In "Developing the Semantic Web", D. Fensel, J. Hendler, H. Lieberman, and W. Whalster (Eds.), MIT Press. Forthcoming.

Knowledge Mobility: Semantics for the Web as a White Knight for Knowledge-Based Systems

Yolanda Gil, USC/Information Sciences Institute

"No Man is an island
Entire of itself.
Every man is a piece of the continent...."

-- John Donne (1572-1631)

1 Introduction

One of the challenges for knowledge-based systems is interoperation with other software, intelligent or not. In recent years, our research group has participated in various integration efforts, where interoperation was supported through translation techniques, mostly at the syntactic level and more recently through ontology-based approaches. In this article, I argue that the interoperation challenge will not be met with current approaches, since they entail trapping knowledge into formal representations that are seldom shareable and often hard to translate, seriously impairing its mobility. I propose an approach to develop knowledge bases that captures at different levels of formality and specificity how each piece of knowledge in the system was derived from original sources, which are often Web sources. If a knowledge base contains a trace of information about how each piece of knowledge was defined, it will be possible to develop interoperation tools that take advantage of this information. The contents of knowledge bases will be more mobile and no longer be confined within a formalism. The Semantic Web will provide an ideal framework for developing knowledge bases in this fashion. We are investigating these issues in the context of TRELLIS, an interactive tool to elicit from users the rationale for choices and decisions as they analyze information that may be complementary or contradictory. Starting from raw information sources, most of them originating on the Web, users are able to specify connections between selected portions of those sources. These connections are initially very high level and informal, and our ultimate goal is to develop a system that will help users to formalize them further.

2 The Need for Knowledge Mobility

This section argues the need for knowledge mobility from two viewpoints. First, knowledge-based systems need to be built to facilitate interoperability and thus the knowledge they contain needs to be more accessible to external systems. Second, our knowledge bases capture knowledge in such a way that it is very hard to get out or translate what they contain.

2.1 No Knowledge Base is an Island

Knowledge-based systems are no longer built to function in isolation. Every application that we have built with the EXPECT architecture [Blyhe et al., 2001; Kim and Gil 2000; Gil and Tallis, 1997; Gil and Melz, 1996] in recent years has been integrated within a larger end-to-end system in order to provide the overall functionality required. This section describes several integration efforts that illustrate important challenges in terms of knowledge mobility:

- **Significant differences in representation languages**. We have found on several occasions that different systems use representations and languages that adequately support certain functionality or problem solving capabilities. Requiring that all the systems in the integration adopt the same representation may be an option in some cases, but many times doing this may be technically challenging or simply unfeasible in practice. An alternative approach that ends up being more acceptable in practice is to allow each problem solver to use different languages and representations and then have ways to map back and forth between the two.
- **Discrepancies in modeling styles.** Even when different knowledge bases are developed in similarly expressive languages, different knowledge engineers practice different modeling styles and approaches. Translators between different languages help import knowledge bases written by different developers, but they only translate one syntactic expression into another and do not address deeper differences in representational styles. The rationale for modeling the original knowledge as a certain suite of expressions is never captured in the knowledge base.

- **Maintaining consistency among related pieces of knowledge**. A single domain description may be modeled piecemeal in separate parts of the knowledge base to reflect different views or functions of the description. The individual pieces may not be explicitly related in the knowledge base, especially when different pieces are only used by different components of the overall integration. In general, very few of the many and varied relations between individual pieces of knowledge are captured in the resulting knowledge bases.

2.1.1 Plan Generation and Evaluation

Figure 1 shows the architecture of a knowledge-based system for Workarounds Analysis that we developed for the DARPA High Performance Knowledge Bases Battlespace Challenge Problem [Cohen et al., 1999]. This system estimates the delay caused to enemy forces when an obstacle is targeted (e.g., a bridge is destroyed or a road is mined), by reasoning about how they could bypass, breach, or improve the obstacle in order to continue movement (e.g., by repairing a damaged bridge or installing a temporary one, fording a river, removing mines, choosing an alternative route). Several general ontologies relevant to battlespace (e.g., military units, vehicles), anchored on the HPKB upper ontology, were used and augmented with ontologies needed to reason about workarounds (e.g., engineering equipment). These ontologies were represented in LOOM, a knowledge representation and reasoning system based on description logic [MacGregor 1991]. The system also included two main problem solvers. A Course of Action Generation problem solver creates several alternative solutions to the problem.

Each solution lists several engineering actions for that workaround (e.g., deslope the banks of the river, then install a temporary bridge, etc.), includes information about the resources used (e.g., what kind of earthmoving equipment or bridge is used), and states temporal constraints among the individual actions to indicate which can be executed in parallel. This problem solver was developed using a state-of-the-art planner [Veloso et al., 1995]. A Temporal Estimation/Assessment problem solver evaluates each of the alternatives and selects one as the most likely choice for an enemy workaround. This problem solver was developed in EXPECT. It used the information about engineering equipment and techniques included in the ontologies as well as several dozen problem solving methods to estimate how long it would take to carry out a given workaround plan generated by the Course of Action Generation problem solver. This system was the only one to attempt full coverage in this DARPA Challenge Problem, and demonstrated best performance at this task.

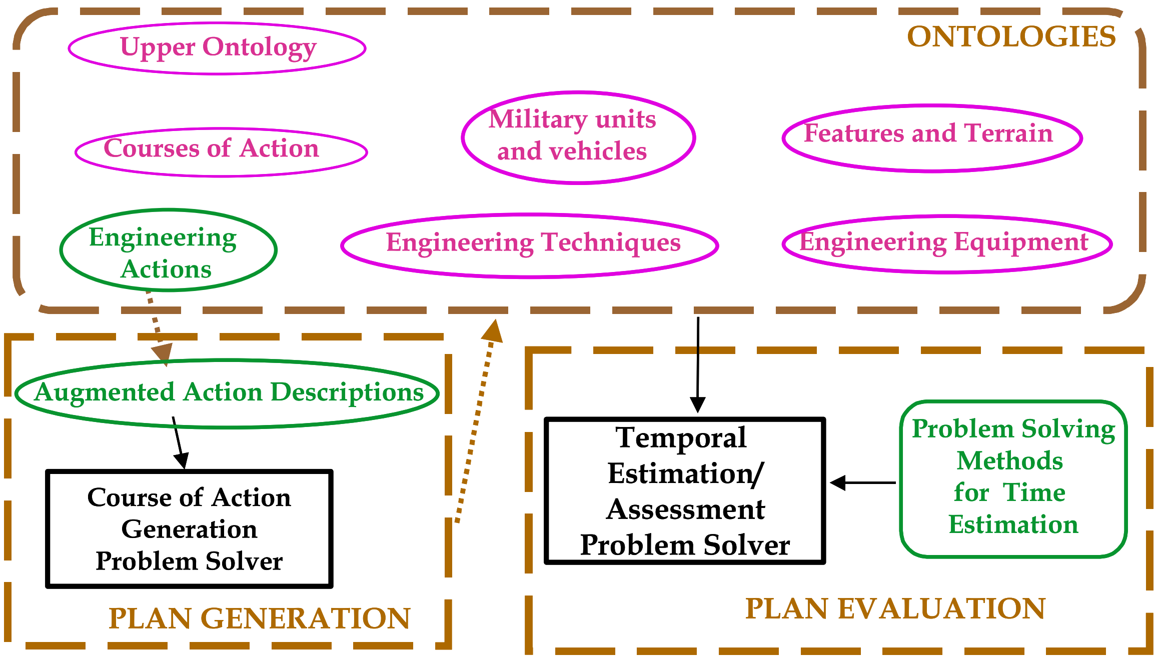

Upper Ontology
ONTOLOGIES
Courses of Action
Military units and vehicles
Features and Terrain
Engineering Actions
Engineering Techniques
Engineering Equipment
Augmented Action Descriptions
Course of Action Generation Problem Solver
PLAN GENERATION
Temporal Estimation/ Assessment Problem Solver
Problem Solving Methods for Time Estimation
PLAN EVALUATION

Figure 1 A Knowledge-Based System for Plan Generation and Evaluation

One of the challenges for integration illustrated by this system is differences in representation. We found a huge gap between the representations used in the state-of-the-art knowledge representation and reasoning systems and those of a special-purpose reasoner, in our case a state-of-the-art plan generation system. The ontology of engineering actions shown in Figure 1 was sufficient to support the temporal estimation problem solver, but it needed to be significantly augmented by knowledge engineers to develop the action descriptions required by the planner. These descriptions could not be represented declaratively in the ontology, and thus had to be separately developed and maintained. Conversely, the planner generated very detailed plans that included many causal and temporal links among the actions, as well as steps and bindings that were helpful for plan generation but had no counterpart in an engineering plan. Thus, only portions of the final plan were incorporated back in the knowledge base. It is important to notice that the issue was not a difference in content, but in the expressivity of the language regarding planning-specific knowledge. Although it may be possible (though certainly non-trivial) to coerce the planner's representations into the ontology, it is not clear that it is a superior representation for that kind of knowledge.

Another problem was maintaining consistency in the knowledge base. We needed to keep three different parts of the knowledge base closely aligned: the ontology engineering actions, the ontology of engineering techniques, the methods for time estimation, and the planner's augmented engineering actions. Since each of these ontologies required different expertise, different knowledge engineers were resposible for each of them. The EXPECT knowledge acquisition tools [Kim and Gil, 1999] were very useful in pointing out inconsistencies between the methods and some of the ontologies, but the planner's actions were not accessible to EXPECT. Consistency across the different parts of the knowledge base had to be maintained largely by hand. Some of the performance problems during the evaluation of this system were traced to these kinds of inconsistencies.

2.1.2 Mixed-Initiative Plan Generation

Another active area of research within our project is mixed-initiative tools to help users create strategic plans, in particular in air campaign planning. In order to plan what air operations will be conducted in a campaign, military planners use a strategies-to-task methodology where low-level tasks are derived from higher level national and campaign goals. For example, a campaign objective like attaining air superiority results in an operational subobjective of suppressing enemy air sorties, which in turn creates an operational task of damaging key airbase support facilities. Ultimately, these objectives are turned into missions stating which specific aircraft, crews, and airfields will be used to accomplish the lower-level tasks. At the higher, strategic levels of planning users prefer to maintain control of the planning process and specify the objectives themselves, leaving automatic plan generation to the lower levels of the planning process. A series of plan editors were built to allow a user to define objectives and decompose them into subobjectives, possibly invoking automated plan generation tools to flush out the plan at the lower levels. Because the plan creation process is mostly manual (at least at the higher levels), it is prone to error. This is aggravated by the size of the plans (several hundreds of interdependent objectives and tasks) and by the number of different people involved in its creation. In order to help users detect potential problems as early as possible in the planning process, we developed INSPECT, a knowledge-based system built with EXPECT that analyzes a manually created air campaign plan and checks for commonly occurring plan flaws, including incompleteness, problems with plan structure, and unfeasibility due to lack of resources. For example, INSPECT would point out that if one of the objectives in the plan is to gain air superiority over a certain area then there is a requirement for special facilities for storing JPTS fuel that is currently not taken into account. Without INSPECT, this problem might only be found when a logistics expert checks the plan, a day or more after the execution of the overall plan is started. INSPECT reasons about the kinds of objectives in the plan, notices that gaining air superiority requires providing reconnaissance missions, which are typically flown in stealth aircraft that need specific kinds of fuel (e.g., JPTS) that need to be stored in special facilities. INSPECT was very successfully received by users, since it always found unintentional errors in the plans created by INSPECT's very own designers.

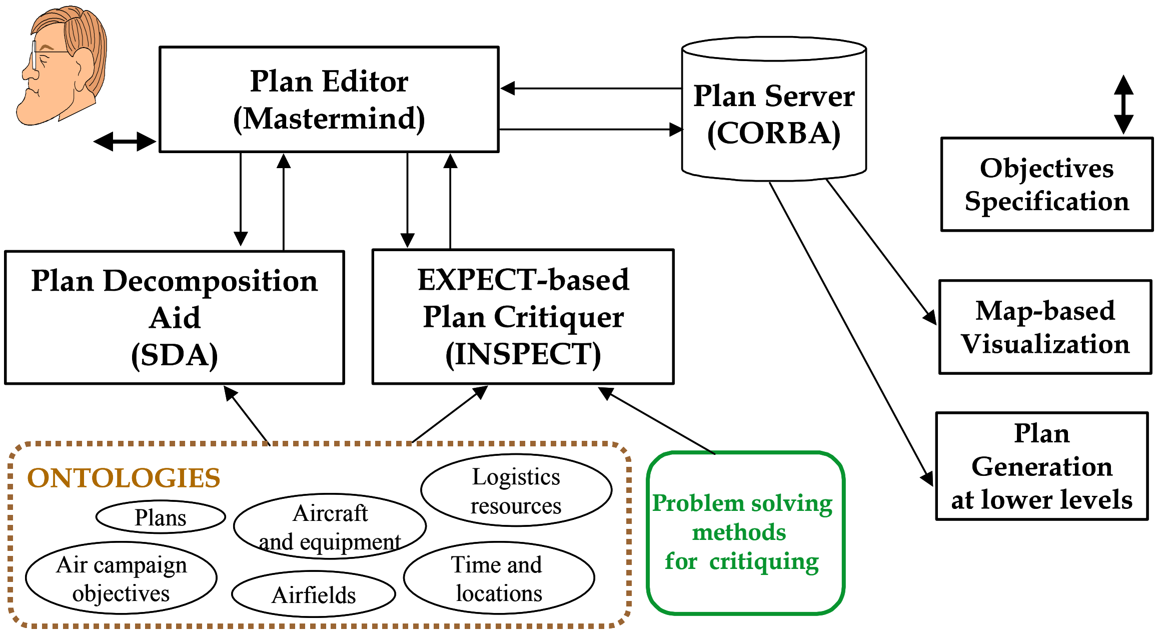


Figure 2 An Integration Framework for a Plan Critiquer

INSPECT was integrated in several architectures of very different nature. Figure 2 shows an integration of INSPECT with a few knowledge-based tools within a pre-existing software architecture that was CORBA-based. There was a plan server that contained not only air campaign plans but also land and maritime activities in joint operations. The plan server was not amenable to any major changes, so many of the plan details and constraints that were used by the knowledge-based tools never made it there. Figure 2 also shows the different ontologies in INSPECT's knowledge base, whose integration and development is described elsewhere [Valente et al., 1999].

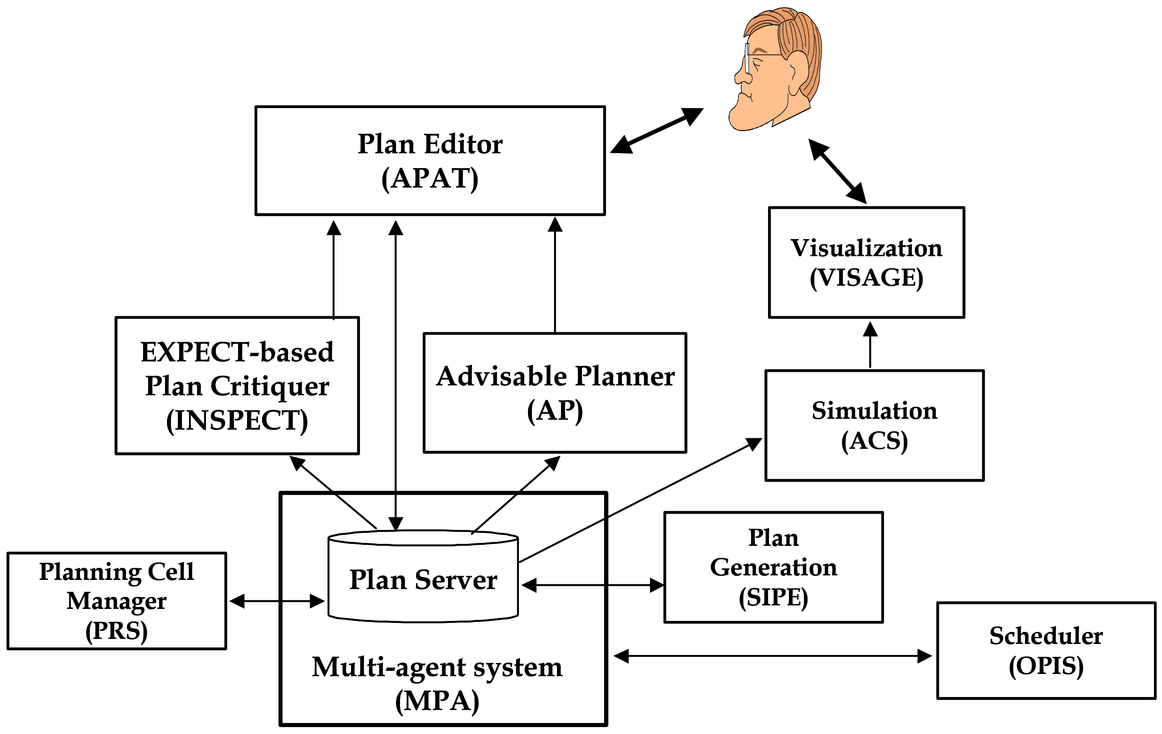


Figure 3 Integrating a Plan Critiquer in a Multi-Agent System

Figure 3 shows another integration, this time with several other planning aids within a multi-agent planning architecture.   This integration is another illustration of the representational mismatch among different systems. The plan server in this architecture contained rich representations of plans regarding their hierarchical decomposition, causal and temporal dependencies, and overall planning process management. It lacked, however, rich representations of objectives that were needed by the plan editor and by INSPECT in order to analyze the requirements of each objective, and as a result these representations were not incorporated in the plan server and were not available to any other tool. We suspect that other modules found similar discrepancies between their particular representation formalisms and the plan server.

2.1.3 Critiquing Courses of Action

Another important integration effort was to develop an end-to-end system to help users develop military courses of action, depicted in Figure 4. Users input a sketch on a map to specify the activities of each force and a statement of the objectives of the course of action. Several critiquers were developed by different groups to analyze different aspects of the course of action in order to give users feedback about possible problems in their design. We developed a critiquer with EXPECT to check problems related to plan structure and use of resources. The other critiquers had complementary capabilities since each of them addressed different kinds of critiques.

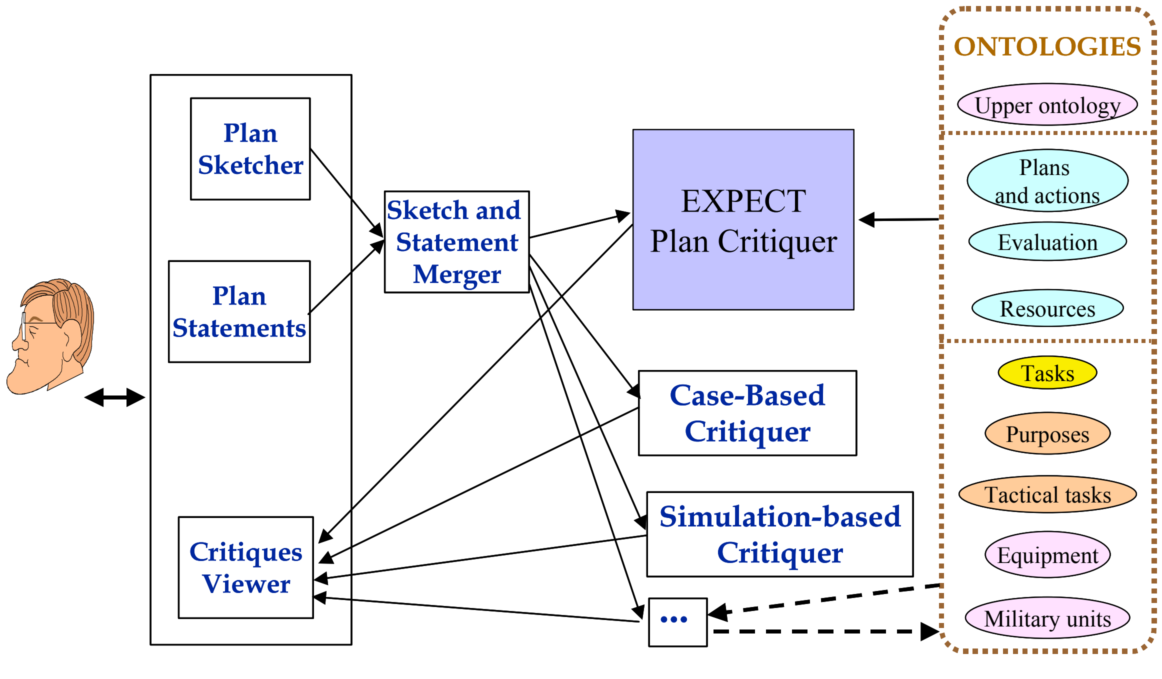


Figure 4 Critiquing Courses of Action

The application-specific ontologies used by our system were developed by other groups involved in this effort. Importing these ontologies into our knowledge base was a significant effort, as it was transforming the input course of action into a format consistent with our representation.

The simpler translation issues arise in translating original ontologies in MELD or KIF into LOOM, although sometimes mapping different names was necessary (e.g., to map "TranslationLocationChange" into "Move"). More complex translations are required because of deeper representational mismatches. For example, some expressions used to describe the course of action stated that a certain task had protect as a purpose when there was something to protect, while our representation created an instance of protect as the purpose of the task. At times, some of the decomposition structure of the course of action was implicit in the order of the statements, and we had to reconstruct that structure so that it would be available in our knowledge base. We used the OntoMorph rewrite engine to support these more complex mappings, which are described in more detail in [Chalupsky, 2000]. In essence, supporting this integration was challenging even though the languages used by the different systems were comparable in terms of their expressivity. The differences in modeling approaches and representations were quite significant and required a complex translation system like OntoMorph.

2.1.4 Translation among Intelligent Agents

As part of the initial stages of a new project to develop ontology-based translation services for multi-agent systems, we conducted a study of an existing multi-agent system (not developed by us) to determine the requirements of such a translation service [Gil, 2000]. This multi-agent system was developed to support non-combatant evacuation operations, and was used to organize groups of helicopters to take people out of a dangerous area and into a safe heaven. The agents participating included a suite of interface agents that interact with users to get the initial helicopter team composition and critical locations as well as unexpected threats to the mission (e.g., explosions), a route planner, a threat (SAM site) finder, and helicopter agents that can work in teams in a simulated environment. All of these agents were initially developed in radically different contexts, and originally did not even share the application domain.

In a typical run, a few thousand messages are exchanged among the agents. We analyzed the content of messages that are representative of the communications regarding significant events and activities during the mission. Some of the findings include:

- Syntactic translators are relatively easy to build and are often readily available for many formats and languages. Yet, in the years ahead we will see large communities of heterogeneous agents where pairwise translations will become impractical. Each agent will have to advertise its capabilities and requirements in one (or just a few) formats, and rely on translators and mediators to map those requirements into those of other agents. In this particular system, syntactic translations were often required even though the developers often agreed to message formats that would minimize these needs. Even when the same kind of information was transmitted different formats were used. For example, the lat-long coordinates sometimes use the convention of an "X" and "Y" label, in other cases the coordinates were preceded by "-lat" and "-long" (this happens when the critical locations are given to helicopters by the interface agents), or just ordered within parentheses (this is the format used by the threat finder to send threat sites to the helicopter agents). A complex case of syntactic mapping was required because the threat finder takes queries in SQL format.

- Mismatches across representations were numerous and often due to modeling with different granularities and levels of abstraction. For example, the route generated by the route planner needed to be transformed to a coarser-grained route in order to be useful to the helicopter agents. The route planner sends a route composed of points that are 9 meters apart. The helicopters are tasked one route segment at a time, and when a helicopter is given a route that is of a small size (like 9 meters) it will say that the route is achieved without even taking off. Dropping details from a plan is often needed when planning agents exchange plans. This may involve dropping steps, dropping parts of steps, or may require a completely different plan altogether. Another example of the need for such transformations is that the helicopters consider routes to be composed of point-to-point segments, and the threat finder needed to be given a rectangular area to initiate its search. The route segment was made into the diagonal of the rectangular area, and as a result there were threats returned that were in the area but not very close to the route itself. Similarly, the interface agents specified helicopter landing zones as areas, while the helicopter agents take only a point as landing zone.

- Mapping among different representations may be complex, and invocations of third-party agents may be necessary. The route planner and the threat finder use a lat-long coordinate system, the helicopters use an "x,y, cell" format, and the interface agents use UTM coordinates. Although it is possible to build ontology-based translators to mediate these communications, it may be more practical to use the ontologies to simply detect the mismatch and then invoke a third party to do the conversion since many such conversion systems are already available. Other cases when invocation of other agents would be useful arise when an agent can fulfill a request but needs information that the requesting agent may not have. For example, the route planner required a map URL, the interface agents require a range around a location, etc.

This particular multi-agent system illustrates the challenges in supporting interoperation among intelligent systems, even in relative small scale and numbers. It would not be simple to replace a subset of the agents involved by other agents of equivalent functionality, which could probably not be done without some additional changes and adaptations in the original remaining agents.

2.1.5 Summary

Current efforts in the knowledge-based systems community aim to support interoperability through shared ontologies and diverse translation tools. Drawing from past experiences within our own research project, this section illustrates several points:

- Shared ontologies do not address all integration issues. Content can be agreed upon, but representational needs are determined by the functionality required of the individual problem solvers and components within the system.
- Maintaining consistency in the knowledge used by several components is a great challenge. Some components may have some consistency checking facilities, but that is often not the case with all the components in the overall system.
- Differences in modeling methodologies make translation an arbitrarily complex task. Syntactic translation and mappings are adequate only when such differences are minor.

To paraphrase John Donne, no knowledge based-system is an island. There is an increasing demand to have individually developed intelligent systems become part of larger end-to-end application, and current paths to integration are not able to support interoperability appropriately.

2.2 Educating Knowledge Bases

Large knowledge bases contain a wealth of information, and yet browsing through them often leaves an uneasy feeling that one has to take the developer's word for why certain things are represented in certain ways, why other things were not represented at all, and where might we find a piece of related information that we know is related under some context. Although the languages that we use are quite expressive, they *still force knowledge into a straightjacket*: whatever fits the language will be represented and anything else will be left out. Many other things are also left out, but for other reasons such as available time and resources or perhaps lack of detailed understanding of some aspects of the knowledge being specified.

Furthermore, *knowledge ends up represented piecemeal,* compartmentalized in whatever expressions the modeling language supports. Many of the connections between different pieces of knowledge are never stated, nor can they be derived by the system given what it knows. We see no value in representing redundant information or alternative ways to deduce the same facts: if the system can derive something in one way that may be more than sufficient.

Knowledge base developers may consult many sources presenting contradictory or complementary information, analyze the different implications of each alternative belief, and decide what and how to model the knowledge. In essence, developers often capture in the knowledge base only their final beliefs about some body of knowledge. *The rationale for modeling the knowledge the way it appears in the knowledge base is not* *captured* declaratively. Only consistent and complete information is captured. No indication of inconsistent but possible statements is added to the knowledge base.

As Minsky argues it [Minsky, 1970]:

> There is a real conflict between the logician's goal and the educator's. The logician wants to minimize the variety of ideas, and does not mind a long thin path. The educator (rightly) wants to make the paths short and does not mind -- in fact, prefers -- connections to many other ideas.

Knowledge base developers seem to prefer the role of logicians rather than seeing themselves as educators of intelligent systems.

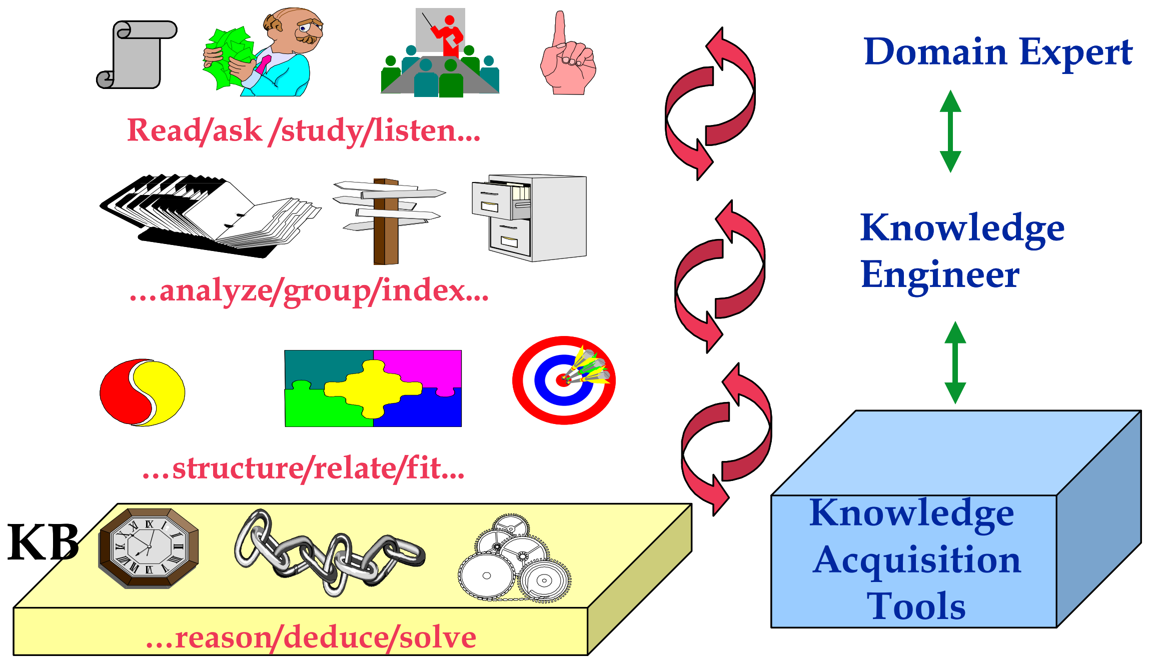

Read/ask /study/listen...
...analyze/group/index...
...structure/relate/fit...
KB
...reason/deduce/solve
Domain Expert
Knowledge Engineer
Knowledge Acquisition Tools

Figure 5 How Knowledge Bases are Built Today

Figure 5 illustrates the limited kinds of knowledge that are captured in the final knowledge base. Developers (at least non-experts) start by consulting manuals and tutorial material, asking questions, and requesting clarifications. Their main task is to analyze and different information sources, grouping information, indexing related definitions and terms, and gathering as much raw material as possible in order to understand what needs to be represented and how. Next, they organize the information in semi-formal ways by structuring it in tables, itemized lists, and detecting opposite and complementary descriptions. Finally, they build the knowledge base itself by turning the refined results of this analysis into formal expressions that fit in the particular knowledge representation language used. Whatever documentation ends up in the knowledge base will be the only trace left of all the design and analysis process that was done to create it. None of the documentation is captured in declarative languages. The rationale of the knowledge base design is lost: the original sources, the analysis and structuring of the knowledge therein, and the tradeoffs that were considering before deciding on the final formalization. As a result:

- It is hard to extend knowledge bases. When the knowledge base needs to be extended or updated, the rationale for their design is lost and needs to be at least partially reconstructed. The knowledge sources are no longer readily available and may need to be accessed.
- It is hard to reuse knowledge contained in knowledge bases. While it is the case that entire knowledge bases can be reused and incorporated into new systems, it is harder to extract only relevant portions of them that are appropriate in the new application. Parts of the knowledge base may be too inaccurate for the new task, or may need to be modeled in a different way to take into account relevant aspects of the new application.

In summary, knowledge has a very modest degree of mobility once it is incorporated into our current systems. Some researchers are creating shared upper ontologies that can serve as a common substrate for the more detailed knowledge in specific knowledge bases, thus facilitating interoperation and reuse. Some have argued that the brittleness in our knowledge bases can be addressed by providing common-sense and other background knowledge to these systems. These approaches may be part of the solution, but it will not address some of the issues brought up here. Current intelligent systems are hard to integrate, maintain, and understand because their *knowledge bases have not been truly educated on the topics they are supposed to know about.*

3 A New Generation of Knowledge Bases: Resilient Hyper Knowledge Bases

In order to empower intelligent systems, we believe we need to allow them access to the roots and rationale of the knowledge they contain. Furthermore, the knowledge base should not just contain a body of formalized expressions; rather, we should extend our view of a knowledge base to include a variety of formats and representations as well as alternative renderings of the same (or related) knowledge. They should include as many connections as possible, as stated in the original sources and as they result from the analysis of the knowledge base developer. This approach would create a new generation of knowledge bases: Resilient Hyper Knowledge Bases (RHKBs), that will be more resilient to change and reuse and will be heavier in connections and hyperlinks.

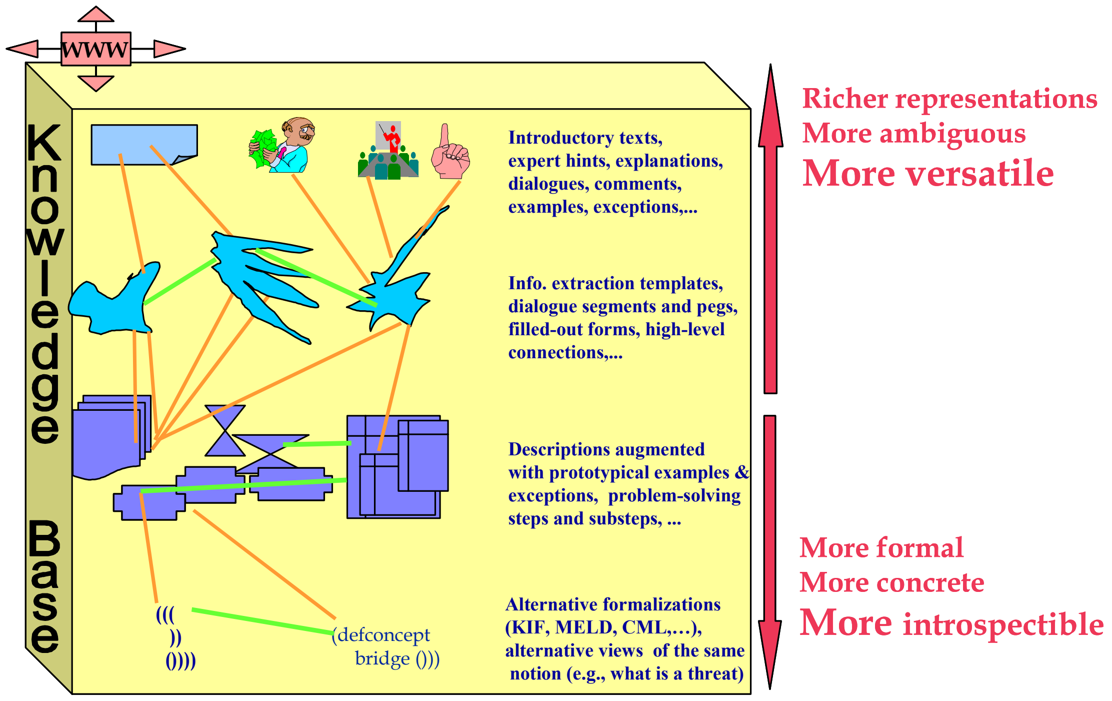

WWW
Knowledge Base
Introductory texts, expert hints, explanations, dialogues, comments, examples, exceptions,...
Info. extraction templates, dialogue segments and pegs, filled-out forms, high-level connections,...
Descriptions augmented with prototypical examples & exceptions, problem-solving steps and substeps, ...
Alternative formalizations (KIF, MELD, CML,...), alternative views of the same notion (e.g., what is a threat)
(((
))
()))
(defconcept
bridge ()))
Richer representations
More ambiguous
More versatile
More formal
More concrete
More introspectible

Figure 6 A Resilient Hyper Knowledge Base (RHKB)

Figure 6 depicts a Resilient Hyper Knowledge Base in contrast with the current approaches illustrated in Figure 5. Originally, the development of the knowledge base starts with documentation, examples, dialogues (perhaps with experts), detailed explanations of special cases, notes on exceptions, hints and comments on specific topics, etc. From these, the developer will extract templates, relevant dialogue segments, itemized lists and tables to organize information, etc. This should be done while always keeping a trail of connections to the original sources. The developer will also indicate some connections between different portions of the original sources. It is our experience that many of the original sources either exist or are converted into resources on the Web, and as a result the developer can exploit the hyperlinks and connections that already exist in these original sources. As the developer continues this analysis, additional sources may be sought and incorporated at the higher levels, further enriching the grounding of the final knowledge base that is being developed.

Next, the developer can identify descriptions and associate with them prototypical examples as well as exceptions, pieces of problem solving knowledge in terms of steps and substeps, tables of parameters and values, etc. Any of these new distillations will continue to be connected to any other pieces in the knowledge base that they were derived from. The developer can also mark alternative views on the same knowledge, indicate contradictory statements by different sources, and dismiss some pieces of knowledge that may not seem useful for current purposes.

Finally, a developer can turn the more structured components into formalized expressions, in one or more languages and formalisms. Contradictory statements can be formalized and connected and marked as contradictory, for someone to pick and choose as they incorporate knowledge into a reasoning engine.

During this process, the developer can annotate the reasons for making certain decisions regarding which knowledge to model and how to model it. These annotations will help further in understanding the rationale for the development of the knowledge base.

We have described here the process with four stages to show the incremental nature of this analysis, but there may be as many levels of refinement as the nature of the knowledge and the final system may require.

Notice that in the higher levels of refinement, the representations may be richer, more versatile, but at the same time more ambiguous. In some sense, plain human language (i.e., text) may be the most mobile vehicle to state knowledge. The many users of the World Wide Web use the same pages for a variety of purposes and tasks, the ultimate signature of true knowledge reuse. At the lowest levels of refinement, the representations are more formal, more concrete, and also more introspectible, lending themselves more to automated analysis and reasoning.

There are many benefits to this approach:

- **Knowledge can be extended more easily.** The formalized, final expressions may not necessarily contain every detail in every knowledge source, but if the need arises the system is better positioned to digest additional knowledge. This could be done in two ways: the developer could incorporate the additional knowledge or perhaps the system could use some automated tools to extract that knowledge itself (since it has access to the sources and how they were originally processed).
- **Knowledge can be reused and translated at any level.** Another system can be built by reusing only the higher levels of the design process and incorporating other sources to create different final formalized expressions. Other developers can tap into any intermediate results of the analysis and do not have to start from scratch. Knowledge does not have to be reused only at the lowest level as it is today.
- **Knowledge can be integrated and translated at any level to facilitate interoperability.** Translators can be built to transform and map knowledge at higher levels. The rationale and meaning of different pieces of knowledge can be available to support translation and interoperation.
- **Intelligent systems will be able to provide better explanations.** We find that many users are reluctant to accept the solutions presented by the systems and ask for explanations not of how the system derived an answer automatically but instead ask for explanations of why the system starts out with a certain fact or belief. When users are shown the reasons for certain assumptions and the fact that certain sources were consulted to make that assumption they are reassured in the competence of the system to provide those answers. Capturing this trail within the knowledge base will enable the system to generate these kinds of justifications and explanations.

- **Content providers will not need to be knowledge engineers.** Although only those trained in the art of designing, modeling, and writing formal expressions can build the more refined portions of RHKBs, anyone can contribute to the higher levels. Many people in diverse disciplines acquire the analytical skills that suffice to organize and digest knowledge sources.

Many existing tools for text extraction (e.g, to extract significant event descriptions from news articles) and discourse analysis (e.g., to segment text into meaningful portions) could be used to support these earlier steps of the analysis. Existing approaches to derive interdependencies among pieces of knowledge may be used to help users create connections among diverse pieces of knowledge. Other tools can be developed to support transformations at the lower levels (e.g., to turn tables into instances and role values). The overhead that may be incurred in creating knowledge bases using this approach is, in our view, not as significant compared to the analysis efforts that developers undergo. It may even save developers time if others can look up the rationale trail instead of asking them directly detailed questions about the portion of the knowledge base they are developing.

The approach presented here has many relations to software engineering methodologies to capture the rationale for software development, and to higher-level languages and frameworks to develop knowledge-based systems. Unfortunately, these methodologies are not common practice among developers of knowledge bases for lack of adequate tools to support developers in this process. Moreover, these methodologies are aimed at software and knowledge engineers and are not very accessible to other potential knowledge base developers, such as end users and/or domain experts.

The Semantic Web will provide an ideal substrate to ground knowledge bases into their original knowledge sources, and to contain the progressively defined pieces of knowledge and the connections among them. More and more every day, knowledge originates and ends in the Web, and we find ourselves extracting knowledge from the Web, processing it inside of a knowledge base, then putting the results back on the Web. It only makes sense to integrate knowledge bases (their content and their reasoning) more closely with the Web.

4 TRELLIS: Building Resilient Hyper Knowledge Bases

The TRELLIS project is our first step towards enabling the creation of RHKBs. In previous work within the EXPECT project, we have investigated several approaches to developing knowledge acquisition tools to enable end users to extend a knowledge base, including analysis of Interdependency Models, scripts to plan acquisition dialogue, exploiting problem solving methods and other background knowledge, and creating English-based structured editors [Blythe et al., 2001; Kim and Gil, 2000; Gil and Tallis, 1998; Swartout and Gil 1995]. EXPECT helps users enter knowledge at the lower levels of an RHKB, and has been shown to be quite effective in several user evaluations with subjects not familiar with programming and formal languages. TRELLIS acquires more informal knowledge and is aimed to support the higher levels of development of RHKBs.

The key innovative ideas behind our approach are:

- **Supporting users to create knowledge fragments from the original sources as well as from other fragments.** The key is to capture how a developer progressively generates new knowledge that results in added value to the original raw information sources. Our goal is to support users to highlight key salient information from large reports and documents, to add new knowledge fragments based on their analysis and integration of existing information, and to finally create semi-formal fragments.
- **Capturing and exploiting semantic interrelationships among information items.** TRELLIS will 1) facilitate semantic markup of relationships between different pieces of knowledge, 2) exploit semantic markups in given problem solving contexts, and 3) suggest additional relationships based on those already specified by the user. Users will be encouraged and rewarded to add valuable annotations over raw information sources, since the more annotations they add the more help the system can provide in their work. When the user chooses to do little or no annotation, the system will provide weaker support (based on default heuristics and strategies) and will still help the user as much as possible.
- **Extensible semantic markup of information items and their relationships**. Users will be able to draw from a core semantic markup language that will contain a basic domain-independent vocabulary to formulate annotations. They will also be able to extend this core language with additional terminology useful in their particular domain. Using this language, users will be able to annotate not only the information items themselves, but they will also be able to annotate the relationships among them, which will enable them to qualify and describe interdependencies between different

information sources and how they relate to a new conclusion or assessment added by the developer. In essence, links between the information items will be first class citizens in the knowledge base.

Figure 7 shows an overview of the architecture of TRELLIS. A User typically starts searching the Web for a certain document, or indicating a pointer to a specific Web resource that contains useful information. Each is considered an information item. Information items may include raw information sources (an image, a text document, a video, etc.) as well as products of previous analysis (by the user or by other users.) All the information items are in some sense the knowledge base that TRELLIS operates on, and we refer to it as the Semantically Marked Information Base, or SMIB. We refer to an information item as an EI2 (Extended Information Items).

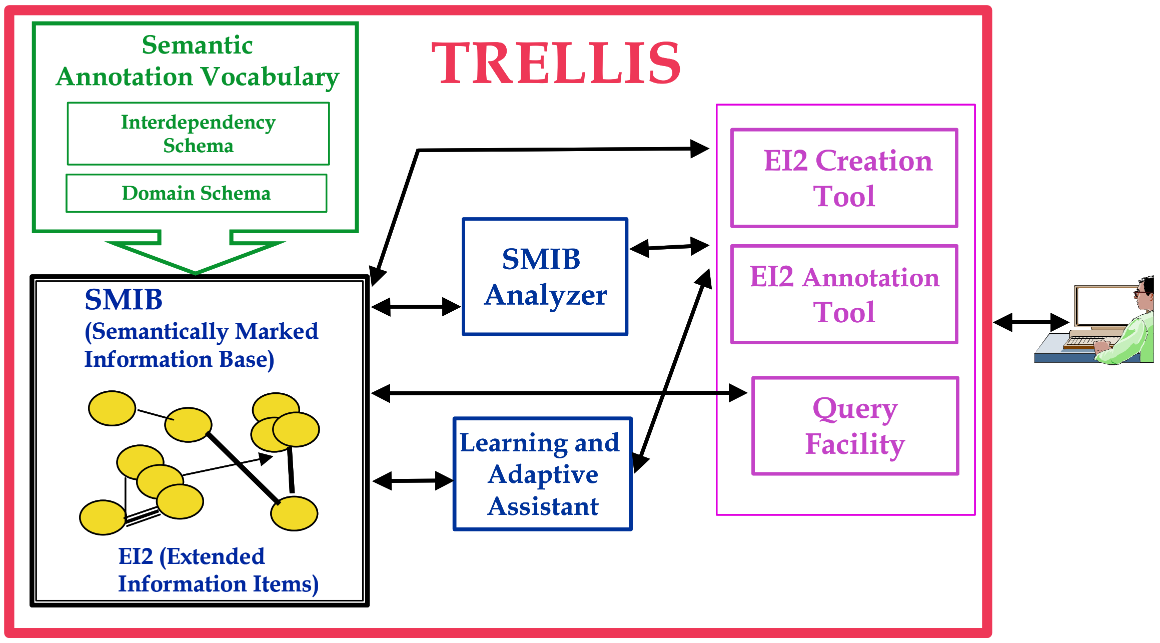

TRELLIS
Semantic Annotation Vocabulary
Interdependency Schema
Domain Schema
SMIB (Semantically Marked Information Base)
EI2 (Extended Information Items)
SMIB Analyzer
Learning and Adaptive Assistant
EI2 Creation Tool
EI2 Annotation Tool
Query Facility

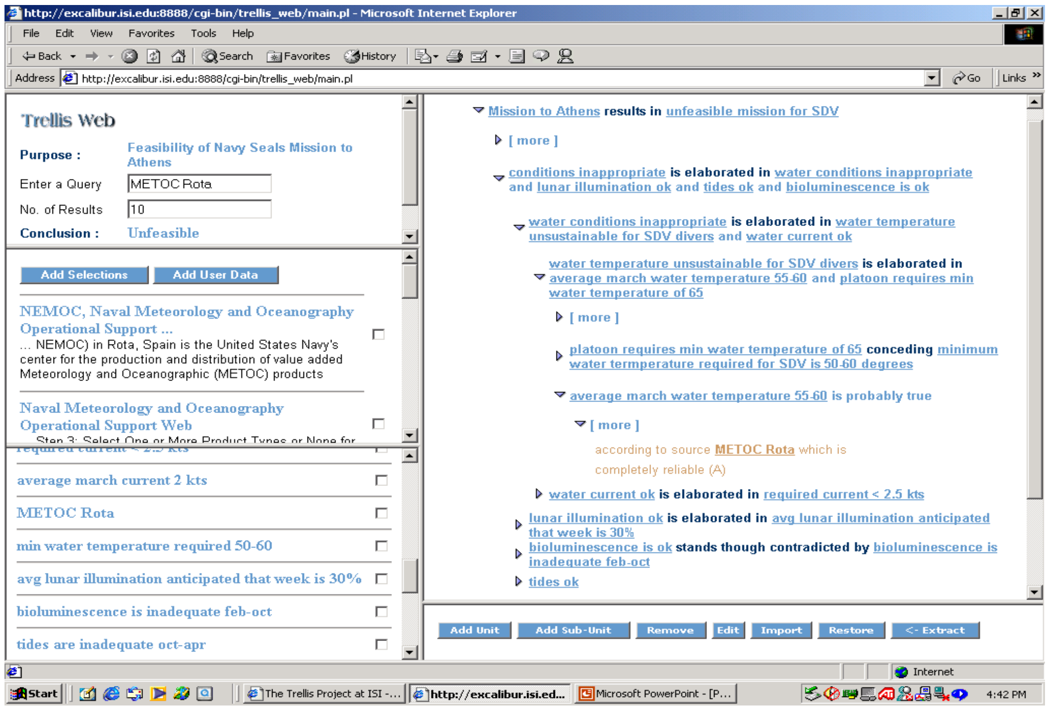
Figure 7 Overview of the TRELLIS Architecture

Users extend the SMIB using two tools: the Annotation Tool and the Creation Tool. They can use the EI2 Annotation Tool to add semantic annotations to an EI2 to describe its contents and to relate them to other EI2. For example, an EI2 may be annotated as containing a map, or an interesting event. The Annotation tool can also be used to relate EI2. The tool will provide an editor with a set of connectors. An example is a connector to denote that two EI2s are contradictory. This way, the user may link an EI2 that contains a description of a product as having a tag price of $20 to another EI2 that has the same product with a price of $25.

The Annotation tool draws on a library of semantic annotations and connectors that will be based on a core domain-independent language defined by the Semantic Annotation Vocabulary. An Interdependency Schema defines a vocabulary for connectors based on a variety of dimensions: pertinence, reliability, credibility, structural (x is example of y, x is part of y, x describes y, etc.) causality (x1 x2...xn contribute to y, x1 x2...xn indicate y, etc.) temporal ordering (x before y, x after y, x simultaneous with y, etc.), argumentation (x may be reason for y, x supports y, etc.). The Domain Schema contains a core vocabulary to annotate the content of documents that extends the Interdependency Schema with domain terms. Our plan is that TRELLIS will provide a core vocabulary, and users will be able to extend it with additional terms.

The Creation Tool enables users to create new EI2. For example, a user may create an EI2 as an assessment that he or she formulates based on existing EI2. If a combination of some subparts of EI2 lets a user conclude or decide on some definition, then the subparts can be captured into a new Information Item, that drops all other irrelevant parts of the original EI2. A new EI2 can be added by extracting or summarizing some of the previous results.

Figure 8 shows a snapshot of the current user interface of TRELLIS. In this case, a user is using TRELLIS to decide whether a mission to take Navy SEALs to Athens is feasible. Given the Web sources consulted and the indicated capabilities of the SEAL team (shown on the left), the user has entered the rationale for deciding that the operation is not feasible.


http://excalibur.isi.edu:8888/cgi-bin/trellis_web/main.pl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://excalibur.isi.edu:8888/cgi-bin/trellis_web/main.pl
Go Links
Add Selections
Add User Data
Add Unit
Add Sub-Unit
Remove
Edit
Import
Restore
<- Extract
Internet
Start
The Trellis Project at ISI -...
http://excalibur.isi.ed...
Microsoft PowerPoint - [P...
4:42 PM

Trellis Web

Purpose : Feasibility of Navy Seals Mission to Athens

Enter a Query: METOC Rota

No. of Results: 10

Conclusion : Unfeasible

NEMOC, Naval Meteorology and Oceanography Operational Support ...
... NEMOC) in Rota, Spain is the United States Navy's center for the production and distribution of value added Meteorology and Oceanographic (METOC) products

Naval Meteorology and Oceanography Operational Support Web
Step 3: Select One or More Product Types or None for

required current < 2.5 kts

average march current 2 kts

METOC Rota

min water temperature required 50-60

avg lunar illumination anticipated that week is 30%

bioluminescence is inadequate feb-oct

tides are inadequate oct-apr

Mission to Athens **results in** unfeasible mission for SDV

- [more]
- conditions inappropriate **is elaborated in** water conditions inappropriate and lunar illumination ok and tides ok and bioluminescence is ok
  - water conditions inappropriate **is elaborated in** water temperature unsustainable for SDV divers and water current ok
    - water temperature unsustainable for SDV divers **is elaborated in** average march water temperature 55-60 and platoon requires min water temperature of 65
      - [more]
      - platoon requires min water temperature of 65 **conceding** minimum water termperature required for SDV is 50-60 degrees
      - average march water temperature 55-60 is probably true
        - [more]
          according to source METOC Rota which is completely reliable (A)
    - water current ok **is elaborated in** required current < 2.5 kts
  - lunar illumination ok **is elaborated in** avg lunar illumination anticipated that week is 30%
  - bioluminescence is ok **stands though contradicted by** bioluminescence is inadequate feb-oct
  - tides ok

Figure 8 A Snapshot of the Current TRELLIS Interface

We plan to extend TRELLIS with learning and adaptive techniques in order to offer to the user improved and extended capabilities over time. As users annotate more EI2 and create new EI2 that capture their analysis, TRELLIS will be able to exploit this information and become increasingly more proactive. We also plan to add a Query Facility that will allow users to search the SMIB based on the semantic annotations of the EI2. It will include a structured editor to guide users to formulate queries using the semantic annotation vocabulary defined in the schemas.

In summary, TRELLIS provides users with tools that enable them to specify information in increasingly more structured form, and to specify semantic annotations that can be exploited for processing and integration of separate information items.

5 Conclusions

Integrating knowledge-based systems within end-to-end systems remains a challenge. In this article, we have shown some practical examples of the difficulties of enabling interoperability among knowledge bases. We have argued that current approaches will only partially solve the interoperability problems, since the knowledge will continue to be trapped in low-level representations with no roots or connections to the rationale that was used in creating them. We have presented a different approach that would create Resilient Hyper Knowledge Bases, which contain knowledge from its initial rendering in unstructured raw information sources and captures how it was refined and interrelated until the developer created the final formal representations. This new generation of knowledge bases will be more resilient to changes and integrations, and will contain highly interconnected knowledge fragments at all levels of abstraction. It will also enable people who have no training in knowledge engineering to contribute content, at least in the initial stages of development. The Semantic Web is an ideal substrate for these knowledge bases, and will be a great contributor to solving interoperability issues.

The spirit of the Web was clearly to empower all people to access and publish information. The Semantic Web will enable them to turn this information into knowledge that can be understood by machines. We must embrace this spirit, and not continue to keep knowledge bases out of the reach of the public at large. Or each other.

6 Acknowledgements

I would like to thank Jim Blythe, Jihie Kim, Larry Kite, Surya Ramachandran, and Pablo Yanez for their helpful discussions on TRELLIS, and Varun Ratnakar for implementing its interface. I would also like to thank Jim Blythe, Jihie Kim, Surya Ramachandran, Bill Swartout , and Andre Valente, for their participation in our group's integration efforts described in this paper, which motivated much of this discussion, as well as other researchers that contributed to the integration that are too many to list here. The TRELLIS project is supported by a grant from the Air Force Office of Scientific Research (AFOSR) with award number F49620-00-1-0337. The integrations of EXPECT's knowledge-based systems described here were funded under the DARPA

High Performance Knowledge Bases (HPKB) program with award number F30602-97-1-0195, the DARPA/Rome Planning Initiative (ARPI) with award number DABT 63-95-C-0059, the DARPA Joint Forces Air Component Commander program (JFACC) with award number F30602-97-C-0118, and the DARPA Control of Agent-Based Systems (CoABS) with award number F30602-97-1-0195.

7 Bibliography

[Blythe et al., 2001] Jim Blythe, Jihie Kim, Surya Ramachandran, and Yolanda Gil. "An Integrated Environment for Knowledge Acquisition." Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI-2001), Santa Fe, New Mexico, January 2001.

[Chalupsky, 2000] Hans Chalupsky. "OntoMorph: A Translation System for Symbolic Knowledge". In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Breckenridge, CO, April 2000.

[Cohen et al., 1999]. Paul R. Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Easter, David Gunning and Murray Burke. "The DARPA High Performance Knowledge Bases Project". In Artificial Intelligence Magazine. Vol. 19, No. 4, pp.25-49, 1998.

[Gil, 2000] Yolanda Gil. "An Analysis of the CoABS TIE-1 Messages: Opportunities and Challenges for Agent Translation Services". USC/ISI Internal Project Report. Marina del Rey, CA, 2000.

[Gil and Melz, 1996] Yolanda Gil and Eric Melz. "Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition." Proceedings of the Thirteen

National Conference on Artificial Intelligence (AAAI-96), Portland, OR, August 4-8, 1996.

[Gil and Tallis, 1997] Yolanda Gil and Marcelo Tallis. “A Script-Based Approach to Modifying Knowledge Bases”. Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), pp. 377-383, AAAI Press, July 27-31 1997.

[Kim and Gil, 1999] Jihie Kim and Yolanda Gil. “Deriving Expectations to Guide Knowledge Base Creation.” Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99), pp. 235-241, AAAI/MIT Press, July 18-22 1999.

[Kim and Gil, 2000] Jihie Kim and Yolanda Gil. "Acquiring Problem-Solving Knowledge from End Users: Putting Interdependency Models to the Test." Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX, July 30-August 3, 2000.

[MacGregor 1991] Robert MacGregor. “The Evolving Technology of Classification-Based Knowledge Representation Systems”. In J. Sowa (Ed.), Principles of Semantic Networks. San Mateo, CA: Morgan Kaufmann. 1991.

[Minsky, 1970] Marvin Minsky. “Form and Content in Computer Science”. Journal of the ACM, 17(2), pp. 197-215, April 1970.

[Swartout and Gil 1995] Bill Swartout and Yolanda Gil. "EXPECT: Explicit Representations for Flexible Acquisition". In Proceedings of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop, February 26-March 3, 1995. Banff, Alberta, Canada.

[Valente et al., 1999] Andre Valente, Thomas Russ, Robert MacGregor and William Swartout. "Building and (Re)Using an Ontology of Air Campaign Planning". In IEEE Intelligent Systems 14:1, pp. 27-36, Jan-Feb, 1999.

[Veloso et al., 1995] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink and Jim Blythe. "Integrating Planning and Learning: The PRODIGY Architecture." Journal of Theoretical and Experimental AI, 7(1), 1995.