Towards the Semantic Grid: Putting Knowledge to Work in Design Optimisation

Feng Tao, Liming Chen, Nigel Shadbolt
Department of Electronics and Computer Science, University of Southampton, UK
{ft,lc,nrs}@ecs.soton.ac.uk

Graeme Pound, Simon Cox
School of Engineering Sciences, University of Southampton, UK
{gep,sjc}@soton.ac.uk

Abstract: Modern computational Problem Solving Environments (PSEs) become more and more complex and knowledge intensive in terms of their integrated toolsets, in particular for engineering design search and optimization. Whether these toolsets can be assembled effectively to produce satisfactory results depends heavily on using the best domain practice and following decisions made by skilled engineers in practical situations. In this paper, a knowledge based approach is used to acquire this knowledge from existing sources and model it in a maintainable fashion. Ontologies are used to develop the conceptualization of a knowledge base. In order to reuse this knowledge to provide guidance at knowledge intensive points, we propose a knowledge based advisor, which can give a context-aware critique to guide users through effective operations of building domain workflows. The concept of a state panel is proposed to collect system state information, which is then reasoned about together with various task models in the JESS (Java Expert System Shell) environment. Two reasoning strategies are designed for different advising styles. A multilayer and client-server style architecture is proposed to illustrate how this advisor can be deployed to make available its knowledge advising service to a real workflow construction PSE in a maintainable fashion. Throughout we use the example of these knowledge services in the context of design optimization in engineering.

Key Words: Knowledge engineering, ontology, XML, knowledge base, JESS, production rules, workflow planning.
Category: SD I.2.5 (System tools and techniques, Expert [Artificial Intelligence])

1 Introduction & Background

The UK initiative in e-Science [7] addresses scientific collaboration, data management and process enactment on a global scale. Knowledge plays an important role in delivering these domain specific operations in an effective way. Knowledge

can play a key role in the semantic grid [5] which adopts a service-oriented approach to bridge the gap between grid computing and semantic web.

Knowledge services are important aspects of the GEODISE project [2]. This project aims to make available a toolbox for grid-enabled engineering design search & optimization (EDSO). A Problem Solving Environment (PSE) integrates this suite of tools with various knowledge services and database services aiming to provide an intuitive interface for engineering users.

Knowledge acquisition (KA) techniques [12] [13] have been used in [10] to elicit the domain knowledge and best practices in engineering design search and optimization. Based on this, various types of knowledge services [10] have been deployed in a knowledge portal architecture. In this paper, we focus in detail on one of the knowledge services in the knowledge portal: the advice service.

2 Design of knowledge based Advising system

The goal of knowledge engineering is to exploit knowledge and demonstrate how knowledge can be used in guiding users through a smooth operation of workflow construction in the EDSO problem solving environment.

2.1 Design of a state panel ontology

After years of working with experts, researchers in knowledge engineering and design domains have recognized the importance of the situation in which the expert acts, i.e. expert experience only applies in the context of a real problem solving situation [9]. This context can be modeled as a State Panel (SP) representing the environment's working memory. It should contain most key factors from which experts make their decisions. A state panel ontology is designed using Protégé [4] where each concept is modeled as a class with slots that resemble its properties. Furthermore, some constraints can be applied on the slot so that they can only be assigned pre-declared values as shown in Figure 1.

The state panel ontology captures three factors:

1. The user's skill level:
   This indicates whether the user is highly skilled and infers the appropriate level of advice to be given. In this paper, we set it as either "high" or "low" where in the first case, advice is not necessary.

2. Available resources: This stores information indicating resources already available for use. The type "Resource" is further sub-categorized into

GAMBIT[1] resource type and workflow resource type, each of which will be used in making decisions with the corresponding resource type involved. Here we only make use of the workflow resources and tasks. The decision to perform a certain task depends on whether available resources satisfy the input of this task. For example, the "input" slot of the workflow task concept takes an instance of the workflow resources.

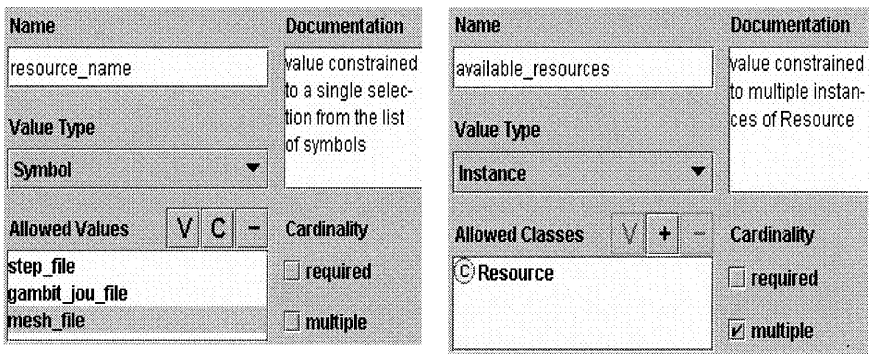

Figure 1 Protégé constraints on slot value

3. Finished tasks:
   Similarly, a finished task property is designed in the state panel since workflow related decisions are based on the pre-condition that certain other tasks have been finished. This is modeled as "depend_on" slot in the workflow task concept. The slot value is the instance of a workflow task.

An ontology is a standard description of the concepts and relationships that are being used or shared in a specific domain. It establishes the standard terminologies in that domain and makes sure that concepts are always consistent for the purpose of machine processing and reasoning. An ontology forms a conceptualization foundation for knowledge sharing in many artificial intelligence applications [11].

The state panel ontology is visualized as shown in Figure 2 using the Protégé Ontoviz plugin [6]. It can be noted that slots of the state panel related concepts always take values from a pre-defined enumeration. For example, in the "workflow_task" concept, the "task_name" is constrained to a single selection from a list of symbols. The "available_resources" slot in the "State_panel" concept takes only multiple instances of Resource type, where the resource name is again constrained to an enumeration of declared symbols. This characteristic guarantees that each symbol will be recognized and matched precisely in a reference engine, which we will describe later.

---

[1] GAMBIT is a preprocessing tool for fast geometry modeling and high quality meshing, both of which are crucial to successful use of Computational Fluid Dynamics

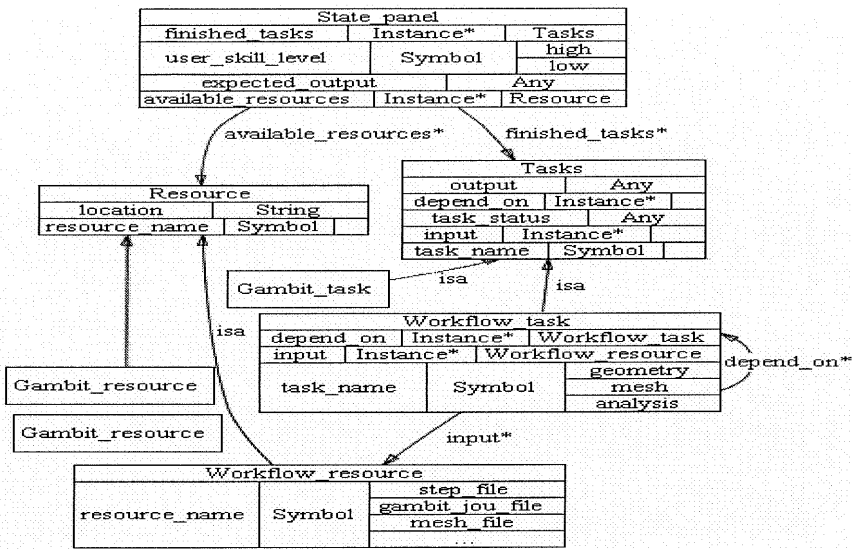

Figure 2 State panel ontology

2.2 Rule base and Inference engine

While the state panel resembles a short-term memory (working memory) of the current situation, a rule base contains the long term memory of accumulated experience in form of "IF-THEN" rules [8]. The rules are formulated in the form of the CLIPS [1] language which is then manipulated through a JESS [3] rule engine. The basic elements of the rules are concepts and pre-defined knowledge models upon which the forward chain reasoning can be applied to infer a solution. In certain circumstances, advice can be actions that change the state panel so that forward chaining happens. An inference engine can use a rule base to generate a prioritized list of actions appropriate to the current situation based upon the condition of a state panel.

3 Construction of the working memory

Jess is used in developing the advice system where the "working memory" is modeled as a set of pre-defined templates which may or may not initially contain any data, assertions or information. Three main templates are shown in Table 1 as they are defined in Jess. These templates serve as a connecting point joining the facts asserted and the facts in the LHS of the rules. In other words, both of them must conform to these templates.

(**deftemplate** workflow_task (**slot** name) (**multislot** input) (**slot** output) (**slot** finished?) (**slot** constrains) (**multislot** dependance))	(**deftemplate** state_panel (**multislot** finished_tasks) (**slot** user_skill_level) (**multislot** available_resources) (**slot** expected_output)) (**deftemplate** request_resource_provider (**multislot** requested_resources))

Table 1 Working memory templates in JESS

When starting the knowledge-based system, a set of facts are asserted. Each of them conforms to its corresponding template. This collectively forms an instance of the "working memory".

```
... ...
f-6  (MAIN::resource (name "step_file") (location "d:/geodise/res/airFoilStepFile"))
f-7  (MAIN::resource (name "gambit_jou_file") (location "d:/geodise/res/gambit.jou"))
f-8  (MAIN::workflow_task (name "geometry") (input nil) (output "step_file") (relevant_commands nil) (finished?
nil) (constrains nil) (dependance ))
f-9  (MAIN::workflow_task (name "mesh") (input "step_file" "gambit_jou_file") (output "mesh_file")
(relevant_commands nil) (finished? nil) (constrains nil) (dependance ))
f-10 (MAIN::workflow_task (name "analysis") (input "mesh_file" "fluent_jou_file") (output nil)
(relevant_commands nil) (finished? nil) (constrains nil) (dependance ))
f-11 (MAIN::workflow_task (name "optimisation") (input nil) (output "avs_file") (relevant_commands nil)
(finished? nil) (constrains "run time not very high")(dependance ))
f-12 (MAIN::workflow_task (name "doe") (input "budget_time" "single_run_time") (output nil)
(relevant_commands nil) (finished? nil) (constrains nil) (dependance ))
f-13 (MAIN::workflow_task (name "visulization") (input "avs_file" "others") (output "graph") (relevant_commands
nil) (finished? nil) (constrains nil) (dependance ))
f-14 (MAIN::state_panel (finished_tasks "geometry") (user_skill_level "low") (available_resources
"fluent_jou_file") (expected_output nil))
... ...
```

Table 2 Facts asserted to form the working memory

In Table 2, "geometry", "mesh" and "analysis", etc. are high level workflow tasks in EDSO. Each of them takes different resources as input and some of them have output and other properties defined. We demonstrate how input properties of the workflow task can be used together with the state panel fact to reason against the rules in order to provide workflow advice. To simplify the problem for illustration, the workflow advice focuses on suggesting what next step can be carried out depending on current resources availability as indicated in the state panel. Further advice can be triggered to suggest the resources that need to be obtained in order to perform a specific task.

4 Strategies of reasoning in the workflow advising system

The system supports two types of reasoning strategies. Both of them aim to provide state-driven advice to assist workflow planning.

4.1 Elimination strategy

This strategy designs and deploys rules in a way that allows one workflow task to be retracted from working memory if any of its input resource does not exist in the available resources specified in the state panel. After eliminating all tasks that cannot be carried out, those remaining are tasks that satisfy the current system state. The elimination strategy guarantees that any workflow task surviving has a full provision of input resources indicated in the state panel.

```
... ...
(defrule rule1
        (not (state_panel (available_resources $?x "step_file" $?y)))
        ?taskID<-(workflow_task(input $?a "step_file" $?b))
        =>
        (retract ?taskID)
        (printout t ?taskID " Retract this workflow task because it needs step_file as input, which
is not available according to      the state panel. "  crlf))
(defrule rule2
        (not (state_panel (available_resources $?x "gambit_jou_file" $?y)))
        ?taskID<-(workflow_task(input $?a "gambit_jou_file" $?b))
        =>
        (retract ?taskID)
        (printout t ?taskID " Retract this workflow task because it needs gambit_jou_file as input,
which is not available according to the state panel. "  crlf))
... ...
```

```
(defrule workflow-answer-1
        (declare (salience -10))
        (workflow_task (name ?n))
        =>
        (printout t "In term of the work flow, next step you can do: " ?n crlf))
```

Table 3 Rules designed for elimination strategy

Table 3 lists part of the rules coded in CLIPS. These rules are loaded into the JESS reasoning engine and their LHS are matched with the state panel and workflow task facts. The logic is quite simple: firstly, all workflow tasks are asserted into the working memory as possible candidates. Then for each available resource that is NOT specified as available in the state panel fact, if there is a workflow task fact whose input property includes that resource, then this rule is fired with the action of retracting (eliminating) that workflow task from the working memory ("$?x "step_file" $?y" expresses a pattern that matches to a list of literals that include "step_file", $?x is a JESS expression of multifields). The default salience of rules is 0 which makes sure that these rules are checked first before checking rule "workflow-answer-1", which has a lower salience of -10. After all the "retracting" rules have been checked (some of them may be executed), the "answer_rule" simply prints out all facts of workflow tasks that haven't been retracted yet. In other words, these tasks can be performed according to the current resource availability. Table 4 shows the reasoning result when applying the fact list to the rule set in Table 3.

Facts	*Reasoning results*
... ... (MAIN::workflow_task (name "mesh") (input "step_file" "gambit_jou_file") (output "mesh_file") (relevant_commands nil) (finished? nil) (constrains nil)) (MAIN::workflow_task (name "analysis") (input "mesh_file" "fluent_jou_file") (output nil) (relevant_commands nil) (finished? nil) (constrains nil)) (MAIN::state_panel (finished_tasks "geometry") (user_skill_level "low") (available_resources "step_file" "gambit_jou_file") (expected_output nil))	Found state panel with user_skill_level low in fact list. So Switching ON Advisor... <Fact-16> Retract this workflow task because it needs mesh_file as input, which is not available according to the state panel. In terms of the workflow, next step you can do: mesh

Table 4 Reasoning result using elimination strategy

4.2 Direct strategy with forward chain

The elimination strategy guarantees that any workflow task surviving has a full provision of input resources indicated in the state panel, but this is sometime too strict, for example, there might be some (not necessarily all) resources available for doing a workflow task, which the user might intend to do. However, the elimination strategy may rule out this possibility by retracting this task. The direct strategy is designed to relax this restriction. It assumes that the user may need to perform a task if s/he creates resources that match part of the task's input space. Therefore, depending on the situation, action is taken to suggest that this task can be carried out either straight away or on the premise of some further resources ($?expected_input in Table 5) being made available, where in the latter case, a new fact called "request_resource_provider" (defined in Table 1) is asserted with its property filled with this expected resource ($?expected_input multi-field variable).

A forward chain of potential rule activation happens at the assertion of the "request_resource_provider" fact. This is defined in the "resource_to_task" rule in Table 5. It tries to match the requested resources with the output property of every workflow task in the working memory. In this way, the rule fires and outputs suggestions of what workflow task should be preformed in order to obtain the requested resources.

```
(defrule task_input_against_resources
        (state_panel (available_resources $?resource))
        ?taskID<-(workflow_task(name ?n)(input $?inputs ))
        (test (> (length$ (intersection$ $?resource $?inputs)) 0))
        =>
        (printout t $?resource " is now available. " crlf)
        (printout t "You may be able to do " ?n )
        (bind $?intersection (intersection$ $?resource $?inputs))
        (printout t " by using " $?intersection crlf)
        (bind $?expected_input (complement$ $?intersection $?inputs))
        (if (= (length$ $?expected_input) 0)
    then
            (printout t "without having to obtain any further resources" crlf crlf)
    else
            (printout t "but still need input:" $?expected_input crlf crlf)
            (assert (request_resource_provider (requested_resources $?expected_input)))))
```

```
(defrule resource_to_task
        (request_resource_provider (requested_resources $?requested_resources))
        (workflow_task (name ?n) (output ?outputs))
        (test (member$ ?outputs $?requested_resources))
        =>
        (printout t ?n " can produce the resource " ?outputs crlf crlf))
```

Table 5 Direct matching strategy with forward training

Table 6 shows the result of reasoning by using this strategy. It is slightly modified for the purpose of clearer illustration. Workflow task facts stay the same as shown in Table 4. The two "request_resource_provider" facts are asserted in run time, which trigger a forward chain response aiming to discover workflow tasks that output the requested resoures. Text in the right column is the output from the JESS reasoning engine.

(workflow task facts are same as in Table 4) (MAIN::state_panel (finished_tasks "geometry") (user_skill_level "low") (available_resources "step_file" "gambit_jou_file" "fluent_jou_file") (expected_output nil))	found state panel with user_skill_level low in fact list. So Switching ON Advisor... ("**fluent_jou_file**" "**gambit_jou_file**") is now available. You may be able to do ***analysis*** by using ("**fluent_jou_file**") but still need input:("**mesh_file**") ***mesh*** can produce the resource **mesh_file** ("**fluent_jou_file**" "**gambit_jou_file**") is now available. You may be able to do ***mesh*** by using ("**gambit_jou_file**") but still need input:("**step_file**") ***geometry*** can produce the resource **step_file**
facts asserted in run time: (MAIN::request_resource_provider (requested_resources "mesh_file")) (MAIN::request_resource_provider (requested_resources "step_file"))	

Table 6 Reasoning result using direct strategy with forward chain

5 Framework of dynamic knowledge advising in workflow constructing PSE

We put all the pieces together to show an integrated framework capable of providing dynamic knowledge advice in the work flow construction environment. The framework is designed to be service-oriented and has a client-server architecture as shown in Figure 3.

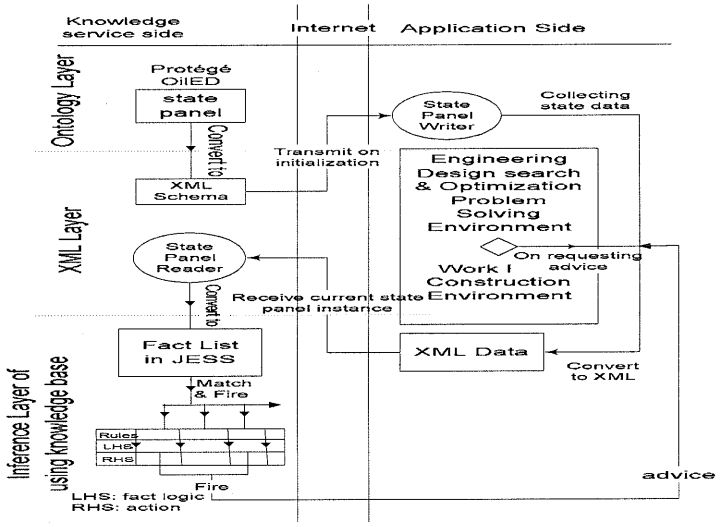

Figure 3 Knowledge advising system architecture

The knowledge advisor resides on the server side waiting for knowledge requests fired from the client side where a state panel writer collects key state information and passes it to the server side for processing and analysis. The state panel writer is internally operated by the workflow construction PSE so that any new state information, once available, is written to the state panel while users interact with the PSE. The state panel is a template of concepts with relationships and constraints. This template enumerates all possible factors that can potentially affect the advice. Relationships are expressed in the form of a concept hierarchy and attribute reference. Constraints make sure that factors can only take values as recognized concepts and terms so that they match precisely when doing inference with the rule base reasoning engine. Knowledge acquisition methods are used in designing the state panel template. We use the Protégé 2000 ontology editor for the design and maintenance of the state panel.

While this forms the ontology layer of the knowledge service, an XML layer transmits state panel information over the internet. This layer also guarantees that the template is translated to a format that is easy to interpret and process by machine. It is in this layer that templates modeled in Protégé are converted to XML Schema using

the XML tab plug-in in Protégé. This XML schema of the state panel is transmitted to the application side when it first initializes the knowledge advice service. A state panel writer on the application side collects environment data and generates an XML data file that conforms to the XML schema received. The XML data is then sent back to the server side where a state panel reader parses it and converts it to a set of facts used as input to the inference layer of the knowledge base.

6 Scenarios and a working example

The knowledge based advising system has been integrated into a large research prototype. This prototype aims to assemble a complete set of grid enabled optimisation services that could be managed via a knowledge portal. The problem is the design and optimisation of a NACA airfoil aiming to reduce the noise while maximizing the lift subject to a particular air flow. A typical workflow of engineering design search and optimisation usually starts from geometry design using standard CAD software such as Pro-E, where geometry related parameters are defined. Pro-E can produce a STEP file to represent the modelled geometry. In order to carry out numerical analysis, in particular dynamic fluent analysis in this case, on various parts of the geometry, mesh has to be done to provide more information about the surface of the geometry. Gambit is used to do this job. It can take the STEP file as input and output a mesh file. Gambit generates a piece of journal file logging every operation executed in its interactive session. This journal file can also be pre-edited or modified to support the batch operation of meshing, as well as some complex operations that can not be possibly done in the interactive session. Since the Gambit journal file is used to control the operation of meshing, it is often used as the second input parameter in the meshing task. With these two parameters, the gambit can produce a mesh file that describes the surface of the studied part of the geometry more in detail so that fluent analysis can be applied. Similarly, by using the mesh file and a fluent journal file as the input parameters to the Fluent software, objective function can be evaluated once to obtain the length of the single run time. Depending on the number of design parameters and the length of single run time, Design of Experiment (DOE) can be carried out repetitively to find the optimum of the objective function. In this case, we only focus on the workflow construction and the knowledge advising involved in the first three steps.

Figure 4 shows a working example of the knowledge based advisor integrated into the workflow construction environment. Various ontologies can be loaded and used to construct workflows of different domain. While users are building the workflow of engineering design search and optimization by selecting and configuring task components from the task ontology, the state panel is filled and sent to the reasoning engine of the knowledge base. The critiques that the knowledge base produces are displayed in the knowledge-based advice panel to suggest possible actions to be taken in the next step.

In the example, the workflow is related to the design of a NACA airfoil, a simple design that can be defined by three parameters. Since the mesh operation makes

available the mesh file as its output, the advisor, after checking the best practice, suggests that objective function analysis can be carried out using the mesh file as one of its input parameters. Besides, it also suggests that other resources are necessary in order to fulfil the suggested next step and how the user might be able to obtain the requested resources.

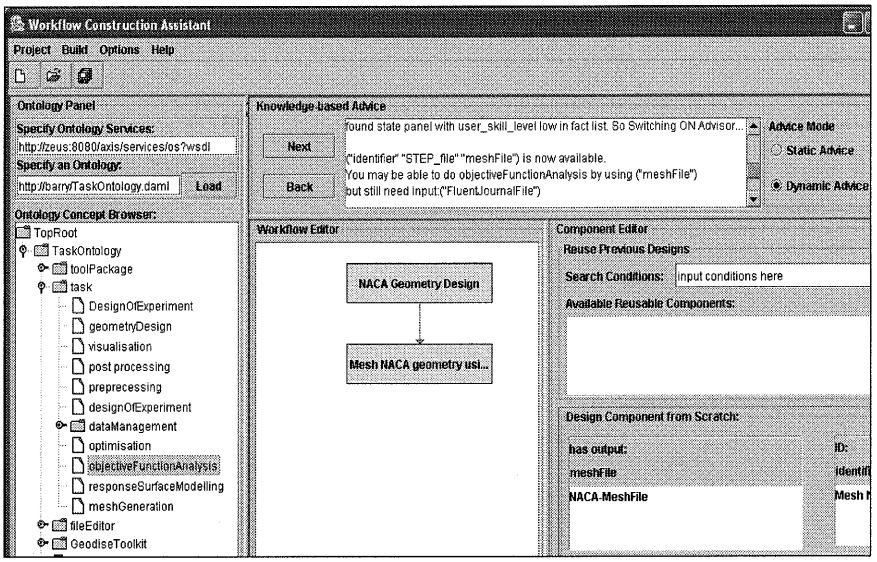

Figure 4 A snapshot of the advisor working in workflow construction PSE

7 Conclusions and future work

In this paper, we focus on the knowledge advising service aspect of the knowledge portal architecture. We demonstrate a knowledge based advisor that can reason over previously elicited domain knowledge and assist in building workflows by providing advice according to the current state. We demonstrate that ontologies can be used in constructing a maintainable conceptual foundation for the knowledge base. Two reasoning strategies are designed and tested. Experiments show that they are capable of deriving new knowledge, providing primary advice, best practice of the domain and helping users in making decisions at knowledge intensive points.

The deployment of the knowledge based advisor in the workflow construction PSE has been prototyped to assist engineering design search and optimisation. The integration shows that knowledge advice can be generated from the knowledge base. The advice is used to help users handle knowledge intensive operations in the PSE.

In the current demonstration, only a limited number of factors are taken into account in the reasoning, and the styles of advice are rather limited. However, we believe that

more factors can be encoded in the reasoning process together with more complicated reasoning strategies so as to provide more versatile advice.

Acknowledgements

This work is supported by the GEODISE e-Science pilot project (UK EPSRC GR/R67705/01). The authors gratefully acknowledge many helpful discussions with the GEODISE team. Thanks also go to K.O'Hara for syntax checking.

References

1. CLIP, http://www.ghg.net/clips/WhatIsCLIPS.html

2. GEODISE project homepage, http://www.geodise.org

3. Jess, the rule engine for the Java platform, http://herzberg.ca.sandia.gov/jess/

4. Protege homepage, http://protege.stanford.edu/index.html

5. Semantic Grid, http://www.semanticgrid.org/

6. The OntoViz Tab - Visualizing Protégé-2000 Ontologies, http://protege.stanford.edu/plugins/ontoviz/ontoviz.html

7. The UK e-Science programme and the Grid, http://www.escience-grid.org.uk/

8. B.G.Buchanan and R.G.Smith, "Fundamentals of expert systems," *The Handbook of Artificial Intelligence*, No. 4, 2002, pp. 149-192.

9. W.J.Clancey, *Situated Cognition: on Human Knowledge and Computer Representations*, Cambridge University Press, Cambridge, 1997.

10. L.Chen, S.J.Cox, C.Gobel, A.J.Keane, A.Roberts, N.R.Shadbolt, P.Smart, and F.Tao, "*Engineering Knowledge for Engineering Grid Applications*" EuroWeb2002 - The Web and the GRID: from e-science to e-business, Oxford, UK, http://www1.bcs.org.uk/DocsRepository/03700/3775/chen.pdf, 2002, pp. 12-25.

11. T.R.Gruber, "A translation approach to portable ontologies," *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 199-220.

12. G. Schreiber, H.Akkermans, A.Anjewierden, R.D.Hoog, N.Shadbolt, Van de Velde, and Wielinga, *Knowledge Engineering and Management*, The MIT Press, London, 2000.

13. N.Shadbolt, K.O'Hara, and L.Crow, "The Experimental Evaluation of Knowledge Acquisition Techniques and Methods: History, Problems and New Directions," *International Journal of Human-Computer Studies*, Vol. 51, No. 4, 1999, pp. 729-755.