it to the K-WfGrid Brussels office. By "technical deliverable" we mean all deliverables that were required to undergo such a process, save the managerial reports to the European Commission (i.e. Periodic Activity Reports, Interim Reports, Managerial Reports, Financial Reports, the Final Report etc.)

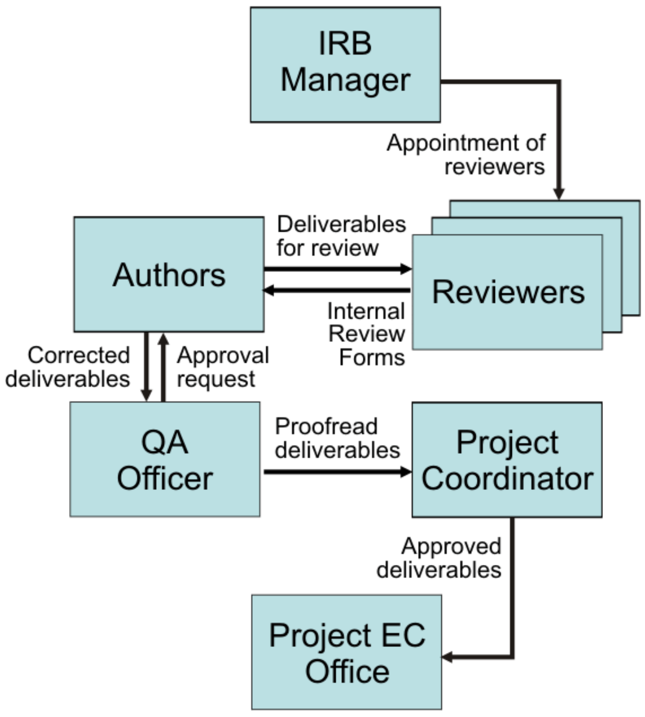


Fig. 3: Deliverable review process.

In order to facilitate deliverable review and institute the appropriate internal review mechanisms, as stipulated in the Technical Annex, an Internal Review Board was established. This Board consisted of a manager and a pool of potential reviewers, drawn from members of the Project's Technical Board. For each deliverable, the following process took place:

- The partner(s) responsible for the deliverable prepared a draft version approximately one month before the deliverable was due.
- The deliverable was submitted to the internal review process (as described below), under the guidance of the Internal Review Board manager. Two to three weeks were allowed for this process.
- For each document, the Internal Review Board manager assigned two or three reviewers to review the particular deliverable.
- The reviewers inspected the document's contents for conformance with the requirements of the Technical Annex. Each reviewer then issued a standard Internal Review Form (IRF), which detailed the results of the review and suggested changes required before the final version of the document was submitted. The IRFs were handed back to the IRB manager with a recommendation on whether the deliverable should be accepted, accepted with changes or rejected. The IRB manager made a final decision on the deliverable's acceptance or rejection. The IRFs were then provided to the partner responsible for the deliverable being reviewed.
- When the review was complete, the document's author introduced the necessary changes to the document and filled out a standard Corrective Action Form (CAF), which listed all the alterations called for by the Internal Review Forms prepared earlier. If the document was not approved, the author needed to rework and resubmit it for a second internal review.
- The updated deliverable was submitted to the Project's Quality Assurance Officer for editorial formatting and proofreading. One week was allowed for this process.
- Once the deliverable passed the QA stage, it was delivered to the Project Coordinator, who approved it and submitted it to the Project Office in Brussels.

5 Conclusions Regarding K-WfGrid Software Engineering and QA Aspects

The two-and-a-half year course of K-WfGrid ended in successful completion of the final Project Review in October 2006. Following the requirements of the Technical Annex, the Consortium prepared a number of documents and reports, all of which were accepted by the European Commission as having accurately depicted the final outcome and status of the Project. We therefore conclude that the safeguards and obligations specified by the Project's Quality Assurance Plan accurately reflect the requirements of multinational scientific collaborations as well as European research projects.

Acknowledgements. This research has been partially supported by the K-WfGrid project (EU IST-2002-511385). The authors thank the K-WfGrid Consortium for feedback received during the design and development process.

References

1. Eric J. Braude, Software Engineering: An Object-Oriented Perspective (John Wiley & Sons, Inc., 2001)
2. The DataGrid Project Management Website; `http://eu-datagrid.web.cern.ch/eu-datagrid/WP12/default.htm`
3. The GridLab Project Quality Assurnce Plan, available at `http://www.gridlab.org/Project/Deliverables.html`
4. The IEEE Software Engineering Standards catalog; `http://standards.ieee.org/catalog/olis/arch_se.html`

Author Index