WebPlan: Dynamic Planning for Domain-Specific Search in the Internet

Jochem Hüllen & Ralph Bergmann & Frank Weberskirch
University of Kaiserslautern, Dept. of Computer Science
P.O. Box 3049, D-67653 Kaiserslautern, Germany
E-mail: {huellen|bergmann|weberski}@informatik.uni-kl.de

Abstract

Current search engines on the Web basically rely only on purely syntactical textual retrieval mechanism and do not support the complete search process. We argue that searching in the Internet is a dynamic, goal-directed, knowledge-intensive, and iterative process which needs to be planned. The goal of the recently started DFG project *WebPlan* is to develop a search assistant for domain-specific search on the Internet based on dynamic planning and plan execution techniques. During the course of project, the existing planning system *CAPlan* will be extended in different ways in order to deal with incomplete information, information seeking opertors, user interaction, and interleaving planning and execution. For these developments, the domain of localizing specific PC software on the Internet will provide significant application specific guideance.

1 Introduction

During recent years, the World Wide Web emerged to one of the most important sources of up-to-date information about nearly every topic, as well as a huge repository of software, multi-media, and other resources. Although almost anything is available somewhere, one of the biggest problems today is to locate the right resources or pieces of information one is interested in. With the expected further growth of the Web this problem will get dominant or even might prevent the acceptance of Web technology by the larger group of non-expert users. Although currently available search engines on the Web are of some use, their results are often not satisfactory. The main reason for this is that search engines primarily apply syntactic textual retrieval mechanisms. Experienced Web users know how to apply such search engines appropriately during a search process and, in addition, also apply lots of general domain knowledge and search strategies to achieve successful search results. Analyzing search strategies of expert Web users discloses that Web search often becomes a complicated process in which classical search engines are used as a single step executed as part of a more sophisticated search plan. There are following basic problems that make Web search hard: (1) The information sources in the Internet are distributed, extremely heterogeneous and have different structures. (2) Retrieving information from the Internet has a dynamic nature, because servers can be temporarily inaccessible, sources can be moved to other locations or could get even removed completely and continual new sources emerge. (3) The users queries are often incomplete and need to be reformulated several times. (4) Although the Web resources are machine readable they are often not understandable for a machine. (5) Some amount of domain knowledge is required to interpret the retrieved information and to decide whether it is appropriate or not.

Researchers from different areas have started to adress these problems by constructing information gathering systems that automatically query multiple, relevant information sources (Etzioni and Weld, 1994; Arens et al., 1996b; Levy et al., 1996; Kwok and Weld, 1996; Golden et al., 1996; Friedman and Weld, 1997; Singh, 1998).

The *WebPlan* project [1] (funded by the German Science Foundation, DFG) which started in August 1998 aims at developing techniques for building domain-specific search assistants for the Internet. The goal is to support the whole search process by looking at it as a goal directed, dynamic planning and plan execution problem. Since developing a universal domain-independent retrieval assistant does not seem feasible because it would have to deal with a huge amount of common sense knowledge, we focus on domain-specific search. This enables considering some domain specific knowledge in order to recognize the semantic and pragmatic aspects of a user query to some degree. In the project we will primarily focus on the domain of PC software, i.e., finding an appropriate piece of software from the Web (e.g., a Windows95 driver for a particular graphics card).

This paper describes the preliminary architecture of a domain specific search assistant that will be refined and implemented in the course of the project. It shows how this work will extend the *CAPlan* architecture (Weberskirch, 1995; Weberskirch and Paulokat, 1995) developed at the University of Kaiserslautern in recent years.

We will illustrate the planning aspect of systematic domain-specific internet search, first by giving a detailed example and second by introducing the process by a general model. Then, section 4 gives some details about query formulation and interpretation before section 5 describes all important aspects of planning for information gathering. In section 6 some aspects of meta information sources and abstract information goals are presented and section 7 describes the *WebPlan* architecture.

2 Example for Domain-Specific Planned Search

The following example shows a typical systematic search process in the domain of PC software. It illustrates how a user who is familiar with searching the Internet looks for a driver for a graphics board. It demonstrates that this search process is in fact a goal directed planning process. Suppose that an experienced Internet user is searching for a driver for his new graphics board *miroVideo22SD* for *WINDOWS 95*. On an abstract level he usually has the following plan:

1. *Find an URL which offers a driver for the* miroVideo22SD *graphics board.*
2. *Copy this driver to the local harddisk.*

There are several possibilities to refine the first step. One possibility is to find an HTML page that contains the strings: *miro*, *Video* and *22SD*. This plan may not have a high probability of success, but it could be worth trying because it is very simple. A possible refinement is this action sequence:

1.A.1 Choose a conventional search engine.
1.A.2 Construct a query for this machine.
1.A.3 Follow the retrieved links and check if they offer the required driver.

The execution of these actions lead to the following results:

[1] http://wwwagr.informatik.uni-kl.de/~webplan

1.A.1 The search engine AltaVista *was choosen.*
1.A.2 The query miro AND Video AND 22SD *is entered in the advanced mode.*
1.A.3 No links have been found.

This execution doesn't solve the problem but it gives some new information: Since no suitable links were found it would be useful to pose a generalized query. Hence, the following steps are added to the plan:

1.A.4 Generalize the query.
1.A.5 Follow the retrieved links and check if they offer the required driver.

The execution of the added actions had the following effects:

1.A.4 The query miro AND Video *has been asked in the advanced mode.*
1.A.5 More than 8000 links were returned by the search engine. The first hits point to pages containing videos about the surrealist Jean Miro [2].

The 8000 links lead to a huge amount of information and there were even a few links which lead to information about graphics boards from *miro* but it was not possible to find a link to a suitable driver. So, a second more complicate plan is developed:

1.B.1 Find out the manufacturer of the graphics board.
1.B.2 Search for information sources from the manufacturer.
1.B.3 Localize the web area about graphics drivers.
1.B.4 Localize the driver for the miroVideo22SD *board.*

The execution of theses actions lead to the following:

1.B.1 miro *is the manufacturer of the graphics board.*
1.B.2 http://www.miro.com *is the manufacturer's homepage.*
1.B.3 On this page there is a link to graphics driver.
1.B.4 There is a link to a file 243711.exe *which is the searched driver.*
2. Copy 243711.exe *to the local harddisk.*

This example illustrates the following general planning issues: (1) Domain-specific internet search is a dynamic process. New information is determined during this process which influences further (planning) activities. (2) This process is goal directed; in the example the user is not interested in general information about graphics boards or drivers - he wants one specific driver. (3) Domain specific background knowledge is required in different ways (like in steps 1.B.x). (4) AI Planning is an appropiate approach to support this process provided it is able to deal with the dynamics and integrates smoothly with plan execution and monitoring.

3 Process of Systematic Domain-Specific Search

Generalizing the example presented in the previous section, we can identify the cycle for systematic domain-specific internet search shown in Fig.1. This cycle is a modification of the knowledge-based information seeking process in (Carranza and Lenski, 1997). In this cycle the user plays an important role in most of the following phases (the importance of the user is also described in (Belkin, 1996)).

[2] This is a typical result for search engines because they only consider syntactic matches but ignore the semantic or pragmatic aspects.

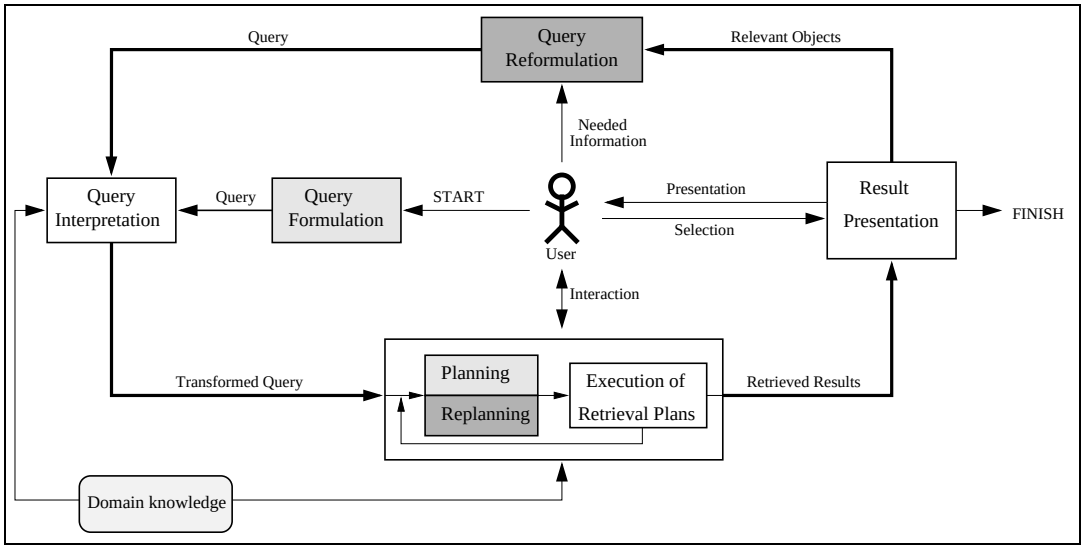

Figure 1: Planning-based domain-specific internet search

- The user poses the initial domain-specific query to the system using a predefined form (**Query Formulation**). It should contain a description of the required information or resource. Often this initial query is vaguely stated, uncomplete or ill-defined. The query is expressed in the language offered by the system. This language should support the formulation of the semantic and pragmatic aspects of a query. Further details are described in the next section.

- In the **Query Interpretation** phase the query is analyzed and transformed. The domain knowledge is used to extract the semantic and pragmatic aspects and to transform the query into an internal representation based on a domain ontology. One important part of the transformed query is the identified planning problem.

- **Planning and execution** of retrieval plans has to be interleaved because the planning of further steps could depend on the results of the execution of a retrieval operator. In this phase domain knowledge is also needed, for example to select certain information sources. There could be interaction between the system and the user (e.g., the structure of some information sources is not completely known by the system, so the user has to help to interpret some retrieved results). Moreover, the user should be able to monitor the whole planning process and especially the execution of the retrieval operators.

- In many cases the first **Result Presentation** will not satisfy the user, therefore, further runs through the cycle are necessary. So the user can select some objects and reformulate the query using the new knowledge. The search process will then start with the reformulated query.

- In the most cases the reformulated query is a modification of the first query but not completely different to it. So, the planning system should be able to replan, starting with the already available retrieval plan from the previous queries. Hence it would be unappropiate to plan from scratch. *CAPlan*, which is the basis for *WebPlan*, supports **Replanning** since it uses the dependence maintenance system *REDUX* (Petrie, 1992; Weberskirch, 1995; Carranza et al., 1998).

4 Query Formulation and Interpretation

The query for classical search engines is normally a list of some keywords. Most search engines support several boolean functions to connect these keywords. Why is the response to such queries often not satisfactory (typically there are no matches or hundreds or thousands of matches and the most interesting are possibly missing)? The main reason for this is that only syntactic matches are performed, but there are no possibilities to consider the semantic and pragmatic aspects of the users query. It would be rather difficult to embed such aspects into universal search engines, because the system would have to deal with a lot of common sense knowledge. In domain specific search, however, it is possible to overcome this problem since domain-specific background knowledge could lead to the real purpose of the query. Furthermore, in a domain specific scenario, the possible semantics and pragmatics a user might have can be analyzed in advance.

Specific query forms can be developed that capture these predefined aspects. Moreoever, it is usefull that the system guides the user during the formulation of the query. For the WebPlan system we want to use domain-specific hierarchically structured query schemes. The user can fill some of the fields of these schemes with free text and others by choosing keywords from defined lists. So, the system will be able to recognize the semantic and pragmatic aspects of the query. Due to the cycle mentioned above, the module for the query input will support mechanisms to reformulate queries using objects from the results of the previous queries.

The purpose of the Query Interpretation phase is to construct a planning problem. Executing a solution plan of this problem performs the actual retrieval operations. Therefore, the purpose of the query has to be identified and the query has to be transformed into a set of information goals (Etzioni et al., 1992; Golden et al., 1996). For this purpose, a lot of domain knowledge (the domain ontology and the knowledge about the information sources) has to be considered. Besides the domain-specific operators, a set of domain independed information goals and information-seeking actions have to be defined.

5 Planning for Domain-Specific Internet Search

Since information gathering is a dynamic goal-directed process and domain-specific search needs domain-specific background knowledge, action planning is an appropriate method. Though a classical AI planning system has to be extended with several features. *WebPlan* will be built on top of *CAPlan*, a SNLP-based (McAllester and Rosenblitt, 1991) plan-space planner. For this approach the following aspects have to be considered:

Incomplete Knowledge: Classical AI planners assume complete knowledge about the start situation of a planning problem. When dealing with information gathering not only the needed information is unknown. Also information like the status of a particular server or which objects an information source contains is unknown. So the system has to deal with incomplete knowlede about the state of the world. One consequence is that the CWA (closed world assumption) does not hold. Hence, if a certain fact cannot be deduced by the system this does not mean that this fact is not true. Instead, the system has to decide whether it should try to get information about it. To support such decisions it is useful to annotate the preconditions and effects of the operators to characterize them: The preconditions can be divided into those that should be deduced and those that should be satisfied by seeking for new information. Similarly, the effects can be devided into those

that change the world and those that change the system's knowledge about the world (Etzioni et al., 1992; Golden et al., 1996). To overcome the problem of missing CWA one can represent facts that are *false*, explicitly. Another approach is to use LCWA (local closed world assumption (Golden and Weld, 1996)). This means that the system uses LCW in single fields, e.g., an information seeking operator surely gives all objects with certain attributes, so the system knows that all other objects do not have these attributes.

Information Seeking Operators: To retrieve new information some information seeking operators a necessary. To integrate them in the system it is usefull to merge the domain-specific operators with some general information seeking operators. To refer to the retrieved information later on, one possibility is to use run-time variables (Golden and Weld, 1996; Knoblock, 1996). They are bound when an information seeking operator is executed and contain the retrieved information.

Planning and Execution: Usually, a solution to a planning problem is a *plan* which transforms the start situation into the finish situation. The execution of such a plan is a task for another system (e.g., a robot). In contrast to this, a solution of an information gathering planning problem is the *execution* of a plan which solves the problem. Since there are operators which retrieve new information, it is possible that other decisions of the planning system depend on the result of those information seeking operators (in the example of section 2 the execution of the operator 1.A.3 causes the system to use operator 1.A.4). So it is necessary to interleave planning and execution of information seeking operators. Consequently, one needs a combined system for planning and execution. Interleaving planning and execution causes two problems: First the system has to decide which operator should be executed at which time (several operators could be executed simultaneously and planning and execution could be done parallel). Secondly, it is not possible to backtrack over executed operators. So, the decision to execute an operator has to be made very carefully. If only operators which change the knowledge about the world but which don't change the world itself are executed, there would be no problems with effects which are impossible to reverse but there could arise unnecessary costs.

Controlling the planning process: The control component of a planner is an important part since it is a common wisdom that the control strategy can have wide influence on the efficiency of planning and on the quality of solution plans. There are a lot of control strategies for classical planning but these have to be modified and expanded for planning for information seeking. In this context the following selection criterions for the three different types of decisions are important:
Operator selection: *costs of the operator and the belonging mandatory operators* [3], *probability that the preconditions can be achieved*, *probability that the operator can be executed with success* (this criterion is important for information gatheting operators)
Goal selection: *probability that this goal can be achieved* (the *difficult* goals should be handled first to avoid late backtracking), *size of the conflict set* [4]
Operator execution: (this decision point occurs not during classical planning) *costs of the operator*, *probability for successful execution of this operator and the blonging mandatory operators* (choosing problematic operators first can give information of the need of replanning), *does the operator have irreversible effects?*, *does the operator produce new information which is profitable for further planning?*

[3] The belonging mandatory operators are the operators which have to be added to the plan if the considered operator is choosen, it is important to regard them

[4] The conflict set consists of all operators which can possibly achieve the goal.

Plan-Space/State-Space: A consequence of interleaving execution and planning is the necessity to modify the general planning algorithm. Since the worlds state could be changed by some executed operators during planning, the plan-space based planning algorithm should be combined with a state-space approach, like in UCP (Kambhampati and Srivastava, 1995; Kambhampati et al., 1998). An important difference to UCP is, as mentioned above, that it is not possible to backtrack over the state-space plan refinements because they are already executed.

User Interaction: As mentioned before, the user plays an important role in the process of information gathering. There are several points in the cycle where the user interacts with the system. So, the planning system has to be extended to support such interaction. The user should be able to control the planning process. Hence, the control component of the planning system has to present the current state of planning to the user. Of course the system should be able to plan independently but the control component should also allow the user, for example, to prefer some operator in a certain situation or to start the execution of an operator. Especially, if there is not enough knowledge about some information source (in the example for the step 1.B.3) the user may help to interpret some results (this could also be done by a text identification system but this is not the goal of the *Webplan* project).

Hierarchical Planning: In the example presented in section 2 there are abstract and concrete operators (1. and 1.A.1). This is typical for information gathering domains. So, a hierarchical planning approach seems to be usefull. *WebPlan* will use an extended version of *CAPlan* that supports hierarchical task decompositions (similar to *DPOCL* (Young et al., 1994) and *UCP* (Kambhampati et al., 1998)) combined with *SNLP* and dependence maintenance for interactive planning.

Information Sources: The advantage to use the Internet as an information source is to have a lot of information about nearly all domains. On the other hand, the different sources in the Internet have no standardized structure. So it is usefull to know the location and the structure (format of query and answer) of some sources that are relevant for a specific domain. In a further state the *WebPlan* system will be able to determine relevant sources automatically. One approach is to involve the user in the problem of interpreting the retrieved information. Another possibility in the future will be to do this automatically since there are a lot of efforts to comment web documents in a formal standardized form (Luke et al., 1997; Fensel et al., 1998). An advanced use of information sources will be discussed in the next section.

6 Using Meta Information

In a system which searches for information the information sources plays an important role. Questions like 'What do I know about the sources?' and 'How can I represent this knowledge?' arise. In this context meta information can be seen as information about information sources. Consequently, we want to examine the use of information sources about information sources. Using such meta information sources is a typical behaviour of humans and so an AI-search agent should do so also.

Example: Suppose a visitor to an automobile fair wants to get information about a certain car from BMW. Of course he wouldn't ask his questions at all stands beginning by the first one. He would try to get an information folder for this fair. Then he would localize the stand of BMW, go there and ask his questions there.

Such actions occur in many search processes (as in the example from section 2). However, when dealing with meta information sources several problems arise:

1. The content of meta information sources have to be characterized. Concerning searching in the internet the content of such sources is like a list of web adresses. To describe a normal information source one use the type of information one can find in this source. This will become difficult if a meta source leads to sources with different types of information.

2. If a meta source leads to a source which is unknown by the system it will be difficult to create a query automatically because the query format is unknown. Also the interpretation of the answer is difficult, because the structure of the answer is unknown. In the first state of our project the user has to help to overcome these problems.

3. The modelling of the planning domain (domain-specific and domain-independent parts) should support the use of meta sources.

To solve these problems the description language for information sources in the knowledge base has to cover the features of meta sources. Furthermore some operators in the planning domain should realize this indirect search approach.

Abstract Information Goals: Using meta information sources can be seen as a kind of abstraction. The following search problem explains the term *abstract* in this context which is different as in the context of abstract planning operators:

Example: Suppose a user is interested in a printer driver for his LaserJet4000 from HP for AutoCAD13 for Windows98. This query leads to a lot of possible keywords:

printer, driver, laser, jet, laserjet4000, hp, hewlett, packard,
auto, cad, autocad13, window, windows98, ...

Of course, some of them are not very usefull, like *auto*. Using all these keywords, the most search engines would give no answer because the search string is too specific. Testing all possible combinations would be very ineffective. What are the best keywords for this query? To decide this, knowledge about the semantic connections of these terms is necessary. Figure 2 shows a selection of such possible connections.

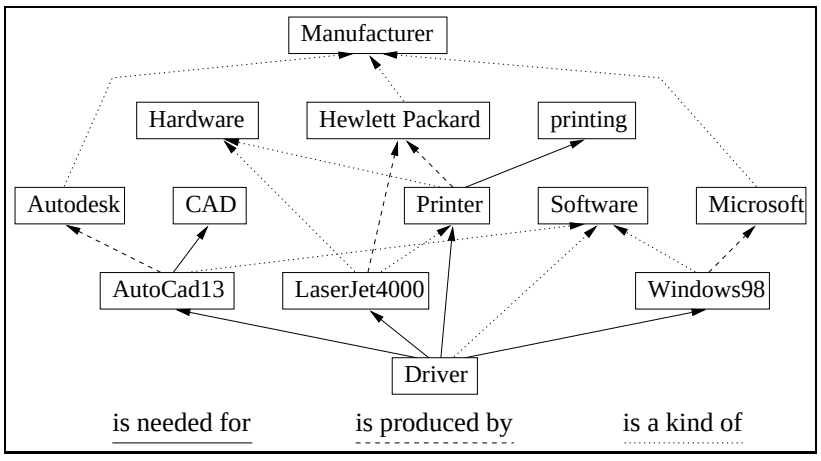

Figure 2: semantic topology of the relevant terms

An advanced user would start his search at the homepage of *Hewlett Packard*, why? Obviously there is a high probability to find a link in the web hypertext from A to B, if there is an edge in the semantic topology from B to A [5] (especially the *is produced by* edges are important since the manufacturer are interested to give informations due to advertising). So probably there would be a sequence of links from *Hewlett Packard* to *driver* (remembering that the driver is the essential object the user is interested in). Other possible starting points would be *Manufacturer* (too abstract), *Autodesk, Microsoft* (not so significant).

So, abstraction in this context means a selection of some of the possible keywords. To route through the web hypertext from *Hewlett Packard* to *Driver* the omitted keywords are reinserted. The major point is that it is impossible (or extreme difficult) to find the searched URL directly. The intelligent way is to find a URL from where a link to the searched URL exists.

7 Architecture of the *WebPlan* System

The *WebPlan* system will be built on top of *CAPlan*. So we extend this system for classic action planning with the features mentioned in the previous sections. Figure 3 illustrates the architecture of the *WebPlan* system, which is connected to the Internet via a WWW-Server. Most components are generic; they realize the basic mechanism for planning-based internet search. The grey components are domain-specific; they contain the knowledge and the code for actions in the concrete application.

Planning Kernel: This module realizes the basic planning algorithm for domain-specific internet search. It can be seen as an extension of the planning kernel of *CAPlan*.

Action Execution: This component is responsible for controlling the execution of actions. It interacts frequently with the planning kernel which determines operators that should be executed. The execution module monitors the execution of the actions which correspond to the operators and gives the results to the planning kernel.

Query Interpretation: In this module the user's query has to be tranformed into a planning problem as discussed in section 4.

Knowledge Base: The knowledge base consists of three main parts: (1) The definition of the deductive and information seeking operators. This declarative knowledge could be compared with the domain definition in a classic planning system and is needed by the planning kernel. (2) The code for the domain-specific information seeking actions. This procedural knowledge is needed by the execution module. (3) The vocabulary and the mechanisms for the queries. This knowledge is mainly used by the Query Input, Control and Interpretation component.

Knowledge Base Editor: This editor should support the application developer to create and maintain the domains. It offers mechanisms to edit the three parts of the knowledge base mentioned above.

Control: This module controls the search process. It shouldn't be mixed up with the control component of the planning kernel which has to control the planning process. The control module of the *WebPlan* system has to monitor the search process and

[5] Of course the directions of the edges in Figure 2 are arbitrary and could be vice versa.

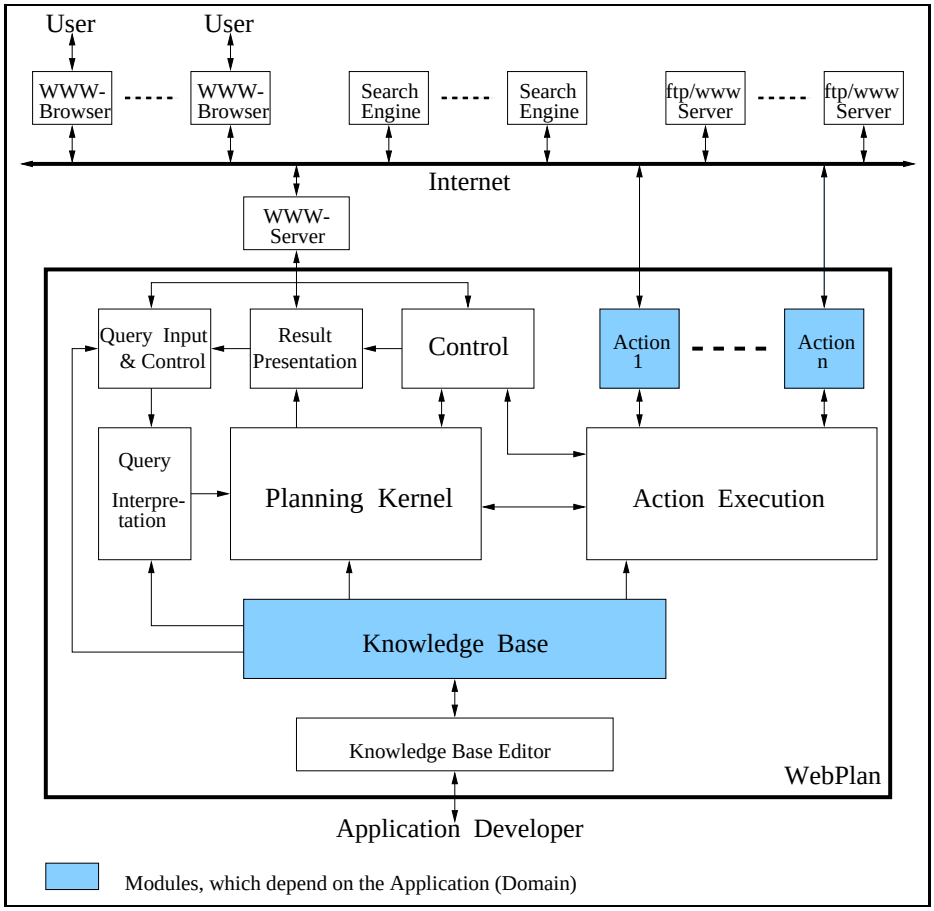

Figure 3: The Architecture of the WebPlan system

has to present the current state (current plan and results of executed actions) to the user, which is done via the Result Presentation module. The user should be able to interfere in the search process as described in the preceding sections.

Result Presentation: This component works up the search results and presents them to the user. Retrieved information could be used to reformulate the initial query.

Query Input & Control: This module implements a WWW-User-Interface for the input and refinement of queries, as well as for interactive control of the search process.

In the *WebPlan* project this general architecture will be refined and implemented. Further, a domain-specific knowledge base for the domain of PC software will be built to realize a search agent for PC software in the Internet.

8 Conclusion

This paper presented the goals of the recently started DGF project *WebPlan*, which focuses on the development of a generative, interactive tool for planning-based domain-specific search in the Internet. Unlike building specific search engines by hand, *WebPlan* aims at developing a generic tool for building domain-specific search agents more easiliy. We will primarily focus on the domain of localizing specific PC software on the internet.

There are several related efforts for building information agents such as *Internet Softbot* (Etzioni and Weld, 1994), *XII* (Golden et al., 1996), *SIMS* (Knoblock, 1996; Arens et al., 1996a), *Occam* (Kwok and Weld, 1996), *Razor* (Friedman and Weld, 1997), *Tessarae* (Singh, 1998), and *Ariadne* (Knoblock et al., 1998). The most closely related projects are those from *ISI*, in which also an AI planner is applied to the information seeking task. *SIMS* focuses on the integration of well-structured databases, while the *Ariadne* project deals with accessing information from more loosly structured Web sources.

Further, the project also aims at advancing basic research in the area of integrating planning and execution in dynamic and uncertain environments (Bergmann and Kott, 1998). The project will be a perfect research platform to study issues about (1) how to build robust plans that also work when the environment changes, (2) developing flexible re-planning strategies, (3) means (e.g. actions) for sensing or monitoring the world, (4) strategies for mapping plans to reactive behavior.

References

Arens, Y., Hsu, C.-N., and Knoblock, C. A. (1996a). Query processing in the sims information mediator. In *Advanced Planning Technology*. AAAI Press.

Arens, Y., Knoblock, C., and Shen, W. (1996b). Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2/3):99–130.

Belkin, N. (1996). Intelligent information retrieval: Whose intelligence? In *Proceedings of 5th Intern. Symposium on Information Science (ISI-96)*.

Bergmann, R. and Kott, A. (1998). Integrating planning, scheduling and execution in dynamic and uncertain environments. In *AAAI Technical Report WS-98-02*. AAAI Press.

Carranza, C. and Lenski, W. (1997). Planning the information seeking process in heterogeneous bibliography-related sources. In *Workshop KI-Techniken für intelligente Mensch-Maschine-Schnittstellen, 21. Deutsche Jahrestagung für KI (KI-97)*.

Carranza, C., Muñoz-Avila, H., Weberskirch, F., and Bergmann, R. (1998). Proposal for a planning approach for information seeking. In Bergmann, R. and Kott, A., editors, *Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*. AAAI Press.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., and Williamson, M. (1992). An approach to planning with incomplete information. In *Proceedings of KR-92*.

Etzioni, O. and Weld, D. (1994). A softbot-based interface to the internet. *Comm. of ACM*.

Fensel, D., Decker, S., Erdmann, M., and Studer, R. (1998). Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conf.*

Friedman, M. and Weld, D. (1997). Efficiently executing information-gathering plans. In *Proceedings of IJCAI-97*.

Golden, K., Etzioni, O., and Weld, D. (1996). Planning with execution and incomplete information. Technical Report UW-CSE-96-01-09, Dep. of Computer Science and Engineering, University of Washington.

Golden, K. and Weld, D. (1996). Representing sensing actions: The middleground revisited. In *Proceedings of KR-96*.

Kambhampati, S., Mali, A., and Srivastava, B. (1998). Hybrid planning for partially hierarchical domains. In *Proceedings of AAAI-98*.

Kambhampati, S. and Srivastava, B. (1995). Universal classical planner: An algorithm for unifying state-space and plan-space planning. In *Proceedings of the 3rd European Workshop on Planning (EWSP-95)*.

Knoblock, C. (1996). Building a planner for information gathering: A report from the trenches. In *Proceedings of the 3rd Intern. Conf. on AI Planning Systems (AIPS-96)*, pages 134–141.

Knoblock, C. A., Minton, S., Ambite, J. L., Modi, N. A. P. J., Muslea, I., Philpot, A. G., and Tejada, S. (1998). Modeling web sources for information integration. In *Proceedings of the 15th National Conf. on Artificial Intelligence*.

Kwok, C. and Weld, D. (1996). Planning to gather information. In *Proceedings of AAAI-96*, pages 32–39.

Levy, A., Rajaraman, A., and Ordille, J. (1996). Query-answering algorithms for information agents. In *Proceedings of AAAI-96*, pages 40–47.

Luke, S., Spector, L., Rager, D., and Hendler, J. (1997). Ontology-based web agents. In *Proceedings of the 1st International Conf. on Autonomous Agents*.

McAllester, D. and Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proceedings of AAAI-91*, pages 634–639.

Petrie, C. (1992). Constrained decision revision. In *Proceedings of AAAI-92*, pages 393–400.

Singh, N. (1998). Unifying heterogeneous information models. *CACM*, 41(5):37–44.

Weberskirch, F. (1995). Combining SNLP-like planning and dependency-maintenance. Technical Report LSA-95-10E, Centre for Learning Systems and Applications, University of Kaiserslautern, Germany.

Weberskirch, F. and Paulokat, J. (1995). CAPlan - ein SNLP-basierter Planungsassistent. In Biundo, S. and Tank, W., editors, *Beiträge zum 9. Workshop 'Planen und Konfigurieren' (PuK-95)*, number DKFI-D-95-01 in DFKI-Dokument. DFKI.

Young, R., Pollack, M., and Moore, J. (1994). Decomposition and causality in partial-order planning. In *Proceedings of the 2nd Intern. Conf. on AI Planning Systems (AIPS-94)*, pages 188–193.