
**Division of Informatics, University of Edinburgh**

**Centre for Intelligent Systems and their Applications**

# High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning

by

John Kingston

**Informatics Research Report EDI-INF-RR-0050**

**Division of Informatics** **November 2001**
http://www.informatics.ed.ac.uk/

# High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning

John Kingston

Informatics Research Report EDI-INF-RR-0050

DIVISION *of* INFORMATICS
Centre for Intelligent Systems and their Applications

November 2001

in Expert Systems with Applications, Volume 21, 4, November 2001.

**Abstract :**

As part of the DARPA-sponsored High Performance Knowledge Bases program, four organisations were set the challenge of solving a selection of knowledge-based planning problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases can be built quickly and efficiently. The domain chosen was workarounds; that is, planning how a convoy of military vehicles can work around (i.e. circumvent or overcome) obstacles in their path, such as blown bridges or minefields.

This paper describes the four approaches that were applied to solve this problem. These approaches differed in their approach to knowledge acquisition, in their ontology, and in their reasoning. All four approaches are described and compared against each other. The paper concludes by reporting the results of an evaluation that was carried out by the HPKB program to determine the capability of each of these approaches.

**Keywords** : knowledge acquisition, knowledge engineering, knowledge based planning, knowledge representation, ontology

**High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning**

John Kingston
AIAI, Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
J.Kingston@ed.ac.uk
Tel. +131 650 2732  FAX +131 650 6513

## Acknowledgements

The work described in this paper has been contributed to by many people: Gheorghe Tecuci, Mihai Boicu, Katie Wright and Mike Bowman at the Learning Agents Laboratory of George Mason University; Yolanda Gil, Bill Swartout and Jim Blythe at the Information Sciences Institute, University of Southern California; Ben Rode, Keith Goolsbey, Fritz Lehmann and Doug Lenat at Cycorp; Adam Pease and Cleo Condoravdi at Teknowledge; Eric Jones and Eric Domeshek at Alphatech; and Stuart Aitken at AIAI. Mention should also be made of David White, the primary domain expert; Dave Gunning and Murray Burke, who have managed the HPKB program; and Albert Lin at SAIC.

Running title: HPKB: Four approaches to Workaround Planning

**High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning**

Abstract

As part of the DARPA-sponsored High Performance Knowledge Bases program, four organisations were set the challenge of solving a selection of knowledge-based planning problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases can be built quickly and efficiently. The domain chosen was 'workarounds'; that is, planning how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as blown bridges or minefields.

This paper describes the four approaches that were applied to solve this problem. These approaches differed in their approach to knowledge acquisition, in their ontology, and in their reasoning. All four approaches are described and compared against each other. The paper concludes by reporting the results of an evaluation that was carried out by the HPKB program to determine the capability of each of these approaches.

# Introduction

The goal of the DARPA-sponsored High-Performance Knowledge Base (HPKB) program, which ran from 1997 to 1999, was to produce the technology needed to enable system developers to construct rapidly large knowledge-bases (with many thousands of axioms) that provide comprehensive coverage of topics of interest, are reusable by multiple applications with diverse problem-solving strategies, and are maintainable in rapidly changing environments. In the original proposal, it was envisioned that the process for constructing these large, comprehensive, reusable, and maintainable knowledge bases would involve three major steps:

- Building Foundation Knowledge: creating the foundation knowledge (e.g., selecting the knowledge representation scheme, assembling theories of common knowledge, defining domain-specific terms and concepts) to enable the construction and population of large, comprehensive knowledge bases for particular domains of interest -- by selecting, composing, extending, specializing, and modifying components from a library of reusable ontologies, common domain theories, and generic problem-solving strategies.
- Acquiring Domain Knowledge: constructing and populating a complete knowledge base -- by using the foundation knowledge to generate domain-specific knowledge acquisition, data mining, and information extraction tools -- to enable collaborating teams of domain (non-computer) experts to easily extend the foundation theories, define additional domain theories and problem solving strategies, and acquire domain facts to populate a comprehensive knowledge base covering the domains of interest.
- Efficient Problem Solving: enabling efficient problem solving -- either by providing efficient inference and reasoning procedures to operate on a complete knowledge base, or by providing tools and techniques to select and transform knowledge from a complete knowledge base into optimized problem-solving modules tailored to the unique requirements of an application.

The objective of HPKB was to develop, integrate, and test the technology needed to enable this process. The intention was to produce alternative knowledge-base development environments, which combined the necessary foundation-building, knowledge-acquisition, and problem-solving technologies into an integrated development environment, and to use those environments to build reusable knowledge-base components for multiple DARPA application projects.

## Challenge Problems

In the first year of the HPKB project, progress towards these goals was encouraged by setting up three competitive scenarios in which several technology developers were tasked to tackle a knowledge-based problem. These were known as "challenge problems". Each problem consisted of a collection of data and a set of sample problems with model answers; an evaluation was performed at the end of the period of development, in which the systems were tested on their ability firstly to handle new problems based on the same knowledge, and secondly to add new knowledge (in the same domain) rapidly, and to answer questions that drew on the new knowledge.

The common availability of input data in agreed formats, and the co-ordination of multiple technology developers in tackling a single challenge problem, was handled by two companies (Teknowledge and SAIC) acting as integrators; the efforts of the technology developers were thus co-ordinated into two "teams". The two teams took slightly different approaches, which are reflected in some of the systems developed; Teknowledge favoured a centralised architecture based on a large common-sense ontology (Cyc) while SAIC had a distributed architecture that relied on sharing specialised domain ontologies and knowledge bases, including a large upper-level ontology based on the merging of Cyc, SENSUS (Swartout et al, 1996) and other knowledge bases.

The three Challenge Problem scenarios that were set up in the first year of the HPKB project were:

- Crisis Management. This work linked up with another DARPA project (GENOA) which aimed to help intelligence analysts understand emerging international crises more rapidly. A scenario was developed in which hostilities between Saudi Arabia and Iran lead to the closure of the Strait of Hormuz (at the mouth of the Persian Gulf) to international shipping. Technology developers were then given the task of building systems to answer situation assessment questions, such as "Is Iran capable of firing upon tankers in the Strait of Hormuz?" and "What risks would Iran face in closing the Strait?" These are questions about motives, intents, risks, rewards and ramifications that may have multiple answers, and so significant common-sense reasoning is required to determine the most plausible answers to these questions.
- Movement Analysis. Given an idealised dynamic radar image showing vehicles moving in an area of several thousand square miles, the challenge problem was to identify types of vehicles (e.g. slower-moving dots travelling in convoys may be military vehicles) and to identify strategic military locations (e.g. places visited regularly by military vehicles).
- Workarounds. If a road or track is blocked by a large object, a crater, a minefield, or a blown bridge, there are a number of ways of "working around" that obstacle. The challenge problem was to calculate the swiftest way of working around an obstacle, given data about the nature of the obstacle, the terrain, and the availability and location of specialised assets such as portable bridges.

Each of these challenge problems required different AI technology to solve it. The Crisis Management scenario required text understanding and a detailed ontology and knowledge base in order to support something close to common sense reasoning. Movement Analysis required spatial

reasoning and fusion of multiple inputs. Workarounds planning required knowledge-based planning. For more details of each of these challenge problems, see the comprehensive paper on the HPKB project in AI Magazine (Cohen et al, 1998).

This paper focuses on the Workarounds challenge problem. Four technology developers provided solutions to this challenge problem: AIAI, Cycorp (with assistance from Teknowledge), George Mason University, and ISI (the Information Sciences Institute of the University of Southern California). This paper will describe the Workarounds planning challenge problem in more detail, describe each of the solutions in terms of their approaches to knowledge acquisition, ontology, and reasoning; and then present the results of the challenge problem evaluations as part of a critical evaluation of each approach.

## The Workaround Planning Challenge Problem

The Workarounds challenge problem required deciding how to circumvent or overcome obstacles to military traffic. Through knowledge acquisition performed in the course of the first year by AIAI and others, it became clear that there were six different ways of circumventing or overcoming obstacles:

- Bridging gaps;
- Filling gaps;
- Reducing obstacles until they are trafficable (or demolishing them completely);
- Finding alternate routes;
- Providing alternate transport (e.g. replacing a bridge over a river with a ferry);
- Clearing minefields.

Each of these classes of solution had several instances; for example, bridging a gap can be done with an AVLB (a light bridge carried on an armoured vehicle), a medium girder bridge, a Bailey bridge, or a ribbon bridge. Each solution instance has its own constraints; for example, AVLBs require both banks to be fairly level, and have a maximum usable length of about 20 metres, while ribbon bridges can only be used on water.

The planning requirements of the problem become clear when it is realised that each solution instance may require multiple steps (e.g. first transport the AVLB to the gap site, then set it up); the various constraints on solutions may require further steps (e.g. one bank must be bulldozed to make it sufficiently level before an AVLB can be set up, which requires getting a bulldozer to the site); and a full workarounds solution may make use of more than one solution instance (for example, the approach to a blown bridge may be mined, requiring both mine clearance and bridging; or a river in a valley may be crossed by bulldozing the banks on both sides to create two alternate routes to the river's edge, and then bridging the river with a ribbon bridge). Workarounds plans usually require less than 20 plan steps, so full-scale AI planning systems are not essential, but some planning capability is needed to solve this problem satisfactorily.

The technology developers were provided with information on the transportation link, the obstacle to be worked around, and key features of the local terrain; the units (tanks or trucks) that would be likely to use that transport route; and a detailed description of resources (such as Army engineering units) in the area that could be used to repair the damage. They were also provided with the written and diagrammatic results of knowledge acquisition sessions conducted over the course of the year. The expected outputs were a reconstitution schedule (an estimate of the capacity of the damaged

link over time, which requires a workaround plan), a time line of engineering actions needed to repair the link (if no alternate transport or alternate route is available, this will be the same as the workaround plan), and a set of assets required to effect the repair. From the point of view of the technology developers, this required considering alternate plans for repairing the link, calculating which plans were the most time-effective, and presenting the full details of these plans as outputs.

## AIAI's approach: Hierarchical Task Network planning within CYC

AIAI approached the workarounds planning problem as a part of the Teknowledge integration team, and with a background in developing the O-Plan AI planner (Tate et al, 1996), and in extracting the reasoning "primitives" from O-Plan in order to allow declarative planning within a standard knowledge-based systems 'shell' (Kingston et al, 1996). These reasoning primitives consist of permitted activities in the domain, represented as task formalisms (TFs); each TF states the preconditions of an activity, the effects of that activity, and (if applicable) the sub-activities of that activity. In the Search and Rescue system (Kingston et al 1996; Cottam et al, 1995), the TFs were represented as CLIPS objects, and CLIPS rules were used to determine which objects could be inserted into the plan given the current plan state, as well as identifying activities that could be done in order to achieve a plan state needed by another key activity.

In many planning tasks, the various activities are all sub-activities of a single top level activity that needs to be achieved; in this case, the system is said to be performing hierarchical task network (HTN) planning. Many well-known planning systems have used HTN planning, including Nonlin, SIPE, and O-Plan. For workarounds planning, the overall goal is to get to the other side of the obstacle, so this can be set up as the top level activity; the six solution classes described above then become the six possible sub-activities of that top level activity, and the various solution instances, and steps that comprise those solution instances, form sub-activities at various lower levels of decomposition.

AIAI offered to build a "proof of concept" workarounds planner in Cyc, using HTN planning. The aim was to represent TFs in Cyc, and then to use Cyc's default reasoning module (which uses backward chaining on axioms that represent implications to determine whether a query can be proved) to construct a full plan. By working entirely within Cyc's capabilities for common-sense reasoning, AIAI hoped to make the system robust to real-world changes and modifications; an example of such a change would be that the problem of crossing a river or a lake is greatly reduced if the air temperature is significantly below 0 degrees Celsius. In the event, few such issues arose in the challenge problems.

AIAI's work was also intended to define a usable, general purpose ontology of planning within Cyc, and to show how reasoning could be performed on constants conforming to that ontology.
One of the key original aims of Cyc was to develop a system that is capable of common-sense reasoning; its developers quickly discovered that ontological accuracy is an essential prerequisite of accurate common-sense reasoning. Cycorp has therefore developed an ontological approach which is divided into three levels: the upper level (where generic predicates such as GeographicalRegion and TransportationDevice are defined), the lower level (where domain-specific or problem-specific predicates are defined), and an intermediate level. In addition, constants in Cyc are differentiated on dimensions such as "stuff-like" versus "object-like" (based on whether identity is retained when the thing is divided up; so water is stuff-like, whereas a human being is object-like), and "always true" versus "sometimes true" (or, more generally, what the *persistence distribution* of a constant is). The effects of these dimensions are mitigated or

magnified by context; readers interested in these ramifications are referred to (Lenat98).

From the viewpoint of the workarounds challenge problem, it became clear that in order to represent TFs and plans in Cyc, a sub-ontology of planning terms needed to be introduced into Cyc. This ontology is shown in Figure 1.

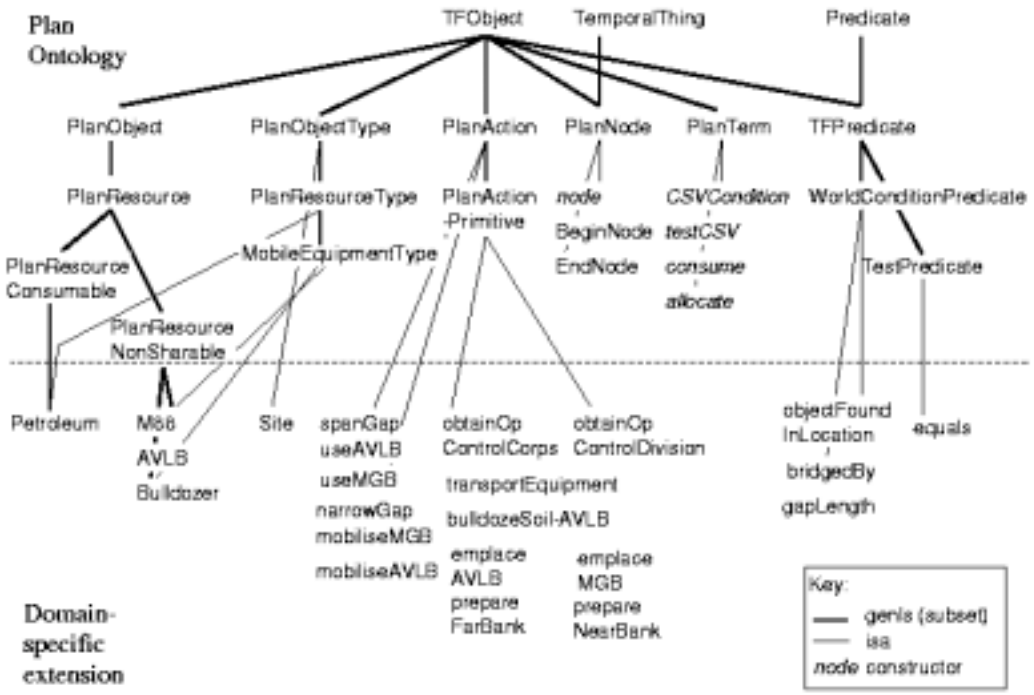



**Figure 1: Plan ontology in Cyc (from (Aitken & Kingston, 1999))**

These ontology definitions were used to create constants in Cyc that represented the TFs, the plan, the conditions of TFs, and the plan resources. For example, the following implication axiom (or 'rule') was created to test if a gap could be spanned by an AVLB:

```
(implies (isa ?AVLB AVLB)
  (potentialAction spanWithAVLB
    (conj (CSVCondition Site ?Site gapLength
                (testCSV lessThan (Meter 17.37)))
    (conj (CSVCondition Site ?Site riverBankMaxSlope
                (testCSV lessThan (Degree-UnitOfAngularMeasure 13.5)))
     (CSVCondition AVLB ?AVLB locationOf (testCSV equals ?Site))))
  (CSVCondition Site ?Site bridgedBy (testCSV equals ?AVLB)) ?I ?J)).
```

This rule states that if there is a site less than 17.37 metres wide, with banks sloped at no more than 13.5 degrees, and the AVLB is present at the site (according to the current plan state, represented by ?I), then a new node can be added to the current plan; at this node, the plan state (represented by ?J) considers the site to be successfully bridged by an AVLB.

As already indicated, the reasoning performed by AIAI's system was based on backward chaining through a set of these rules in order to generate a plan; in effect, Cyc is being 'tricked' into generating a plan when it thinks it is proving a goal. The version of Cyc which was used for this challenge problem (version 1.1) was able to generate plans using this method, but was not able to calculate the total time required for a plan; this calculation was done manually. More recent versions of Cyc have introduced this capability.

Knowledge acquisition for AIAI's planner was done by inspecting the published documents and data, and transforming this information into suitable rules and constants in Cyc. No supporting knowledge acquisition tool was used.

## TFS/Cycorp's approach: Re-using pre-defined ontology in a Lisp-based planner

Cycorp were also part of the Teknowledge integration team; indeed, they worked sufficiently closely with Teknowledge that the integration work was done by Teknowledge with significant input from Cycorp, while the challenge problems were tackled by Cycorp with significant resource from Teknowledge. The technology developers were therefore referred to as the TFS team.

Like AIAI, TFS used no tool that acquired knowledge directly from experts. They did, however, create a translator that automatically generated information about a workaround problem, both from the specific inputs supplied by the developers of the challenge problem, and from other sources. This process of creating axioms based on other sources is (informally) known as "knowledge slurping". A selection of the axioms generated by this translator is shown in Figure 2.

```
(widthOfObject River6 (Meter 16))
(lengthOfObject Bridge1 (Meter 33))
(spans-Bridgelike Bridge1 Crevice3)
(in-ContOpen River6 Crevice3)
(bordersOn Bridge1 Approach1)
(bordersOn Bridge1 Approach2)
(gapWithinPath Bridge1 Damage1)
(isa Damage1 GapInPathArtifact)
(lengthOfObject Damage1 (Meter 22))
(isa Approach1 GeographicalRegion)
(isa Approach2 GeographicalRegion)
(objectTypeFoundInLocation  Rubble Approach1)
(objectTypeFoundInLocation  Rubble Approach2)
```

**Figure 2: Some of the axioms "slurped" by Cyc**

This "knowledge slurping" approach proved to be an effective way of acquiring knowledge; several thousand axioms were acquired in the course of a few weeks.

A planner was implemented using SubL, a Lisp-like language that underlies Cyc, which made use of these axioms and of other axioms describing key terrain, the location of tanks, trucks and engineering units, and so on. The planner was divided into two modules: a "hypothesize" module

and a "test & repair" module. The first module hypothesizes a desired state, looks for the preconditions of that state, then it checks if these preconditions are satisfied; if not, then for each unsatisfied precondition, it recursively hypothesizes actions to fulfil that precondition. This approach relies on some simplifying assumptions about interactions among preconditions. The second module uses the "ask" function in CYC to check whether the preconditions for a desired state are *implied* by the given facts; if not, it explores sets of preconditions to *make* the preconditions true. This exploration covers both rules that can act, and rules that can change the known set of facts. Once all the sub-sub-sub-goals are proved, the workaround plan is considered to be feasible. In short, the planner used backward chaining to "prove" actions, which were then used to build plans, in a similar fashion to AIAI's HTN planner.

As far as ontology is concerned, TFS discovered that every single one of the axioms that were added to Cyc in order to solve the workaround challenge problem inherited – and used – some relevant axioms from Cyc's (pre-existing) upper ontology. It can therefore be claimed that Cyc's existing ontology contributed significantly to the reasoning required for the workarounds challenge problem. However, a fair amount of time was spent defining an intermediate ontology to represent concepts relevant to battle and a "battlespace", such as *spans-Bridgelike*; even more time was spent creating ontological terms that were specific to the challenge problem, such as "the weight of an unladen M88 tracked vehicle". These definitions had to be created before the relevant knowledge could be "slurped". The effort spent on ontology development had an adverse effect on the development of the planner, so the TFS planner that was used to tackle the challenge problems was not able to handle every aspect of the domain.

## ISI's approach: EXPECT and knowledge acquisition scripts

ISI, who were part of SAIC's integration team, took a different approach to the workarounds challenge problem. The focus of their work was on using ISI's EXPECT tool as a framework for ontology representation, as a knowledge base for knowledge acquisition support, and as a reasoning tool.

In EXPECT (Gil & Melz 1996; Swartout & Gil, 1996), both factual knowledge and problem-solving knowledge about a task are represented explicitly. This means that the system can access and reason about the representations of factual and problem-solving knowledge and about their interactions. Factual knowledge represents concepts, instances, relations, and the constraints among them. Knowledge is represented in Loom (MacGregor, 1999), a knowledge representation system of the KL-ONE family based on description logic. Every concept or class can have a definition that intensionally describes the set of objects that belong to that class; relations can also have definitions. Loom uses these definitions to produce a subsumption hierarchy that organises all the concepts according to a class/subclass relationship.

Problem solving knowledge is represented in a procedural language that is tightly integrated with the Loom representations. Sub-goals that arise during problem solving are solved by methods. Each method description specifies:

1. The goal that the method can achieve
2. The type of result that the method returns
3. The method body, containing the procedure that must be followed to achieve the method's goal.

Given these capabilities, and the availability of suitable ontologies in Loom, ISI were able to use

EXPECT to perform much of the reasoning needed for workarounds planning; some of the more complex aspects of the problem, such as action selection, required extension of EXPECT's capabilities with a Powerloom-based partial matcher. The ontologies used included the shared HPKB ontology, problem solving method ontologies, domain ontologies and situation information. While EXPECT needed extension to deal with some of the planning tasks, it was able to reason with the large ontologies needed without a significant effect on its performance; further efficiency gains were obtained by compiling the problem solvers.

Perhaps the biggest benefits of using EXPECT for the workarounds challenge problem arose from its ability to critique knowledge, and its use as a knowledge acquisition tool. By analysing the information needed by problem solving methods, EXPECT was able to detect over-generalisations in ontologies (e.g. use of the term "geographical region" where "city" was more appropriate), assumptions in ontologies that were unjustified in this domain (e.g. that objects can only be in one place), and missing information (e.g. unspecified bridge lengths). These capabilities enabled ISI to develop a consistent knowledge base more quickly than would otherwise have been the case.

EXPECT also made use of an approach known as *knowledge acquisition scripts* (Gil & Tallis, 1997) to assist with modification of problem solving knowledge. KA scripts are designed to help users modify the knowledge base in a structured manner; for example, if a problem solving method existed for calculating round trip time for ships, a KA script could assist the knowledge engineer in generalising that procedure to make it applicable for all vehicles. An example of a script (taken from (Gil & Tallis 1997)) can be seen in Figure 3.

```
Applicable when
       (a) A change has caused argument A of a goal G to become more
           general, resulting in goal G-new
       (b) Goal G was achieved by method M before A changed
       (c) G-new can be decomposed into disjunctive subgoals G-1 G-2
       (d) G1 is the same as G

Modification sequence
       CHOICE 1: Create new method M-new based on existing method
              (1) System proposes M as the existing method to be used as a
                  basis. User chooses M or another method.
              (2) System proposes a draft version of M-new that modifies A
                  to match G2. User can make any additional substitution
                  needed in the body of M-new.
              (3) User edits body of M-new if modifications other than
                  substitution are needed
       CHOICE 2: Create new method M-new from scratch

Description of what this KA-Script does:
       Create a method that achieves goal G2 based on method M.

Reasons why it is relevant to the current situation:
       Method M was used before to achieve goal G, which was generalised
       to become the unmatched goal G-new. M may be used to create a new
       method that achieves the other subgoal in this decomposition.
```

**Figure 3: A KA script to resolve error type: "Goal G-new cannot be matched"**

To summarise, EXPECT appears to be fully capable of reasoning about declarative information and ontologies for planning, as well as performing some knowledge-based planning tasks. The biggest contribution of EXPECT to the workarounds challenge problem was probably its aid for rapid development of knowledge bases, both through critiquing of ontologies and domain

knowledge, and through the use of KA scripts to assist modification of problem solving knowledge.

## GMU's approach: Collaborative Apprenticeship Multi-strategy Learning

The Learning Agents Lab at George Mason University provided the fourth solution to the workaround planning challenge problem. Their approach is based on the Disciple Toolkit (Tecuci 98; Tecuci et al. 99). The foundation of the Disciple Toolkit is an integration of apprenticeship and multi-strategy learning methods within the Plausible Version Space paradigm. This paradigm allows an expert to teach the agent in much the same way in which the expert would teach a human apprentice - by giving the agent specific examples of tasks and solutions, providing explanations of these solutions, and supervising the agent as it performs new tasks. During such interactions, the expert shares his expertise with the agent, which is continuously extending and improving its knowledge and performance abilities. These kinds of agent capabilities are achieved by a synergistic integration of several learning and knowledge acquisition methods: systematic elicitation of knowledge, empirical inductive learning from examples, learning from explanations, and learning by analogy and experimentation.

The interactions that take place within the Disciple Toolkit are illustrated in Figure 4. This diagram shows that the expert interacts with the system in four ways: eliciting knowledge, helping the system learn rules (from examples), refining and generalising the rules, and handling exceptions. From the examples and the rules, Disciple generates solutions; knowledge base refinement consists of critiquing these solutions, while exception handling deals with the inconsistencies in the knowledge base. The rationale for this approach is that it is easier for experts to update an ontology than to create an ontology; easier to supply examples than to supply rules; easier to understand a sentence in a formal language than to create such a sentence; and easier to give hints than to give explanations. The tight integration of various recognised techniques within Disciple creates a whole system that is arguably more useful and usable than the sum of its parts, and the ease of use of Disciple is expected to lead to rapid acquisition of knowledge from an expert.

For the workaround planning challenge problem, the editing and browsing modules were used to build an ontology describing bridges, river segments, army units and their equipment; the learning module was used to learn rules for destroyed bridges from concrete examples of workarounds and their explanations; and the refinement module was used to generalise or specialise the learned rules, based on the evaluation of workaround scenarios generated by the user. A hierarchical nonlinear planner based on task reduction was also developed and integrated into Disciple to solve workaround problems; an example of a learned task reduction rule can be seen in Figure 5. GMU were fortunate to have an ex-military man on their staff, who acted as the primary user of Disciple (i.e. as a domain expert) during acquisition of workarounds data.

It can be seen that the knowledge acquisition, ontology, and reasoning are much more tightly integrated in Disciple than in the other workaround planners. Disciple's primary aim is to be good at knowledge acquisition, although it is capable of ontology representation and reasoning as well.

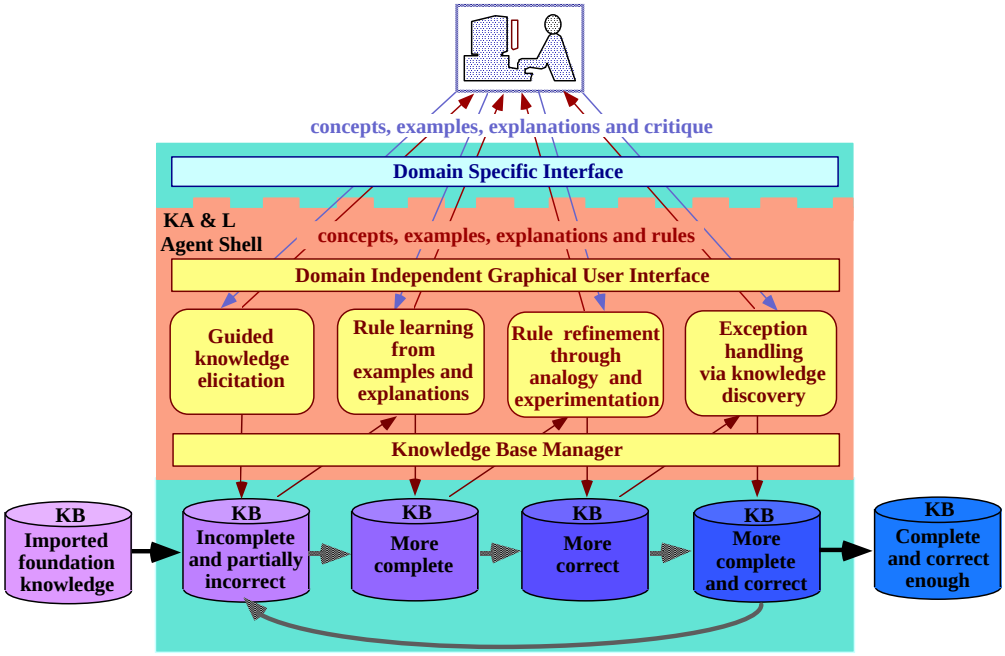



**Figure 4: Knowledge acquisition and learning processes in Disciple.**

# Challenge Problem Evaluation

The challenge problem evaluation was carried out June 1998. The format was as follows:

1. A set of questions was issued, that drew on the knowledge made available to everyone in the course of the year. Technology developers were given a short time to generate answers to these questions and mail them to the challenge problem developers (*Test Phase: the test)*
2. Technology developers were given a week or so to improve their systems in the light of their performance on the first set of test questions. They were then allowed to re-submit answers to the first set of test questions. (*Test Phase: the re-test)*
3. Further knowledge was issued that had not previously been available (specifically, knowledge about craters in roads, with definitions of craters and associated attributes). This tested the ability of the systems to enter new knowledge quickly. Five problems that made use of this knowledge were issued simultaneously, and technology developers answered as many as they could *(Modification phase: the test)*
4. After another week, a second set of test questions was issued that concerned working around craters, to test the systems' ability to reason on that new knowledge. Technology developers supplied answers to the five problems already given plus answers to five new problems (*Modification phase: the re- test*).

The answers to the questions were scored for scope (how many solutions, out of all those identified by the senior expert, were found) and score (how accurate the solutions were). Accuracy was scored on five dimensions: correctness of the overall time estimate; viability of the enumerated workaround options; correctness of solution steps provided for each viable option; correctness of temporal constraints among these steps; and appropriateness of engineering resources employed.

**IF** ***the task to accomplish is***

USE-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPARATION
FOR-BRIDGE ?O1
FOR-GAP-LENGTH ?N1
BY-UNIT ?O2
WITH-BR-EQ ?O3

***condition***

?O1 IS BRIDGE
?O2 IS MILITARY-UNIT
HAS-UPPER-ECHELON-EQUIPMENT ?O4
HAS-UPPER-ECHELON-EQUIPMENT ?O5
?O3 IS MILITARY-MOBILE-BRIDGE-EQ
?O4 IS BREACHING-EQ-SET
COMPONENT-TYPE ?O3
EQUIPMENT-OF ?O6
?O5 IS RUBBLE-CLEARING-EQ-SET
EQUIPMENT-OF ?O6
?O6 IS MILITARY-UNIT
LOCATED-AT ?O7
?O7 IS SITE
?N1 IS-IN (0 1000)

***except when***

?O2 HAS-EQUIPMENT ?O8
?O8 IS BREACHING-EQ-SET
COMPONENT-TYPE ?O3

***except when***

?O2 HAS-EQUIPMENT ?O9
?O9 IS RUBBLE-CLEARING-EQ-SET

**THEN** ***accomplish the subtasks***

?T1 OBTAIN-BRIDGE-AND-PREPARATION-EQUIPMENT-FROM-SAME-UNIT-THROUGH-UPPER-ECHELON
FOR-BR-EQ-SET ?O4
FOR-PREP-EQ-SET ?O5
FROM-UNIT ?O6
BY-UNIT ?O2
AT-LOCATION ?O1

?T2 INSTALL-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPEPARATION-AND-COLOCATED-BRIDGE-AND-PREPARATION-EQUIPMENT
FOR-BRIDGE ?O1
FOR-GAP-LENGTH ?N1
WITH-BR-EQ-SET ?O4
WITH-RC-EQ-SET ?O5
AT-LOCATION-EQ ?O7
FOR-UNIT ?O2

**Figure 5: A rule learned by the Disciple Toolkit for workaround planning**

# Results

The results are shown in Figures 6 and 7 (CREDIT ERIC JONES)**.** To help in understanding the diagrams, AIAI's results will be explained in more detail.

It would be helpful to specify the amount of time available to each technology developer; while

precise figures are not available, most of the technology developers were unable to work on the finalised version of the challenge problem until a couple of months before the testing phase (or in AIAI's case, a couple of weeks!) due to various administrative and co-ordination difficulties. While this time period was shorter than expected, it does provide a good estimate of how much knowledge can be captured and reasoned with in a short time period, which was one of the original aims of the HPKB program.

**Test Phase: Scope** AIAI's system could only answer questions related to bridging of gaps, due to the short development time. The system was only able to provide answers to two of the ten challenge problem questions. Its scope was therefore 20%.

**Test Phase: Score** AIAI's system initially lost accuracy marks for three reasons: in two cases, a possible solution option had been missed (again, this was a scope problem – the omitted solutions concerned types of bridges that AIAI's system didn't cover), and in one case, a calculation of the total time for a workaround was inaccurate. These problems were fixed, so that by the time of the re-test, the score had reached 100%.

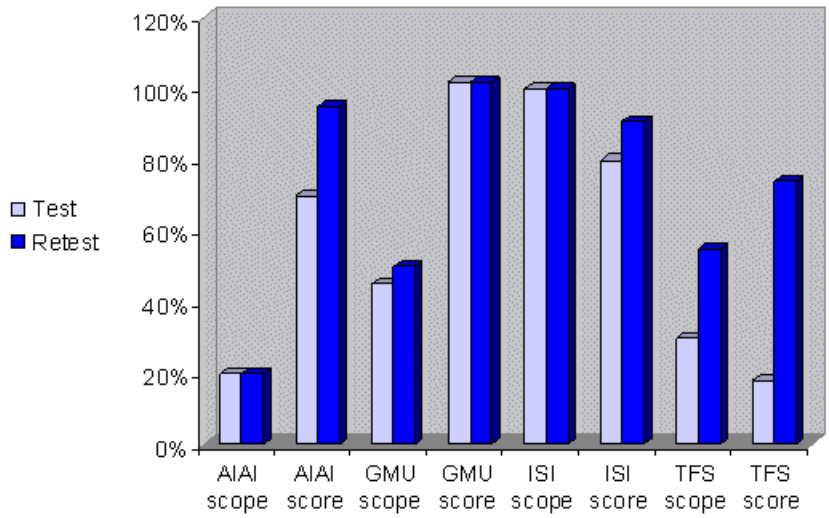



**Figure 6: Test phase: test and re-test scope and scores**

Overall, it can be seen that ISI and GMU significantly out-performed AIAI and Cycorp/ Teknowledge (TFS), with ISI having a very wide scope and GMU obtaining very high scores. Since an increase in scope is actually likely to lead to a decrease in scores (because the more questions that are answered, the more chance there is to make mistakes), the ISI and GMU systems should be considered "joint winners" of the evaluation. This trade-off is particularly clear in the modification phase, which should provide a truer reflection of the capability of each system, since each technology developer had exactly the same amount of time to make modifications to the system; as Figure 6 shows, as the scope of GMU's system went up, its accuracy slightly decreased.

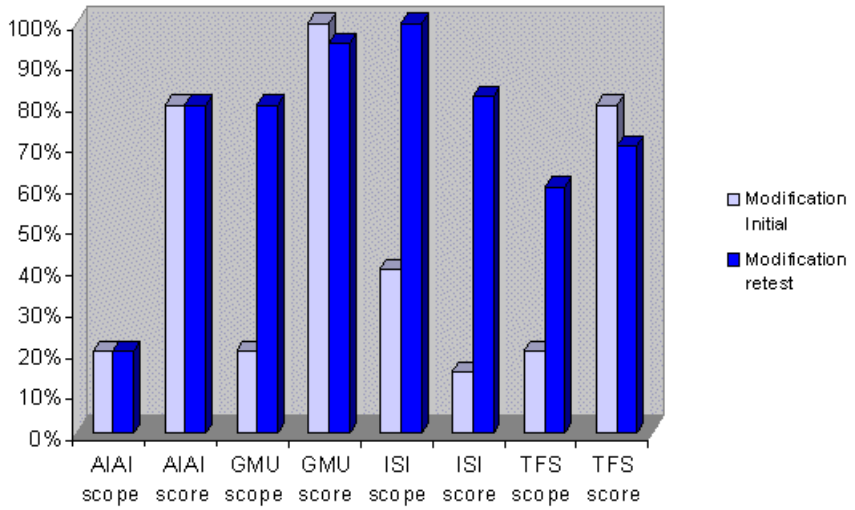



**Figure 7: Modification phase: test and re-test scope and scores**

The modification phase was also intended to demonstrate the capability of systems for rapid knowledge acquisition. AIAI continued to be handicapped by the narrowness of their previously acquired knowledge, but the other technology developers all achieved significant increases in scope during the modification week. This suggests that the goal of developing systems that can be used to build very large knowledge bases rapidly has been achieved, or (at least) can be achieved in this domain of knowledge.

## Strengths of each approach

What were the strengths of each approach?

- **AIAI:** the strength of this system lay in its well-justified ontology of planning, which provided the ability to achieve 100% accurate answers to challenge problem questions – i.e. to generate accurate and fully detailed plans. The fact that such an ontology could not only be built in Cyc, but could also be reasoned about, holds out hope that the reasoning capabilities of Cyc could be significantly extended by further definition of rich ontologies, and of corresponding problem solving methods; see (Sklavakis & Aitken, 1999) for an example of implementation of a problem solving method in Cyc.

- **TFS/Cycorp:** The greatest strength of this system was the ontology framework supplied by Cyc; this provided a noticeable speed-up in acquisition of domain knowledge through knowledge re-use. This effect is shown by the fact that only 35 new concepts had to be developed during the testing fortnight, compared against over 4000 pre-existing assertions that were accessed in one way or another, plus over 1000 assertions regarding vehicles and weapons that were re-used from the ontology developed for the Crisis Management challenge problem. This challenge problem therefore justified Cycorp's claim that a wide-ranging common-sense knowledge base, with a well-determined ontology, supports re-use of knowledge.

- **ISI:** The twin strengths of ISI's system were EXPECT, which can reason about ontologies and their contents as well as performing knowledge based reasoning, and the knowledge acquisition extensions that allowed rapid and accurate modification of the knowledge base (KA-Scripts). This project demonstrated that EXPECT can reason well with a large and rapidly growing ontology. It also demonstrated that EXPECT, with some assistance from Powerloom, was able to reason about the whole domain of knowledge, from the simplest deductions to the most complex calculations. Indeed, some of the calculations regarding Bailey bridges were so complicated that one member of ISI suggested to me (as a representative of the only UK organisation on the HPKB program) that only the British could have invented something like that!

- **GMU:** The rapid development of GMU's system highlighted its integrated knowledge acquisition tool as being its greatest strength. Over the fortnight of testing, GMU added 150 concepts, 100 tasks and 100 problem-solving rules to their knowledge base, representing a 20% increase in concepts, a 100% increase in tasks and a 100% increase in rules. This rate of knowledge acquisition suggests that GMU's system may indeed be able to achieve one of the Holy Grails of knowledge acquisition: rapid, accurate and direct knowledge entry by an expert without intervention from a knowledge engineer. GMU's system was also capable of reasoning about most aspects of the workarounds problem – indeed, it generated a few (correct) solutions that had not been considered by the expert.

## Summary

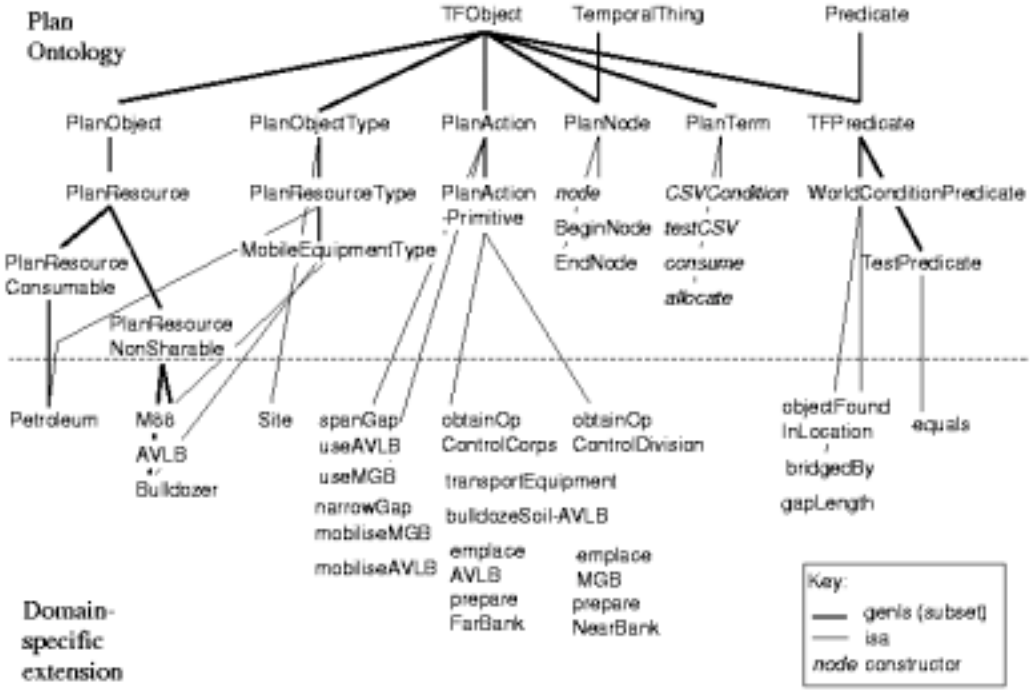
To summarise the lessons to be learned from this paper:

- Rapid development and implementation of very large knowledge bases for planning problems of medium complexity (i.e. plans with about 20 plan steps and 3-4 options for each step) is feasible with current AI technology.
- A knowledge acquisition tool is of great benefit in rapid knowledge base development, whether it is used by the expert to input knowledge or whether it transforms knowledge from other online sources.
- A well-organised and well-justified ontology contributes to both knowledge re-use and a high rate of acquisition of domain knowledge.
- Ontologies of problem solving, and problem solving methods, can provide a big improvement in accuracy of decision making.

## References


1. (Aitken & Kingston, 1999) Aitken S. and Kingston J. (1999). *Implementing a Workarounds Planner in Cyc: An HTN Approach*, Report to the DARPA/HPKB project
2. (Cohen et al, 1998) Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D. and Burke M. (1998). *The DARPA High-Performance Knowledge Bases Project*. AI Magazine, Winter 1998, 25-49.
3. (Cottam et al, 1995) Cottam H., Shadbolt N., Kingston J., Beck H. and Tate A. (1995) *Knowledge Level Planning in the Search and Rescue Domain*, in Research and Development in Expert Systems XII, proceedings of BCS Expert Systems'95, Cambridge, 11-13 December 1995.

4. (Gil & Melz, 1996) Gil Y. and Melz E. (1996) *Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition*. In Proceedings of the Thirteen National Conference on Artificial Intelligence (AAAI-96), Portland, Oregon, August 4-8 1996.
5. (Gil & Tallis, 1997) Gil Y. and Tallis M. (1997) *A Script-Based Approach to Modifying Knowledge Bases*. Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), Providence, RI, July 27-31 1997.
6. (Kingston et al, 1996) Kingston J., Shadbolt N. and Tate A. (1996). *CommonKADS Models for Knowledge Based Planning*, Proceedings of AAAI'96, Portland, Oregon, August 4-8 1996.
7. (Lenat, 1998) Lenat D. (1998). *The Dimensions of Context Space*. Available from http://www.cyc.com/publications.html
8. (Macgregor, 1999) Macgregor R. (1999). *Retrospective on Loom*. http://www.isi.edu/ isd/LOOM/papers/ macgregor/Loom_Retrospective.html
9. (Sklavakis & Aitken, 1999) Sklavakis D. and Aitken S. (1999*). Implementing Problem-solving Methods in Cyc*. In Proceedings of IJCAI-99, Stockholm, Sweden, August 1999. AAAI press.
10. (Swartout et al, 1996) Swartout B., Patil R., Knight K. and Russ T. (1996). *Towards Distributed Use of Large-Scale Ontologies*. In Proceedings of KAW '96, Banff, Canada, November 1996. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/ Banff_96_final_2.html
11. Swartout W.R. and Gil Y.(1996). *EXPECT: A User-Centered Environment for the Development and Adaptation of Knowledge-Based Planning Aids*. In Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, ed. Austin Tate. Menlo Park, Calif.: AAAI Press, 1996.
12. Tate A, Drabble B. and Dalton J. (1996*). O-Plan: A Knowledge-Based Planner and its Application to Logistics*. In Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, ed. Austin Tate. Menlo Park, Calif.: AAAI Press, 1996.
13. Tecuci G. (1998). *BUILDING INTELLIGENT AGENTS: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. San Diego: Academic Press, 1998.
14. Tecuci G., Boicu M., Wright K., Lee S.W., Marcu D., and Bowman M. (1999). *An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents.* In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), July 18-22, Orlando, Florida: AAAI Press, Menlo Park, CA. 1999.


Plan Ontology
TFObject
TemporalThing
Predicate
PlanObject
PlanObjectType
PlanAction
PlanNode
PlanTerm
TFPredicate
PlanResource
PlanResourceType
PlanAction -Primitive
node
BeginNode
EndNode
CSVCondition
testCSV
consume
allocate
WorldConditionPredicate
TestPredicate
PlanResource Consumable
MobileEquipmentType
PlanResource NonSharable
Petroleum
M66
AVLB
Bulldozer
Site
spanGap
useAVLB
useMGB
narrowGap
mobiliseMGB
mobiliseAVLB
obtainOp ControlCorps
transportEquipment
bulldozeSoil-AVLB
emplace AVLB
prepare FarBank
obtainOp ControlDivision
emplace MGB
prepare NearBank
objectFound InLocation
bridgedBy
gapLength
equals
Domain-specific extension
Key:
genls (subset)
isa
node constructor

(widthOfObject River6 (Meter 16))
(lengthOfObject Bridge1 (Meter 33))
(spans-Bridgelike Bridge1 Crevice3)
(in-ContOpen River6 Crevice3)
(bordersOn Bridge1 Approach1)
(bordersOn Bridge1 Approach2)
(gapWithinPath Bridge1 Damage1)
(isa Damage1 GapInPathArtifact)
(lengthOfObject Damage1 (Meter 22))
(isa Approach1 GeographicalRegion)
(isa Approach2 GeographicalRegion)
(objectTypeFoundInLocation Rubble Approach1)
(objectTypeFoundInLocation Rubble Approach2)

Applicable when

(e) A change has caused argument A of a goal G to become more general, resulting in goal G-new
(f) Goal G was achieved by method M before A changed
(g) G-new can be decomposed into disjunctive subgoals G-1 G-2
(h) G1 is the same as G

Modification sequence

CHOICE 1: Create new method M-new based on existing method

(4) System proposes M as the existing method to be used as a basis. User chooses M or another method.
(5) System proposes a draft version of M-new that modifies A to match G2. User can make any additional substitution needed in the body of M-new.
(6) User edits body of M-new if modifications other than substitution are needed

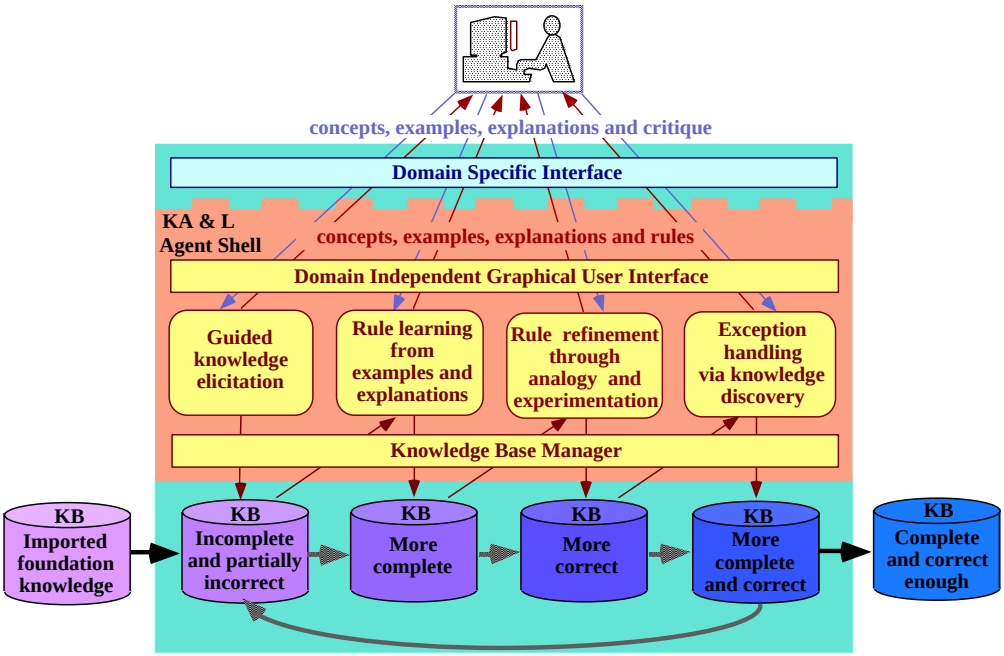
CHOICE 2: Create new method M-new from scratch

Description of what this KA-Script does:

Create a method that achieves goal G2 based on method M.

Reasons why it is relevant to the current situation:

Method M was used before to achieve goal G, which was generalised to become the unmatched goal G-new. M may be used to create a new method that achieves the other subgoal in this decomposition.


concepts, examples, explanations and critique
Domain Specific Interface
KA & L
Agent Shell
concepts, examples, explanations and rules
Domain Independent Graphical User Interface
Guided knowledge elicitation
Rule learning from examples and explanations
Rule refinement through analogy and experimentation
Exception handling via knowledge discovery
Knowledge Base Manager
KB
Imported foundation knowledge
KB
Incomplete and partially incorrect
KB
More complete
KB
More correct
KB
More complete and correct
KB
Complete and correct enough

**IF** ***the task to accomplish is***

USE-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPARATION
FOR-BRIDGE ?O1
FOR-GAP-LENGTH ?N1
BY-UNIT ?O2
WITH-BR-EQ ?O3

***condition***

?O1 IS BRIDGE
?O2 IS MILITARY-UNIT
HAS-UPPER-ECHELON-EQUIPMENT ?O4
HAS-UPPER-ECHELON-EQUIPMENT ?O5
?O3 IS MILITARY-MOBILE-BRIDGE-EQ
?O4 IS BREACHING-EQ-SET
COMPONENT-TYPE ?O3
EQUIPMENT-OF ?O6
?O5 IS RUBBLE-CLEARING-EQ-SET
EQUIPMENT-OF ?O6
?O6 IS MILITARY-UNIT
LOCATED-AT ?O7
?O7 IS SITE
?N1 IS-IN (0 1000)

***except when***

?O2 HAS-EQUIPMENT ?O8
?O8 IS BREACHING-EQ-SET
COMPONENT-TYPE ?O3

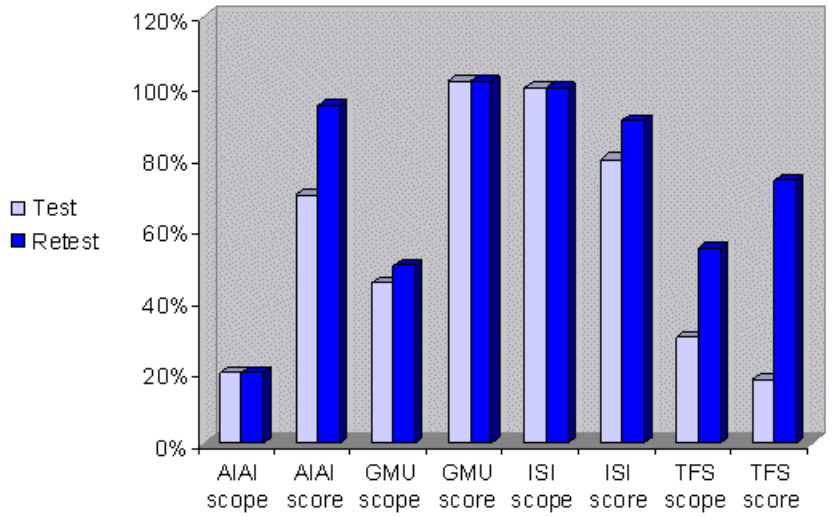
***except when***

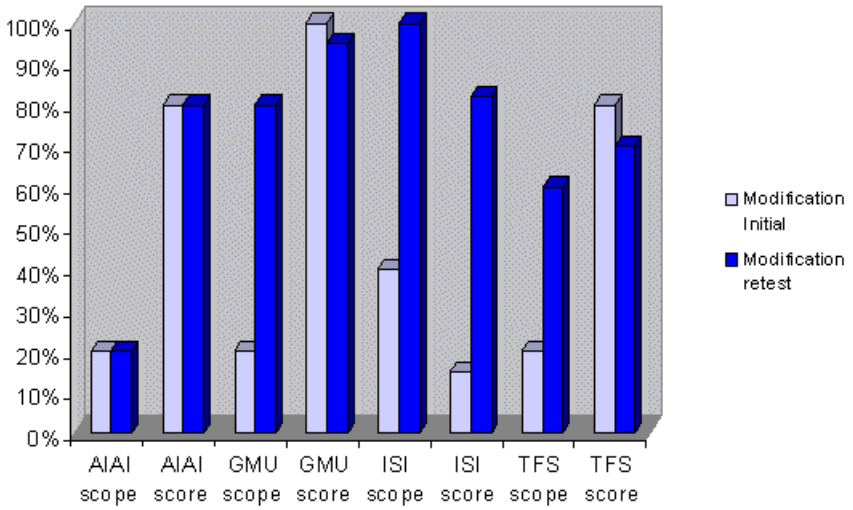
?O2 HAS-EQUIPMENT ?O9
?O9 IS RUBBLE-CLEARING-EQ-SET

**THEN** ***accomplish the subtasks***

?T1 OBTAIN-BRIDGE-AND-PREPARATION-EQUIPMENT-FROM-SAME-UNIT-THROUGH-UPPER-ECHELON
FOR-BR-EQ-SET ?O4
FOR-PREP-EQ-SET ?O5
FROM-UNIT ?O6
BY-UNIT ?O2
AT-LOCATION ?O1

?T2 INSTALL-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPEPARATION-AND-COLOCATED-BRIDGE-AND-PREPARATION-EQUIPMENT
FOR-BRIDGE ?O1
FOR-GAP-LENGTH ?N1
WITH-BR-EQ-SET ?O4
WITH-RC-EQ-SET ?O5
AT-LOCATION-EQ ?O7
FOR-UNIT ?O2


120%
100%
80%
60%
40%
20%
0%
Test
Retest
AIAI scope
AIAI score
GMU scope
GMU score
ISI scope
ISI score
TFS scope
TFS score


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
AIAI scope
AIAI score
GMU scope
GMU score
ISI scope
ISI score
TFS scope
TFS score
Modification Initial
Modification retest