Technical Report

Department of Computer Science
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 97-027

Web Mining: Information and Pattern Discovery on the World Wide Web

by: Robert Cooley, Bamshad Mobasher, Jaideep Srivastava

Web Mining: Information and Pattern Discovery on the World Wide Web *

Robert Cooley, Bamshad Mobasher, Jaideep Srivastava†
{cooley,mobasher,srivasta}@cs.umn.edu
Department of Computer Science
University of Minnesota
4-192 EECS Bldg., 200 Union St. SE
Minneapolis, MN 55455, USA
† *authors are ordered alphabetically*

July 9, 1997

Abstract

Two important and active areas of current research are data mining and the World Wide Web. A natural combination of the two areas, sometimes referred to as *Web mining*, has been the focus of several recent research projects and papers. As with any emerging research area there is no established vocabulary, leading to confusion when comparing research efforts. Different terms for the same concept or different definitions being attached to the same word are commonplace. The term *Web mining* has been used in two distinct ways. The first, which is referred to as *Web content mining* in this paper, describes the process of information or resource discovery from millions of sources across the World Wide Web. The second, which we call *Web usage mining*, is the process of mining Web access logs or other user information user browsing and access patterns on one or more Web localities. In this paper we define Web mining and, in particular, present an overview of the various research issues, techniques, and development efforts in Web content mining and Web usage mining. We focus mainly on the problems and proposed techniques associated with Web usage mining as an emerging research area. We also present a general architecture for Web usage mining and briefly describe the WEBMINER, a system based on the proposed architecture. We conclude this paper by listing issues that need the attention of the research community.

Keywords: data mining, world wide web, association rules, sequential patterns, web mining, access patterns, path analysis.

This work was supported in part by NSF grant ASC-9554517

1 Introduction

With the explosive growth of information sources available on the World Wide Web, it has become increasingly necessary for users to utilize automated tools in order to find, extract, filter, and evaluate the desired information and resources. In addition, with the transformation of the Web into the primary tool for electronic commerce, it is imperative for organizations and companies, who have invested millions in Internet and Intranet technologies, to track and analyze user access patterns. These factors give rise to the necessity of creating server-side and client-side intelligent systems that can effectively mine for knowledge both across the Internet and in particular Web localities.

Web mining can be broadly defined as the discovery and analysis of useful information from the World Wide Web. This broad definition on the one hand describes the automatic search and retrieval of information and resources available from millions of sites and on-line databases, i.e., *Web content mining*, and on the other hand, the discovery and analysis of user access patterns from one or more Web servers or on-line services, i.e., *Web usage mining.*

In this paper, we provide an overview of tools, techniques, and problems associated with both of the dimensions above. Our primary focus, however, is on the second dimension, or Web usage mining. We present a taxonomy of Web mining to clarify our usage of the term, and place various aspects and components of Web mining in their proper context.

There are several important issues, unique to the Web paradigm, that come into play if sophisticated types of analyses are to be done on server side data collections. These include the necessity of integrating various data sources such as server access logs, referrer logs, user registration or profile information; resolving difficulties in the identification of users due to missing unique key attributes in collected data; and the importance of identifying user sessions or transactions from usage data, site topologies, and models of user behavior. We devote the main part of this paper to the discussion of issues and problems that characterize Web usage mining. Furthermore, we survey some of the emerging tools and techniques, and identify several future research directions.

The rest of this paper is organized as follows: Section 2 presents a taxonomy of Web mining and a brief overview of research and development in each of its components. Section 3 identifies the major problems associated with Web usage mining and examines several techniques and approaches used for solving these problems. Section 4 describes the tools available for analyzing and interpreting discovered usage patterns. Section 5 presents a general architecture for Web usage mining and gives an overview of the WEBMINER, as system developed based on this architecture. Finally, sections 6 and 7 present future research directions and conclusions.

2 A Taxonomy of Web Mining

In this section we present a taxonomy of Web mining along its two primary dimensions, namely Web content mining and Web usage mining. We also describe and categorize some of the recent work and the related tools or techniques in each area. This taxonomy is depicted in Figure 1.

2.1 Web Content Mining

The heterogeneity and the lack of structure that permeates much of the ever expanding information sources on the World Wide Web, such as hypertext documents, makes automated discovery, organization, and management of Web-based information difficult. Traditional search and indexing tools of the Internet and the World Wide Web such as Lycos, Alta Vista, WebCrawler, ALIWEB [Kos94], MetaCrawler, and others provide some comfort to users, but they do not generally provide structural

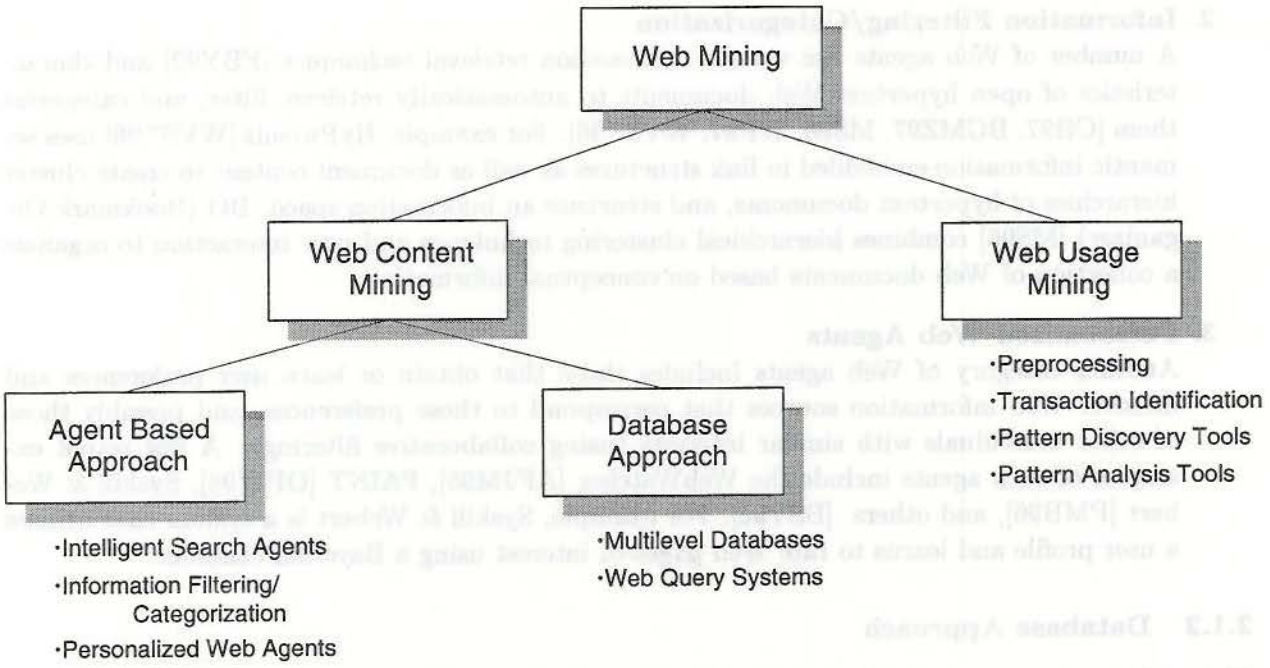


Figure 1: Taxonomy of Web Mining

information nor categorize, filter, or interpret documents. A recent study provides a comprehensive and statistically thorough comparative evaluation of the most popular search tools [LS97].

In recent years these factors have prompted researchers to develop more intelligent tools for information retrieval, such as intelligent Web agents, as well as to extend database and data mining techniques to provide a higher level of organization for semi-structured data available on the Web. We summarize some of these efforts below.

2.1.1 Agent-Based Approach

The agent-based approach to Web mining involves the development of sophisticated AI systems that can act autonomously or semi-autonomously on behalf of a particular user, to discover and organize Web-based information. Generally, the agent-based Web mining systems can be placed into the following three categories:

1. **Intelligent Search Agents**
   Several intelligent Web agents have been developed that search for relevant information using characteristics of a particular domain (and possibly a user profile) to organize and interpret the discovered information. For example, agents such as Harvest [BDH+94], FAQ-Finder [HBML95], Information Manifold [KLSS95], OCCAM [KW96], and ParaSite [Spe97] rely either on pre-specified and domain specific information about particular types of documents, or on hard coded models of the information sources to retrieve and interpret documents. Other agents, such as ShopBot [DEW96] and ILA (Internet Learning Agent) [PE95], attempt to interact with and learn the structure of unfamiliar information sources. ShopBot retrieves product information from a variety of vendor sites using only general information about the product domain. ILA, on the other hand, learns models of various information sources and translates these into its own internal concept hierarchy.

2. **Information Filtering/Categorization**
   A number of Web agents use various information retrieval techniques [FBY92] and characteristics of open hypertext Web documents to automatically retrieve, filter, and categorize them [CH97, BGMZ97, MS96, WP97, WVS+96]. For example, HyPursuit [WVS+96] uses semantic information embedded in link structures as well as document content to create cluster hierarchies of hypertext documents, and structure an information space. BO (Bookmark Organizer) [MS96] combines hierarchical clustering techniques and user interaction to organize a collection of Web documents based on conceptual information.

3. **Personalized Web Agents**
   Another category of Web agents includes those that obtain or learn user preferences and discover Web information sources that correspond to these preferences, and possibly those of other individuals with similar interests (using collaborative filtering). A few recent examples of such agents include the WebWatcher [AFJM95], PAINT [OPW94], Syskill & Webert [PMB96], and others [BSY95]. For example, Syskill & Webert is a system that utilizes a user profile and learns to rate Web pages of interest using a Bayesian classifier.

2.1.2 Database Approach

The database approaches to Web mining have generally focused on techniques for integrating and organizing the heterogeneous and semi-structured data on the Web into more structured and high-level collections of resources, such as in relational databases, and using standard database querying mechanisms and data mining techniques to access and analyze this information.

1. **Multilevel Databases**
   Several researchers have proposed a multilevel database approach to organizing Web-based information. The main idea behind these proposals is that the lowest level of the database contains primitive semi-structured information stored in various Web repositories, such as hypertext documents. At the higher level(s) meta data or generalizations are extracted from lower levels and organized in structured collections such as relational or object-oriented databases. For example, Han, et. al. [ZH95] use a multi-layered database where each layer is obtained via generalization and transformation operations performed on the lower layers. Kholsa, et. al. [KKS96] propose the creation and maintenance of meta-databases at each information providing domain and the use of a global schema for the meta-database. King & Novak [KN96] propose the incremental integration of a portion of the schema from each information source, rather than relying on a global heterogeneous database schema. ARANEUS system [PA97] extracts relevant information from hypertext documents and integrates these into higher-level *derived Web Hypertexts* which are generalizations of the notion of database views.

2. **Web Query Systems**
   There have been many Web-base query systems and languages developed recently that attempt to utilize standard database query languages such as SQL, structural information about Web documents, and even natural language processing for accommodating the types of queries that are used in World Wide Web searches. We mention a few examples of these Web-base query systems here. W3QL [KS95]: combines structure queries, based on the organization of hypertext documents, and content queries, based on information retrieval techniques. WebLog [LSS96]: Logic-based query language for restructuring extracted information from Web information sources. Lorel [QRS+95] and UnQL [BDS95, BDHS96]: query heterogeneous

and semi-structured information on the Web using a labeled graph data model. TSIMMIS [CGMH+94]: extracts data from heterogeneous and semi-structured information sources and correlates them to generate an integrated database representation of the extracted information.

2.2 Web Usage Mining

Web usage mining is the type of Web mining activity that involves the automatic discovery of user access patterns from one or more Web servers. As more organizations rely on the Internet and the World Wide Web to conduct business, the traditional strategies and techniques for market analysis need to be revisited in this context. Organizations often generate and collect large volumes of data in their daily operations. Most of this information is usually generated automatically by Web servers and collected in server access logs. Other sources of user information include *referrer logs* which contains information about the referring pages for each page reference, and user registration or survey data gathered via tools such as CGI scripts.

Analyzing such data can help these organizations to determine the life time value of customers, cross marketing strategies across products, and effectiveness of promotional campaigns, among other things. Analysis of server access logs and user registration data can also provide valuable information on how to better structure a Web site in order to create a more effective presence for the organization. In organizations using intranet technologies, such analysis can shed light on more effective management of workgroup communication and organizational infrastructure. Finally, for organizations that sell advertising on the World Wide Web, analyzing user access patterns helps in targeting ads to specific groups of users.

Most of the existing Web analysis tools [Inc96, eSI95, net96] provide mechanisms for reporting user activity in the servers and various forms of data filtering. Using such tools, for example, it is possible to determine the number of accesses to the server and the individual files within the organization's Web space, the times or time intervals of visits, and domain names and the URLs of users of the Web server. However, in general, these tools are designed to deal handle low to moderate traffic servers, and furthermore, they usually provide little or no analysis of data relationships among the accessed files and directories within the Web space.

More sophisticated systems and techniques for discovery and analysis of patterns are now emerging. These tools can be placed into two main categories, as discussed below.

2.2.1 Pattern Discovery Tools

The emerging tools for user pattern discovery use sophisticated techniques from AI, data mining, psychology, and information theory, to mine for knowledge from collected data. For example, the WEBMINER system [MJHS96, CMS97] introduces a general architecture for Web usage mining. WEBMINER automatically discovers association rules and sequential patterns from server access logs. In [CPY96] algorithms are introduced for finding *maximal forward references* and *large reference sequences*. These can, in turn be used to perform various types of user traversal path analysis such as identifying the most traversed paths thorough a Web locality. Pirolli et. al. [PPR96] use information foraging theory [PC95] to combine path traversal patterns, Web page typing, and site topology information to categorize pages for easier access by users. In subsequent section we will discuss some of these proposed techniques in more detail.

2.2.2 Pattern Analysis Tools

Once access patterns have been discovered, analysts need the appropriate tools and techniques to understand, visualize, and interpret these patterns. Examples of such tools include the WebViz system [PB94] for visualizing path traversal patterns. Others have proposed using OLAP techniques such as data cubes for the purpose of simplifying the analysis of usage statistics from server access logs [Dyr97]. The WEBMINER system [MJHS96] proposes an SQL-like query mechanism for querying the discovered knowledge (in the form of association rules and sequential patterns). These techniques and others are further discussed in the subsequent sections.

3 Pattern Discovery from Web Transactions

As discussed in section 2.2, analysis of how users are accessing a site is critical for determining effective marketing strategies and optimizing the logical structure of the Web site. Because of many unique characteristics of the client-server model in the World Wide Web, including differences between the physical topology of Web repositories and user access paths, and the difficulty in identification of unique users as well as user sessions or transactions, it is necessary to develop a new framework to enable the mining process. Specifically, there are a number of issues in preprocessing data for mining that must be addressed before the mining algorithms can be run. These include developing a model of access log data, developing techniques to clean/filter the raw data to eliminate outliers and/or irrelevant items, grouping individual page accesses into semantic units (i.e. transactions), integration of various data sources such as user registration information, and specializing generic data mining algorithms to take advantage of the specific nature of access log data.

3.1 Preprocessing Tasks

3.1.1 Data Cleaning

Techniques to clean a server log to eliminate irrelevant items are of importance for any type of Web log analysis, not just data mining. The discovered associations or reported statistics are only useful if the data represented in the server log gives an accurate picture of the user accesses of the Web site. Elimination of irrelevant items can be reasonably accomplished by checking the suffix of the URL name. For instance, all log entries with filename suffixes such as, gif, jpeg, GIF, JPEG, jpg, JPG, and map can be removed.

A related but much harder problem is determining if there are important accesses that are not recorded in the access log. Mechanisms such as local caches and proxy servers can severely distort the overall picture of user traversals through a Web site. A page that is listed only once in an access log may have in fact been referenced many times by multiple users. Current methods to try to overcome this problem include the use of cookies, cache busting, and explicit user registration. As detailed in [Pit97], none of these methods are without serious drawbacks. Cookies can be deleted by the user, cache busting defeats the speed advantage that caching was created to provide and can be disabled, and user registration is voluntary and users often provide false information. Methods for dealing with the caching problem include using site topology or referrer logs, along with temporal information to infer missing references.

Another problem associated with proxy servers is that of user identification. Use of a machine name to uniquely identify users can result in several users being erroneously grouped together as one user. An algorithm presented in [PPR96] checks to see if each incoming request is reachable from the pages already visited. If a page is requested that is not directly linked to the previous

pages, multiple users are assumed to exist on the same machine. In [CMS97], user session lengths determined automatically based on navigation patterns are used to identify users. Other heuristics involve using a combination of IP address, machine name, browser agent, and temporal information to identify users [Pit97].

3.1.2 Transaction Identification

Before any mining is done on Web usage data, sequences of page references must be grouped into logical units representing Web transactions or user sessions. A user session is all of the page references made by a user during a single visit to a site. Identifying user sessions is similar to the problem of identifying individual users, as discussed above. A transaction differs from a user session in that the size of a transaction can range from a single page reference to all of the page references in a user session, depending on the criteria used to identify transactions. Unlike traditional domains for data mining, such as point of sale databases, there is no convenient method of clustering page references into transactions smaller than an entire user session. This problem has been addressed in [CMS97] and [CPY96].

[CMS97] assumes that each page reference is used for either navigation purposes to get to another page, or information content purposes. Two types of transactions are defined. The first type is *navigation-content*, where each transaction consists of a single content reference and all of the navigation references in the traversal path leading to the content reference. These transactions can be used to mine for path traversal patterns. The second type of transaction is *content-only*, which consists of all of the content references for a given user session. These transactions can be used to discover associations between the content pages of a site. A given page reference is classified as either navigational or content, based on the time spent on the page. This kind of "page typing" is further delineated in [PPR96], where various page types such as index pages, personal home pages, etc. are used in the discovery of user patterns.

[CPY96] defines the concept of *maximal forward reference* in order to identify transactions. Each transaction is defined to be the set of pages in the path from the first page in the log for a user up the page before a backward reference is made. A new transaction is started when the next forward reference is made. A forward reference is defined to be a page not already in the set of pages for the current transaction. Similarly, a backward reference is defined to be a page that is already contained in the set of pages for the current transaction. For example, an access sequence of `A B C D C B E F E G` would be broken into three transactions, i.e. `A B C D`, `A B E F`, and `A B E G`. The transactions created with this algorithm are similar to the *navigation-content* transactions of [CMS97] and can be used to mine for path traversal patterns.

3.2 Discovery Techniques on Web Transactions

Once user transactions or sessions have been identified as discussed in section 3.1.2, there are several kinds of access pattern mining that can be performed depending on the needs of the analyst. Some of these discovery techniques are discussed below.

3.2.1 Path Analysis

There are many different types of graphs that can be formed for performing path analysis, since a graph represents some relation defined on Web pages (or other objects). The most obvious is a graph representing the physical layout of a Web site, with Web pages as nodes and hypertext links between pages as directed edges. Other graphs could be formed based on the types of Web pages with edges representing similarity between pages, or creating edges that give the number of

users that go from one page to another [PPR96]. Most of the work to date involves determining frequent traversal patterns or large reference sequences from the physical layout type of graph. The *navigation-content* transactions of [CMS97], maximal forward reference transactions of [CPY96], or user sessions of [PPR96] can be used for path analysis. Path analysis could be used to determine most frequently visited paths in a Web site. Other examples of information that can be discovered through path analysis are:

- 70% of clients who accessed `/company/products/file2.html` did so by starting at `/company` and proceeding through `/company/whatsnew`, `/company/products`, and `/company/products/file1.html`;
- 80% of clients who accessed the site started from `/company/products`; or
- 65% of clients left the site after four or less page references.

The first rule suggests that there is useful information in `/company/products/file2.html`, but since users tend to take a circuitous route to the page, it is not clearly marked. The second rule simply states that the majority of users are accessing the site through a page other than the main page (assumed to be `/company` in this example) and it might be a good idea to include directory type information on this page if it is not there already. The last rule indicates an attrition rate for the site. Since many users don't browse further than four pages into the site, it would be prudent to ensure that important information is contained within four pages of the common site entry points.

3.2.2 Association Rules

Association rule discovery techniques [AS94, HS95, SON95, SA95] are generally applied to databases of transactions where each transaction consists of a set of items. In such a framework the problem is to discover all associations and correlations among data items where the presence of one set of items in a transaction implies (with a certain degree of confidence) the presence of other items. In the context of Web mining, this problem amounts to discovering the correlations among references to various files available on the server by a given client. Each transaction is comprised of a set of URLs accessed by a client in one visit to the server. For example, using association rule discovery techniques we can find correlations such as the following:

- 40% of clients who accessed the Web page with URL `/company/products/product1.html`, also accessed `/company/products/product2.html`; or
- 30% of clients who accessed `/company/announcements/special-offer.html`, placed an on-line order in `/company/products/product1`.

Since usually such transaction databases contain extremely large amounts of data, current association rule discovery techniques try to prune the search space according to *support* for items under consideration. Support is a measure based on the number of occurrences of user transactions within transaction logs.

Discovery of such rules for organizations engaged in electronic commerce can help in the development of effective marketing strategies. But, in addition, association rules discovered from WWW access logs can give an indication of how to best organize the organization's Web space. For example, if one discovers that 80% of the clients accessing `/company/products` and `/company/products/file1.html` also accessed `/company/products/file2.html`, but only 30% of those who accessed `/company/products` also accessed `/company/products/file2.html`, then it is

likely that some information in `file1.html` leads clients to access `file2.html`. This correlation might suggest that this information should be moved to a higher level (e.g., `/company/products`) to increase access to `file2.html`.

3.2.3 Sequential Patterns

The problem of discovering sequential patterns [MTV95, SA96] is to find inter-transaction patterns such that the presence of a set of items is followed by another item in the time-stamp ordered transaction set. In Web server transaction logs, a visit by a client is recorded over a period of time. The time stamp associated with a transaction in this case will be a time interval which is determined and attached to the transaction during the data cleaning or transaction identification processes. The discovery of sequential patterns in Web server access logs allows Web-based organizations to predict user visit patterns and helps in targeting advertising aimed at groups of users based on these patterns. By analyzing this information, the Web mining system can determine temporal relationships among data items such as the following:

- 30% of clients who visited `/company/products/`, had done a search in Yahoo, within the past week on keyword w; or
- 60% of clients who placed an online order in `/company/products/product1.html`, also placed an online order in `/company1/products/product4` within 15 days.

Another important kind of data dependency that can be discovered, using the temporal characteristics of the data, are similar time sequences. For example, we may be interested in finding common characteristics of all clients that visited a particular file within the time period $[t_1, t_2]$. Or, conversely, we may be interested in a time interval (within a day, or within a week, etc.) in which a particular file is most accessed.

3.2.4 Clustering and Classification

Discovering classification rules [MAR96, CS96, HCC93, WK91] allows one to develop a profile of items belonging to a particular group according to their common attributes. This profile can then be used to classify new data items that are added to the database. In Web mining, classification techniques allow one to develop a profile for clients who access particular server files based on demographic information available on those clients, or based on their access patterns. For example classification on WWW access logs may lead to the discovery of relationships such as the following:

- clients from state or government agencies who visit the site tend to be interested in the page `/company/products/product1.html`; or
- 50% of clients who placed an online order in `/company/products/product2`, were in the 20-25 age group and lived on the West Coast.

In some cases, valuable information about clients can be gathered by the server automatically from the client browsers. This includes information available on the client side in the history files, cookie files, etc. Other methods used to obtain profile and demographic information on clients include user registration, online survey forms, and techniques such as "anonymous ticketing" [Inc96].

Clustering analysis [KR90, Fis95, NH94] allows one to group together clients or data items that have similar characteristics. Clustering of client information or data items on Web transaction logs, can facilitate the development and execution of future marketing strategies, both online and off-line, such as automated return mail to clients falling within a certain cluster, or dynamically changing a particular site for a client, on a return visit, based on past classification of that client.

4 Analysis of Discovered Patterns

Web site administrators are extremely interested in questions like "How are people using the site?", "Which pages are being accessed most frequently?", etc. These questions require the analysis of the structure of hyperlinks as well as the contents of the pages. The end products of such analysis might include 1) the frequency of visits per document, 2) most recent visit per document, 3) who is visiting which documents, 4) frequency of use of each hyperlink, and 5) most recent use of each hyperlink.

The discovery of Web usage patterns, carried out by techniques described earlier, would not be very useful unless there were mechanisms and tools to help an analyst better understand them. Hence, in addition to developing techniques for mining usage patterns form Web logs, there is a need to develop techniques and tools for enabling the analysis of discovered patterns. These techniques are expected to draw upon from a number of fields including statistics, graphics and visualization, usability analysis, and database querying. In this section we provide a survey of the existing tools and techniques. Usage analysis of Web access behavior being a very new area, there is very little work in it, and correspondingly this survey is not very extensive.

4.1 Visualization Techniques

Visualization has been used very successfully in helping people understand various kinds of phenomena, both real and abstract. Hence it is a natural choice for understanding the behavior of Web users. Pitkow, et al [PB94] have developed the WebViz system for visualizing WWW access patterns. A *Web path paradigm* is proposed in which sets of server log entries are used to extract subsequences of Web traversal patterns called *Web paths*. WebViz allows the analyst to selectively analyze the portion of the Web that is of interest by filtering out the irrelevant portions. The Web is visualized as a directed graph with cycles, where nodes are pages and edges are (inter-page) hyperlinks.

The visualization is composed of two windows, the WebViz control window and the display window [PB94]. The first provides the analyst with controls to adjust the bindings, select a specific time to view, control the animation, and rearrange the layout. The second window's arrangement allows a document's access frequency to be represented by the width of the node representing it, while the node's color represents it recency of access. Link width and color have corresponding meanings. Temporal manipulation is achieved by either the slider of by playback controls.

4.2 OLAP Techniques

On-Line Analytical Processing (OLAP) is emerging as a very powerful paradigm for strategic analysis of databases in business settings. Some of the key characteristics of strategic analysis include 1) very large data volume, 2) explicit support for the temporal dimension, 3) support for various kinds of information aggregation, and 4) long-range analysis, where overall trends are more important than details of individual data items. While OLAP can be performed directly on top of relational databases, industry has developed specialized tools to make it more efficient and effective, e.g. [Adv97]. Also, the research community has recently demonstrated that the functional and performance needs of OLAP require that new information structures be designed. This has led to the development of the *data cube* information model [GBLP96], and techniques for its efficient implementation [HRU96, SDNR96, AAD+96].

Recent work [Dyr97] has shown that the analysis needs of Web usage data have much in common with those of a data warehouse, and hence OLAP techniques are quite applicable. The

access information in server logs is modeled as an append-only history, which grows over time. A single access log is not likely to contain the entire request history for pages on a server, especially since many clients use a proxy server. Because information on access requests will be distributed, and there is a need to integrate it. Since the size of server logs grows quite rapidly, it may not be possible to provide on-line analysis of all of it. Therefore, there is a need to summarize the log data, perhaps in various ways, to make its on-line analysis feasible. Making portions of the log selectively (in)visible to various analysts may be required for security reasons. These requirements for Web usage data analysis show that OLAP techniques may be quite applicable, and this issue needs further investigation.

4.3 Data & Knowledge Querying

One of the reasons attributed to the great success of relational database technology has been the existence of a high-level, declarative, query language, which allows an application to express *what conditions must be satisfied by the data it needs*, rather than having to specify *how to get the required data*. Given the large number of patterns that may be mined, there appears to be a definite need for a mechanism to specify the focus of the analysis. Such focus may be provided in at least two ways. First, constraints may be placed on the database (perhaps in a declarative language) to restrict the portion of the database to be mined for, e.g. [MJHS96]. Second, querying may be performed on the knowledge that has been extracted by the mining process, in which case a language for querying knowledge rather than data is needed. An SQL-like querying mechanism has been proposed for the WEBMINER system [MJHS96], details of which are provided in section 5.

4.4 Usability Analysis

Research in human-computer interactions (HCI) has recently started developing a *computational science of usability* [GSB+94]. The principal goal of this effort is develop a systematic approach to usability studies by adapting the rigorous experimental method of a computational science. The first step is to develop instrumentation methods which collect data about *software usability*, in a manner akin to instrumentation that has been done for analyzing performance. This data is then used to build computerized models and simulations which explain the data. Finally, various data presentation and visualization techniques are used to help analyst understand the phenomenon. This approach can also be used to model the browsing behavior of users on the Web.

As described in this section, there is an increasing need for, as well as interest in, developing techniques and tools to analyze the usage patterns of information on the Web. Some initial ideas have been proposed, but are still in their nascent stages and much work remains to be done. We believe that the techniques which are most effective will include the following characteristics: (i) will be data driven empirical methods, (ii) will use vast amounts of data for validation, (iii) will use rigorous experimental methods and sound statistical analysis, etc.

5 Web Usage Mining Architecture

We have developed a general architecture for Web usage mining which is presented in [MJHS96] and [CMS97]. The WEBMINER is a system that implements parts of this general architecture. The architecture divides the Web usage mining process into two main parts. The first part includes the domain dependent processes of transforming the Web data into suitable transaction form. This includes preprocessing, transaction identification, and data integration components. The second part includes the largely domain independent application of generic data mining and pattern

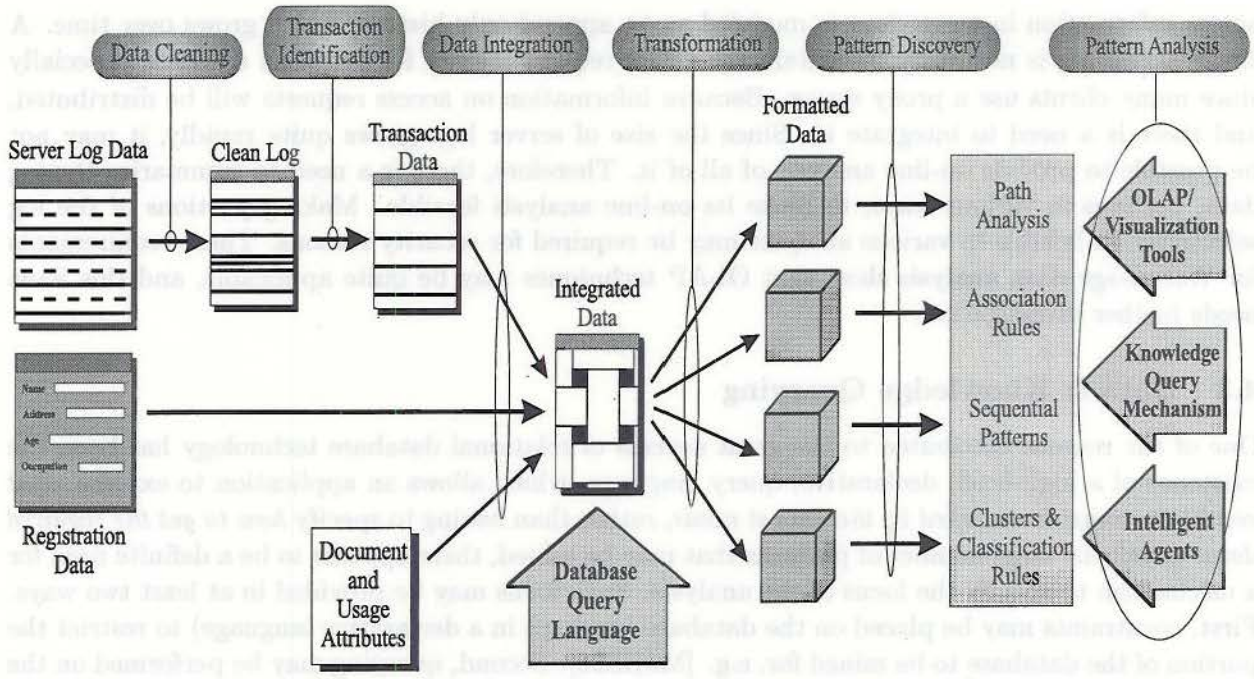


Figure 2: A General Architecture for Web Usage Mining

matching techniques (such as the discovery of association rule and sequential patterns) as part of the system's data mining engine. The overall architecture for the Web mining process is depicted in Figure 2.

Data cleaning is the first step performed in the Web usage mining process. Any of the cleaning techniques discussed in section 3.1.1 can be used to preprocess a given Web server log. Currently, the WEBMINER system uses the simplistic method of checking filename suffixes. Some low level data integration tasks may also be performed at this stage, such as combining multiple logs, incorporating referrer logs, etc.

After the data cleaning, the log entries must be partitioned into logical clusters using one or a series of transaction identification modules. The clean server log can be thought of in two ways; either as a single transaction of many page references, or a set of many transactions each consisting of a single page reference. The goal of transaction identification is to create meaningful clusters of references for each user. Therefore, the task of identifying transactions is one of either *dividing* a large transaction into multiple smaller ones or *merging* small transactions into fewer larger ones. This process can be extended into multiple steps of *merge* or *divide* in order to create transactions appropriate for a given data mining task. A transaction identification module can be defined as either a *merge* or a *divide* module. Both types of modules take a transaction list and possibly some parameters as input, and output a transaction list that has been operated on by the function in the module in the same format as the input. The requirement that the input and output transaction format match allows any number of modules to be combined in any order, as the data analyst sees fit. The WEBMINER system currently has reference length, maximal forward reference, and time window divide modules, and a time window merge module.

Access log data may not be the only source of data for the Web mining process. User registration data, for example, is playing an increasingly important role, particularly as more security and privacy conscious client-side applications restrict server access to a variety of information, such as the client user IDs. The data collected through user registration must then be integrated with the

access log data. There are also known or discovered attributes of references pages that could be integrated into a higher level database schema. Such attributes could include page types, classification, usage frequency, page meta information, and link structures. While WEBMINER currently does not incorporate user registration data, various data integration issues are being explored in the context of Web usage mining. For a study of data integration in databases see [LHS+95]. Once the domain-dependent data transformation phase is completed, the resulting transaction data must be formatted to conform to the data model of the appropriate data mining task. For instance, the format of the data for the association rule discovery task may be different than the format necessary for mining sequential patterns.

Finally, a query mechanism will allow the user (analyst) to provide more control over the discovery process by specifying various constraints. The emerging data mining tools and systems lead naturally to the demand for a powerful data mining query language, on top of which many interactive and flexible graphical user interfaces can be developed [HFW+96]. Some guidelines for a good data mining language were proposed in [HFW+96], which among other things, highlighted the need for specifying the exact data set and various thresholds in a query. Such a query mechanism can provide user control over the data mining process and allow the user to extract only relevant and useful rules. In WEBMINER, a simple Query mechanism has been implemented by adding some primitives to an SQL-like language. This allows the user to provide guidance to the mining engine by specifying the patterns of interest.

As an example, consider a situation where the user is interested in the patterns which start with URL `A`, and contain `B` and `C` in that order, this pattern can be expressed as a regular expression `A*B*C*`. To see how this expression is used within a SQL-like query, suppose further that the analyst is interested in finding all such rules with a minimum support of 1 % and a minimum confidence of 90 %. Moreover, assume that the analyst is interested only in clients from the domain `.edu`, and only wants to consider data later than Jan 1, 1996. The query based on these parameters can be expressed as follows:

```
SELECT association-rules(A*B*C*)
FROM   log.data
WHERE  date >= 960101 AND domain = "edu" AND
       support = 1.0 AND confidence = 90.0
```

This information from the query is used to reduce the scope, and thus the cost of the mining process. The development of a more general query mechanism along with appropriate Web-based user interfaces and visualization techniques such as those discussed in section 4, are planned in the future revisions of the WEBMINER system.

6 Research Directions

The techniques being applied to Web content mining draw heavily from the work on information retrieval, databases, intelligent agents, etc. Most of these techniques are well known and reported elsewhere, hence in this survey we have not focused on Web content mining. Hence, in this survey we have focused on Web Usage Mining, which is just starting as an area of research, and hence has a number of open issues. In the following we provide some directions for future research:

6.1 Data Pre-Processing for Mining

Web usage data is collected in various ways, each mechanism collecting attributes relevant for its purpose. There is a need to pre-process the data to make it easier to mine for knowledge.

Specifically, we believe the following issues need to be addressed:

1. **Instrumentation & Data Collection:** Clearly improved data quality can improve the quality of any analysis on it. A problem in the Web domain is the inherent conflict between the analysis needs of the analysts (who want more detailed usage data collected), and the privacy needs of users (who want as little data collected as possible). This has lead to the development of *cookie files* on one side and *cache busting* on the other. The emerging OPS standard on collecting profile data may be a compromise on what can and will be collected. However, it is not clear how much compliance to this can be expected. Hence, there will be a continual need to develop better instrumentation and data collection techniques, based on whatever is possible and allowable at any point in time.

2. **Data Integration:** Portions of Web usage data exist in sources as diverse as *Web server logs, referral logs, registration files,* and *index server logs.* Intelligent integration and correlation of information from these diverse sources can reveal usage information which may not be evident from any one of them. Techniques from data integration [LHS+95] should be examined for this purpose.

3. **Transaction Identification:** Web usage data collected in various logs is at a very fine granularity. Hence, while it has the advantage of being extremely general and fairly detailed, it also has the corresponding drawback that it cannot be analyzed directly, since the analysis may start focusing on micro trends rather than on the macro trends. On the other hand, the issue of whether a trend is micro or macro depends on the purpose of a specific analysis. Hence, we believe there is a need to group individual data collection events into groups, called *Web transactions* [CMS97], before feeding it to the mining system. While [MJHS96, CPY96, CMS97] have proposed techniques to do so, more attention needs to be given to this issue.

6.2 The Mining Process

The key component of Web mining is the mining process itself. As discussed in this paper, Web mining has adapted techniques from the field of data mining, databases, and information retrieval, as well as developing some techniques of its own, e.g. *path analysis.* A lot of work still remains to be done in adapting known mining techniques as well as developing new ones. Specifically, the following issues must be addressed:

1. **New Types of Knowledge:** Web usage mining studies reported to date have mined for *association rules, temporal sequences, clusters,* and *path expressions.* As the manner in which the Web is used continues to expand, there is a continual need to figure out new kinds of knowledge about user behavior that needs to be mined for.

2. **Improved Mining Algorithms:** The quality of a mining algorithm can be measured both in terms of how *effective* it is in mining for knowledge and how *efficient* it is in computational terms. There will always be a need to improve the performance of mining algorithms along both these dimensions.

3. **Incremental Web mining:** Usage data collection on the Web is incremental in nature. Hence, there is a need to develop mining algorithms that take as input the existing data and mined knowledge, and the new data, and develop a new model in an efficient manner.

4. **Distributed Web mining:** Usage data collection on the Web is distributed by its very nature. If all the data were to be integrated before mining, a lot of valuable information could be extracted. However, an approach of collecting data from all possible server logs is both non-scalable and impractical. Hence, there needs to be an approach where knowledge mined from various logs can be integrated together into a more comprehensive model.

6.3 Analysis of Mined Knowledge

The output of knowledge mining algorithms is often not in a form suitable for direct human consumption, and hence there is a need to develop techniques and tools for helping an analyst better assimilate it. Following are some of the issues that need to be addressed in this area:

1. **Usage Analysis Tools:** There is a need to develop tools which incorporate statistical methods, visualization, and human factors to help better understand the mined knowledge. Section 4 provided a survey of the current literature in this area.
2. **Interpretation of Mined Knowledge:** One of the open issues in data mining, in general, and Web mining, in particular, is the creation of intelligent tools that can assist in the interpretation of mined knowledge. Clearly, these tools need to have specific knowledge about the particular problem domain to do any more than filtering based on statistical attributes of the discovered rules or patterns. In Web mining, for example, intelligent agents could be developed that based on discovered access patterns, the topology of the Web locality, and certain heuristics derived from user behavior models, could give recommendations about changing the physical link structure of a particular site. As a simple example, consider an agent that (among other things) looks at the difference between the visit frequency for a particular page and the number of frequent user paths ending in that page. This difference could be used to determine if the page constitutes an *entry point*. This may suggest the other navigational links should be placed on that page to increase traffic to other clusters of pages.

7 Conclusion

As the popularity of the World Wide Web continues to increase, there is a growing need to develop tools and techniques that will help improve its overall usefulness. Since one of the principal goals of the Web is to act as a *world-wide distributed information resource*, a number of efforts are underway to develop techniques that will make it more useful in this regard. The term *Web mining* has been used to refer to different kinds of techniques that encompass a broad range of issues. However, while meaningful and attractive, this very broadness has caused Web mining to mean different things to different people [HFW+96, MJHS96], and there is a need to develop a common vocabulary for all these efforts. Towards this goal, in this paper we proposed a definition of Web mining, and developed a taxonomy of the various ongoing efforts related to it. Then we presented an brief survey of the research in this area. Next, we concentrated on the aspect of Web mining which focuses on issues related to understanding the behavior of Web users, called *Web usage mining*. We provided a detailed survey of the efforts in this area, even though the survey is short because of the area's newness. We provided a general architecture of a system to do Web usage mining, and identified some of the issues and problems in this area that may require further research and development.

References

[AAD+96] S. Agrawal, R. Agrawal, P.M. Deshpande, A. Gupta, J. Naughton, R. Ramakrishna, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proc. of the 22nd VLDB Conference*, pages 506–521, Mumbai, India, 1996.

[Adv97] Information Advantage. Decision suite users guide. 1997.

[AFJM95] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: A learning apprentice for the world wide web. In *Proc. AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.* 1995.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[BDH+94] C. M. Brown, B. B. Danzig, D. Hardy, U. Manber, and M. F. Schwartz. The harvest information discovery and access system. In *Proc. 2nd International World Wide Web Conference*, 1994.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, 1996.

[BDS95] P. Buneman, S. Davidson, and D. Suciu. Programming constrcuts for unstructured data. In *Proceedings of ICDT'95, Gubbio, Italy*, 1995.

[BGMZ97] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G Zweig. Syntactic clustering of the web. In *Proc. of 6th International World Wide Web Conference*, 1997.

[BSY95] M. Balabanovic, Yoav Shoham, and Y. Yun. An adaptive agent for automated web browsing. *Journal of Visual Communication and Image Representation*, 6(4), 1995.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Irland, Y. Papakonstantinou, J. Ulman, and J. Widom. The tsimmis project: Integration of heterogenous information sources. In *Proc. IPSJ Conference*, Tokyo, 1994.

[CH97] C. Chang and C. Hsu. Customizable multi-engine search tool with clustering. In *Proc. of 6th International World Wide Web Conference*, 1997.

[CMS97] R. Cooley, B. Mobasher, and J. Srivastava. Grouping web page references into transactions for mining world wide web browsing patterns. Technical Report TR 97-021, University of Minnesota, Dept. of Computer Science, Minneapolis, 1997.

[CPY96] M.S. Chen, J.S. Park, and P.S. Yu. Data mining for path traversal patterns in a web environment. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 385–392, 1996.

[CS96] P. Cheeseman and J. Stutz. Baysian classification (autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[DEW96] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison shopping agent for the world wide web. Technical Report 96-01-03, University of Washington, Dept. of Computer Science and Engineering, 1996.

[Dyr97] C. Dyreson. Using an incomplete data cube as a summary data sieve. *Bulletin of the IEEE Technical Committee on Data Engineering*, pages 19–26, March 1997.

[eSI95] e.g. Software Inc. Webtrends. *http://www.webtrends.com*, 1995.

[FBY92] W. B. Frakes and R. Baeza-Yates. *Information Retrieval Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[Fis95] D. Fisher. Optimization and simplification of hierarchical clusterings. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 118–123, Montreal, Quebec, 1995.

[GBLP96] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *IEEE 12th International Conference on Data Engineering*, pages 152–159, 1996.

[GSB+94] M. Guzdial, P. Santos, A. Badre, S. Hudson, and M. Gray. Analyzing and visualizing log files: A computational science of usability. In *HCI Consortium Workshop*, 1994.

[HBML95] K. Hammond, R. Burke, C. Martin, and S. Lytinen. Faq-finder: A case-based approach to knowledge navigation. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press, 1995.

[HCC93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. In *IEEE Transactions on Knowledge and Data Eng.*, volume 5, pages 29–40, 1993.

[HFW+96] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. Dmql: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.

[HRU96] V. Harinarayan, A. Rajaraman, and J.D. Ullman. Implementing data cubes efficiently. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, pages 311–322, Montreal, Canada, 1996.

[HS95] M. A. W. Houtsma and A. N. Swami. Set-oriented mining for association rules in relational databases. In *Proc. of the 11th Int'l Conf. on Data Eng.*, pages 25–33, Taipei, Taiwan, 1995.

[Inc96] Open Market Inc. Open market web reporter. *http://www.openmarket.com*, 1996.

[KKS96] I. Khosla, B. Kuhn, and N. Soparkar. Database search using informatiuon mining. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, 1996.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*. AAAI Press, 1995.

[KN96] R. King and M. Novak. Supporting information infrastructure for distributed, heterogeneous knowledge discovery. In *Proc. SIGMOD 96 Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, Canada, 1996.

[Kos94] M. Koster. Aliweb - archie-like indexing in the web. In *Proc. 1st International Conference on the World Wide Web*, pages 91–100, May 1994.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[KS95] D. Konopnicki and O. Shmueli. W3qs: A query system for the world wide web. In *Proc. of the 21th VLDB Conference*, pages 54–65, Zurich, 1995.

[KW96] C. Kwok and D. Weld. Planning to gather information. In *Proc. 14th National Conference on AI*, 1996.

[LHS+95] E. Lim, S.Y. Hwang, J. Srivastava, D. Clements, and M. Ganesh. Myriad: design and implementaion of federated database prototype. *Software – Practive & Experience*, 25(5):533–562, 1995.

[LS97] H. Vernon Leighton and J. Srivastava. *Precision among WWW search services (search engines): Alta Vista, Excite, Hotbot, Infoseek, Lycos.* http://www.winona.msus.edu/is-f/library-f/webind2/webind2.htm, 1997.

[LSS96] L. Lakshmanan, F. Sadri, and I. N. Subramanian. A declarative language for querying and restructuring the web. In *Proc. 6th International Workshop on Research Issues in Data Engineering: Interoperability of Nontraditional Database Systems (RIDE-NDS'96)*, 1996.

[MAR96] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[MJHS96] B. Mobasher, N. Jain, E. Han, and J. Srivastava. Web mining: Pattern discovery from world wide web transactions. Technical Report TR 96-050, University of Minnesota, Dept. of Computer Science, Minneapolis, 1996.

[MS96] Y. S. Maarek and I.Z. Ben Shaul. Automatically organizing bookmarks per content. In *Proc. of 5th International World Wide Web Conference*, 1996.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, 1995.

[net96] net.Genesis. net.analysis desktop. *http://www.netgen.com*, 1996.

[NH94] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, 1994.

[OPW94] K. A. Oostendorp, W. F. Punch, and R. W. Wiggins. A tool for individualizing the web. In *Proc. 2nd International World Wide Web Conference*, 1994.

[PA97] P. Merialdo P. Atzeni, G. Mecca. Semistructured and structured data in the web: Going back and forth. In *Proceedings of the Workshop on the Management of Semistructured Data (in conjunction with ACM SIGMOD)*, 1997.

[PB94] J. Pitkow and Krishna K. Bharat. Webviz: A tool for world-wide web access log analysis. In *First International WWW Conference*, 1994.

[PC95] P. Pirolli and S. Card. Information foraging in information access environments. In *Proc. of 1995 Conference on Human Factors in Computing Systems (CHI-95)*, 1995.

[PE95] M. Perkowitz and O. Etzioni. Category translation: learning to understand information on the internet. In *Proc. 15th International Joint Conference on AI*, pages 930–936, Montral, Canada, 1995.

[Pit97] J. Pitkow. In search of reliable usage data on the www. In *Sixth International World Wide Web Conference*, pages 451–463, Santa Clara, CA, 1997.

[PMB96] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *Proc. AAAI Spring Symposium on Machine Learning in Information Access*, Portland, Oregon, 1996.

[PPR96] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proc. of 1996 Conference on Human Factors in Computing Systems (CHI-96)*, Vancouver, British Columbia, Canada, 1996.

[QRS+95] D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. In *International Conference on Deductive and Object Oriented Databases*, 1995.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21th VLDB Conference*, pages 407–419, Zurich, Switzerland, 1995.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[SDNR96] A. Shukla, P.M. Deshpande, J. Naughton, and K. Ramaswamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. of the 22nd VLDB Conference*, pages 522–531, Mumbai, India, 1996.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21th VLDB Conference*, pages 432–443, Zurich, Switzerland, 1995.

[Spe97] E. Spertus. Parasite: mining structural information on the web. In *Proc. of 6th International World Wide Web Conference*, 1997.

[WK91] S.M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann, San Mateo, CA, 1991.

[WP97] M. R. Wulfekuhler and W. F. Punch. Finding salient features for personal web page categorization. In *Proc. of 6th International World Wide Web Conference*, 1997.

[WVS+96] R. Weiss, B. Velez, M. A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. K. Gifford. Hypursuit: a hierarchical network search engine that exploits content-link hpertexxt clustering. In *Hypertext'96: The Seventh ACM Conference on Hypertext*, 1996.

[ZH95] O. R. Zaiane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 331–336, Montreal, Quebec, 1995.

Sensor Explication: Knowledge-Based Robotic Plan Execution through Logical Objects

John Budenske and Maria Gini

Abstract— **Complex robot tasks are usually described as high-level goals, with no details on how to achieve them. However, details must be provided to generate primitive commands to control a real robot. A sensor explication concept, that makes details explicit from general commands, is presented.**

We show how the transformation from high-level goals to primitive commands can be performed at execution time and we propose an architecture based on reconfigurable objects that contain domain knowledge and knowledge about the sensors and actuators available. Our approach is based on two premises: (1) plan execution is an information gathering process where determining what information is relevant is a great part of the process; and (2) plan execution requires that many details are made explicit.

We show how our approach is used in solving the task of moving a robot to and through an unknown, and possibly narrow, doorway where sonic range data is used to find the doorway, walls, and obstacles. We illustrate the difficulty of such a task using data from a large number of experiments we conducted with a real mobile robot. The laboratory results illustrate how the proper application of knowledge in the integration and utilization of sensors and actuators increases the robustness of plan execution.

Keywords— **plan execution for robots, knowledge based sensing, object-oriented design.**

I. Introduction

Today's system designers are addressing more and more complex problems to solve applications such as robotics [1], [2], [3], adaptive intelligent systems [4], [5], [6], and manufacturing control systems [7].

These are challenging domains because the environment is dynamic (objects can move in unpredictable ways, and there are multiple independent entities), sensor data are noisy, there are multiple and sometimes conflicting goals, and decisions must be made in real-time. This requires a tight but flexible coupling of perception to action and the ability to take advantage of opportunities that appear in the environment.

The real-time requirement and the unpredictability of the environment make it impossible for a system to obtain complete knowledge of the environment such as to enable globally optimal decisions. So, actions must be selected using only limited, mostly local information. The major difficulty is in controlling and coordinating all the components of the system (sensors, sensor processing units, decision units, actuators, and actuator drivers) so that the global goals are achieved using the resources available.

Detecting achievement of goals is difficult. This is further hampered when goals already achieved become unachieved because of other actions (a stack of blocks might fall down because of a movement of a robot) or because of uncontrollable changes in the environment (a door was unlocked but someone locks it while the robot is inspecting another part of the building so that the robot cannot open it).

In this paper we describe a coherent approach to the execution of plans which allows specification, coordination, and control of a variety of sensors and actuators in a flexible architecture that we call a *Logical Sensor/Actuator (LSA)*. The implementation of our solution resulted in an object oriented system called the *Logical Sensor/Actuator Testbed*. The testbed contains a library of reusable and reconfigurable sensor and actuator processing entities. The design is easy to extend to other sensors or robots and to adapt to simulation systems.

We have focused our investigation on a particular domain where the robot is given a high-level detail-less plan, such as "move through a doorway", that has to be executed in a changing and only partially known environment. Many characteristics in the environment (unknown information, noisy data, etc.) provide variability. Variability not only provides difficulties that must be overcome, but also opportunities to apply knowledge and develop strategies. The task requires precision beyond that of any given single sensor, and requires the combination and interaction of multiple sensors, so providing a rich domain of interaction, abstraction, and application of knowledge.

II. The Application: Robot, Sensors, Actuators, and Knowledge

A large part of our civilized world is unstructured and dynamic. This fact has been one of the leading obstacles to mass-quantity introduction of robots into everyday life. Though there are factories and buildings in which highly structured environments have been introduced for easy integration of robots, the ratio of unstructured to structured environments will always be high. Thus, society's utilization of robotics will depend upon the ability of robots to move safely in these unstructured environments.

The variability of the environment makes it impractical to develop very detailed plans of actions before execution. The environment might change before execution begins and

This work was funded in part by NSF under grant NSF/CDA-9022509, and by the AT&T Foundation.

John Budenske is with Architecture Technology Corporation. The research described in this paper was performed while he was with the Department of Computer Science, University of Minnesota.

Maria Gini is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455. E-mail: gini@cs.umn.edu

thus invalidate the plan. Attempting to determine, prior to execution, everything needed to achieve a goal across the domain of goals, environment states, and sensor availability would be an endless task. It makes more sense to develop, before execution, a high-level plan of what generally should be accomplished, and then generate the necessary explicit details during execution.

Thus, we are faced with a problem: how can a robot accomplish its goals and still remain reactive to its environment? Many approaches have been proposed to make robots reactive [8], [9] and to combine planning with reactivity [1], [10], [11].

Our research has focused on determining what knowledge about sensors, actuators, and processes is necessary for their effective utilization and how to use this knowledge to dynamically reconfigure the flow of data from sensors into actuators. This process of monitoring sensors, collecting data, and using the data to further guide the robot through execution must, at the same time, remain reactive to the external world and also achieve high-level goals. We call this process *sensor explication*[1] which we define as *generating from high-level commands explicit and detailed commands for the sensor and actuators.*

Instead of "directed control from a prespecified plan" the process could be described as "directive reaction to environmental changes during pursuit of a high-level goal". The central research question is determining how to transform a given "goal" into an explicit representation of "what is necessary to achieve the goal".

Suppose a mobile robot is given the task of finding a doorway and passing through it. The task can be decomposed into the sub-tasks of finding the doorway, tracking it during approach, and then passing through it [12].

Suppose the sensors available to the robot are ultrasonic sensors, placed all around the robot to give it a 360° coverage of the surrounding environment. Finding the doorway is more than just looking for a "sonic hole" signifying no object reflections in the sensor stream. The pattern of a short distance, long distance, short distance (or variation thereof) can be attributed to many natural indoor structures as well as to the general occurrence of noise. The real problem comes when the robot finds something that looks like a doorway. How can the robot verify that it is indeed a doorway? To illustrate this point, let us first play a game of "Find the Doorway" (please don't look at the even numbered figures yet!).

Figures 1, 3, 5 show the result of collecting a snapshot of ultra-sonic readings from a ring of 24 sensors. The data are collected as an array of 24 readings (x, y, θ, *distance*), where x and y are the position of the center of the sensor ring on the robot, θ is the angle from the front of the robot to the direction of the sensor, and *distance* is the returned range calculated from the time-of-flight to the object off which the sonar beam is bouncing.

Sonar data are illustrated by drawing the robot (the black square, where the notch indicates the robot's front), and plotting for each sensor an arc at the corresponding angle and range at which reflections appear to bounce off objects. An arc is used since any object in the sensor's cone (30°) can return the sound signal[2]. To further illustrate the data, a line is drawn from the midpoint of each arc to the next. The resulting outline is a visualization of the "open" space around the robot.

Now, examine this visualization and "guess" the location of the doorway. Once you have decided, look at Figures 2, 4, 6, which have a layout of the room superimposed on the original visualization. The game for Figure 1 is easy, but Figures 3 and 5 show how the doorway can be

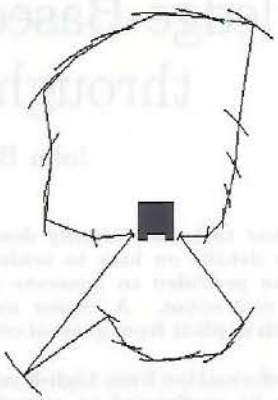


Fig. 1. Sonar data for Case 1. The robot is in a doorway with objects at various distances. The sonar returns are shown as arcs connected by lines through their centers.

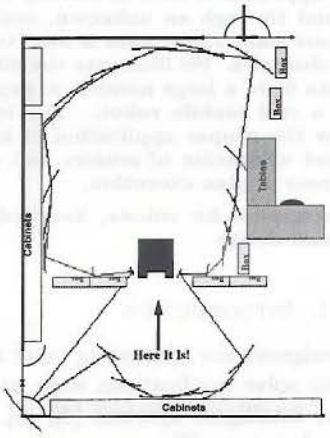


Fig. 2. Sonar data for Case 1 superimposed on a map of the room. The data match closely the walls and furniture in the room and the doorway is easy to find.

[1]from the Webster's dictionary – noun: (origin 1531) 1: to give a detailed explanation of 2: to develop the implications of: analyze logically

[2]Actually, the sensor beam's footprint is far more complex than a cone, but for this illustration, the simplicity of a cone allows for better understanding.

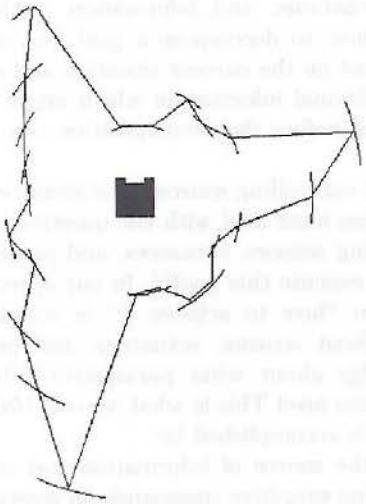


Fig. 3. Sonar data for Case 2. The robot is facing a wall with a doorway slightly to the right of the robot. The doorway is barely visible in the sonar data.

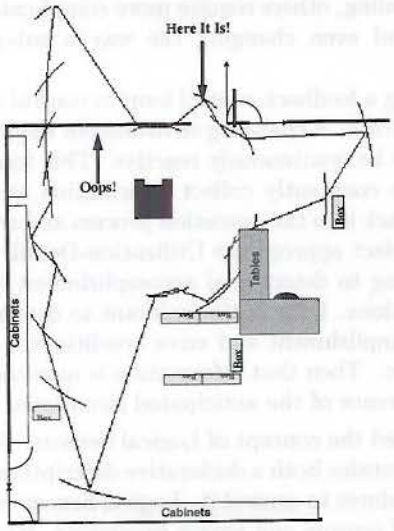


Fig. 4. Sonar data for Case 2 superimposed on a map of the room. The doorway is barely visible in the sonar data. A ghost door, caused by reflections, appears to the left front of the robot. The openings to the right and in the back of the robot could be mistaken for doorways.

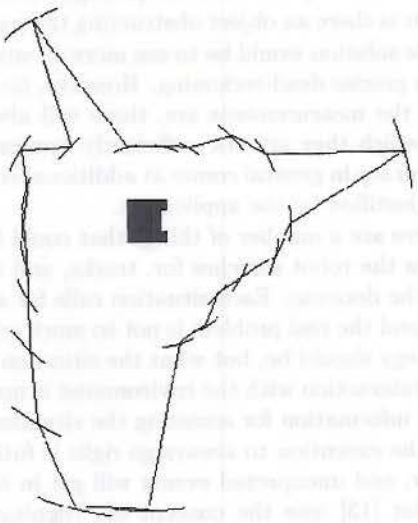


Fig. 5. Sonar data for Case 3. The robot is facing an opening.

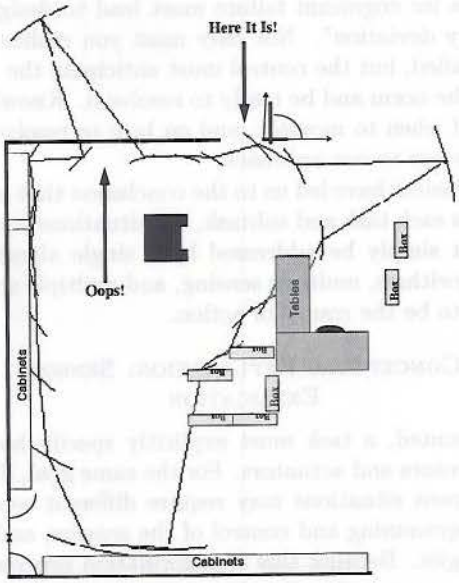


Fig. 6. Sonar data for Case 3 superimposed on a map of the room. There is an opening in front of the robot. A ghost door, caused by reflections, appears to the left rear of the robot. The doorway on the left side of the robot is practically invisible. The openings in front and to the right of the robot could be mistaken for doorways.

missed. Unfortunately, the sensed data are not always as clean as in Figure 1. The last set of figures shows the effect of physical interaction between the sensor beam and surrounding structures. This "noise" (that we call a *ghost door*) was consistently present in our experiments until the robot moved, changing its physical relationship with the environment. Of course moving the robot is not always desirable, since it increases its positional error and might interfere with other parts of the task.

Detecting the doorway is just the beginning. When tracking the doorway, it is easy to lose it and find it again later. If this occurs, is the robot sensing a ghost door or a real one? Even if the robot could sense exactly where the doorway is, moving to and through it can cause other errors. Suppose the bumpers make contact with something. Is this conclusive evidence that there is no doorway? Or does it only mean that there is something between the robot and the doorway? What does it mean to have the bumper contact something when the robot should be passing through the doorway? Is the robot misaligned to the

opening? Is it farther through the opening than previously thought? Is the doorway too small for passage? Is the door in the way? Or is there an object obstructing the pathway?

One possible solution would be to use more accurate sensors and more precise dead-reckoning. However, no matter how accurate the measurements are, there will always be situations in which they are not sufficiently accurate, and increasing accuracy in general comes at additional cost that might not be justified by the application.

Overall, there are a number of things that could happen or go wrong as the robot searches for, tracks, and maneuvers through the doorway. Each situation calls for a different response, and the real problem is not so much what the response strategy should be, but what the situation is. Often extensive interaction with the environment is necessary just to collect information for assessing the situation.

Expecting the execution to always go right is futile. Errors will occur, and unexpected events will get in the way of success. Gat [13] uses the concept of "cognizant failure", or failures which can be monitored for, detected, and thus recovered from. Monitoring for errors is actually monitoring for deviations from the normal expected processing. Thus, design for cognizant failure must lead to design for "anticipatory deviation". Not only must you realize that something failed, but the control must anticipate the deviation from the norm and be ready to resolve it. Knowledge on what and when to monitor, and on how to resolve discrepancies is key to our approach.

Our experiences have led us to the conclusion that as the robot tackles each task and subtask, the situations encountered cannot simply be addressed by a single algorithm. Multiple algorithms, multiple sensing, and multiple strategies appear to be the course of action.

III. Conceptual Explanation: Sensor Explication

To be executed, a task must explicitly specify how to utilize the sensors and actuators. For the same goal, different environment situations may require different sensors, different programming and control of the sensors, and different strategies. Because this transformation process depends on the current situation it must occur at execution time. Essentially, Sensor Explication requires:

1. Identifying the information needed for execution. Sensor Explication must deal with the question of "what information is relevant and thus is needed in order to execute this goal?". The first step in our approach is to use domain knowledge to abstract from the given goal its information needs. To put it another way, the goal "achieve X" is replaced by explicit knowledge on "how to achieve X" and "how to interact with the sensors and actuators which provide the information needed to achieve X". This is what we call *Relevant-Information-Need*. This requires:
   (a) determining the information necessary to accomplish the goal. This depends on the current state, the availability of sensors and actuators, and the goal informational needs;
   (b) decomposing the goal into sub-goals. Sensor Explication must capture the relationship between goals, current situations, and information needs. The choice of how to decompose a goal into sub-goals is contingent on the current situation and often requires additional information which might have to be collected before the decomposition can be completed.
2. Finding and controlling sources of information. Sensor Explication must deal with the question of "what details of using sensors, actuators, and processes are necessary to execute this goal?" In our approach the knowledge on "how to achieve X" is coupled with knowledge about sensors, actuators, and processes, and knowledge about what parameters will satisfy the information need This is what we call *Utilization-Detail*. This is accomplished by:
   (a) selecting the source of information and collecting it. Executing primitive commands on sensors or actuators allows the system to collect the relevant information;
   (b) detecting and resolving conflicts between sub-goals. Some minor conflicts (such as contention for the same resource) can be resolved locally with simple scheduling, others require more complicated reasoning and even changing the way a sub-goal is achieved.
   (c) activating a feedback control loop to control sensors and actuators. A changing environment requires the system to be continuously reactive. This forces the system to constantly collect information and integrate it back into the execution process and continuously reselect appropriate Utilization-Detail values.
   (d) monitoring to detect goal accomplishment and error conditions. Information relevant to determining goal accomplishment and error conditions is determined first. Then that information is monitored for the occurrence of the anticipated situations.

We have utilized the concept of Logical Sensors. A Logical Sensor [14] contains both a declarative description of the sensor and procedures to control it. Logical Sensors are abstracted views of sensors and sensor processing, much like how logical I/O is used to insulate the user from the differences of I/O devices and operating systems. The specifics of the implementation can change without affecting the symbolic-level control system. This allows for easier sensor system reconfiguration, both as a means of providing greater tolerance for sensor failure, and to enhance incremental development of additional sensing and processing devices.

We have expanded the concept of Logical Sensors to also include Logical Actuators, and incorporated it into Logical Sensor/Actuator (LSA) objects. LSAs are plan execution entities, each of them corresponding to an explicit goal. Sensor Explication is basically a process of creating and manipulating these entities by activating/deactivating them and manipulating the flow of information between them. Thus, as the world changes, the entities that pro-

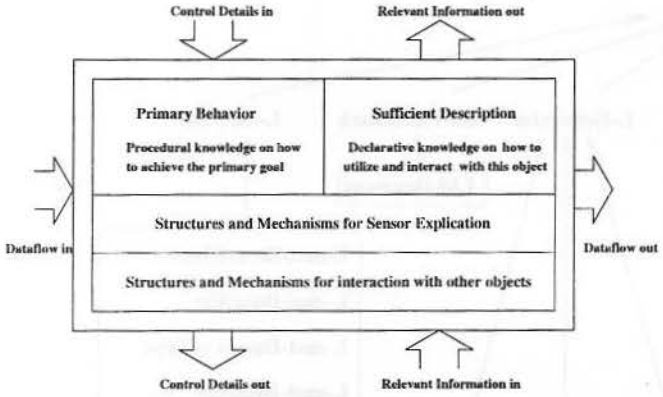


Fig. 7. The Logical Sensor/Actuator Object.

vide information and the datapaths between them change.

By defining Sensor Explication in terms of object interactions, the LSA architecture evolves from simply "plan translation into robotic code" to "coordination of intelligent plan execution objects".

For example, the high-level goal of turning the robot towards a moving object could be defined as a feedback control with little knowledge used within decomposing, determining, and selecting. The high-level goal of moving the robot through a doorway requires decomposing, determining, detecting and resolving conflicts. Sensor Explication decomposes the goal into the lower level goals of finding the doorway, moving towards it, as well as feedback control mapping of sensor-sweeping an area, searching for the doorway until found, and progressively moving through the doorway until clearing it. More details on this example are given later.

IV. System Design: Logical Sensor/Actuator Objects

We developed an object oriented architecture for Sensor Explication built around the concepts outlined in the previous section, and we have implemented it in an object-oriented programming environment. This resulted in a flexible testbed. The testbed is a collection of reconfigurable objects and flexible component structures which can be combined to achieve high-level goals through execution. A general object is shown in Figure 7.

Within the LSA object definition, there are four major sub-components: (1) Primary Behavior; (2) Sufficient Description; (3) Structures and mechanisms for explication; and (4) Structures and mechanisms for interaction with other objects. They are layered into three levels. At the lowest level are the mechanisms for interacting with other objects. The middle level contains the mechanisms for explication. These mechanisms utilize the lowest level to interact with other LSAs. The top level includes the object's Primary Behavior and Sufficient Description. The Primary Behavior defines the actions which this LSA takes, and the Sufficient Description includes, in a declarative form, the information needed by other LSAs to utilize this object.

The LSA object definition has three types of dataflow which pass through it. The first is relevant information from non-subordinate LSAs to controlling LSAs. This is the primary dataflow path where sensor data (raw or processed) pass through the LSA, and are used to set up the lowest level feedback control loops in the architecture. The second is relevant information from subordinate LSAs to a controlling LSA. This path is where internal state information passes from one LSA to another LSA in such a way to be used for control decisions. The third path is used for control details from controlling to subordinate LSAs. This is the control flow path.

V. Implementation of the Testbed

The testbed is implemented on a SUN SPARC 4/330 computer which interacts with a TRC Labmate mobile robot (nicknamed "Eric the Red") over two separate serial lines. The system is comprised of 33K of documented lines of code written in C, Lucid Common Lisp with the Common Lisp Object System and LispView windowing system.

The architecture is implemented as a hierarchy of object classes, as shown in Figure 8. The upper part of the hierarchy (not shown) defines the components that are used in the definition of a general LSA object. The lower part of the object class hierarchy (shown) defines sub-classes organized by overall LSA behavior. The leaf sub-classes (grouped into boxes) correspond to the actual classes used during the system execution. The individual LSAs in the examples shown later resulted from making instantiations of these non-generic object classes. The six primary sub-classes of LSA are explained next.

A. Six Sub-Classes of Logical Sensor/Actuator Objects

The LSA class has been divided into six sub-classes, five major and one used as a special collection point for information about entities in the robot's environment. The primary difference between the sub-classes is in how the Sensor Explication process is implemented and how the dataflow is handled.

L-Controller. The class of L-Controller represents the Sensor Explication process in its entirety, where the Primary Behavior method actually contains the Sensor Explication algorithms. The L-Controller runs the Sensor Explication process and utilizes other LSAs to accomplish its goal. Each instance of L-Controller has a different goal and thus each contains a different collection of domain knowledge that is used by the Sensor Explication process. Internal knowledge is divided into: (1) how to control the subordinate LSAs given the goal and the current situation (input to the L-Controller from its subordinates); and (2) what LSA's to select to satisfy a Relevant-Information-Need. An internal mapping is maintained between the Relevant-Information-Needs identified internally and the external LSA's which provide input for each need. Since LSAs can be built hierarchically, L-Controllers act as building blocks. The subordinate LSAs of an L-Controller can be any of the six sub-classes of LSA, including other L-Controllers.

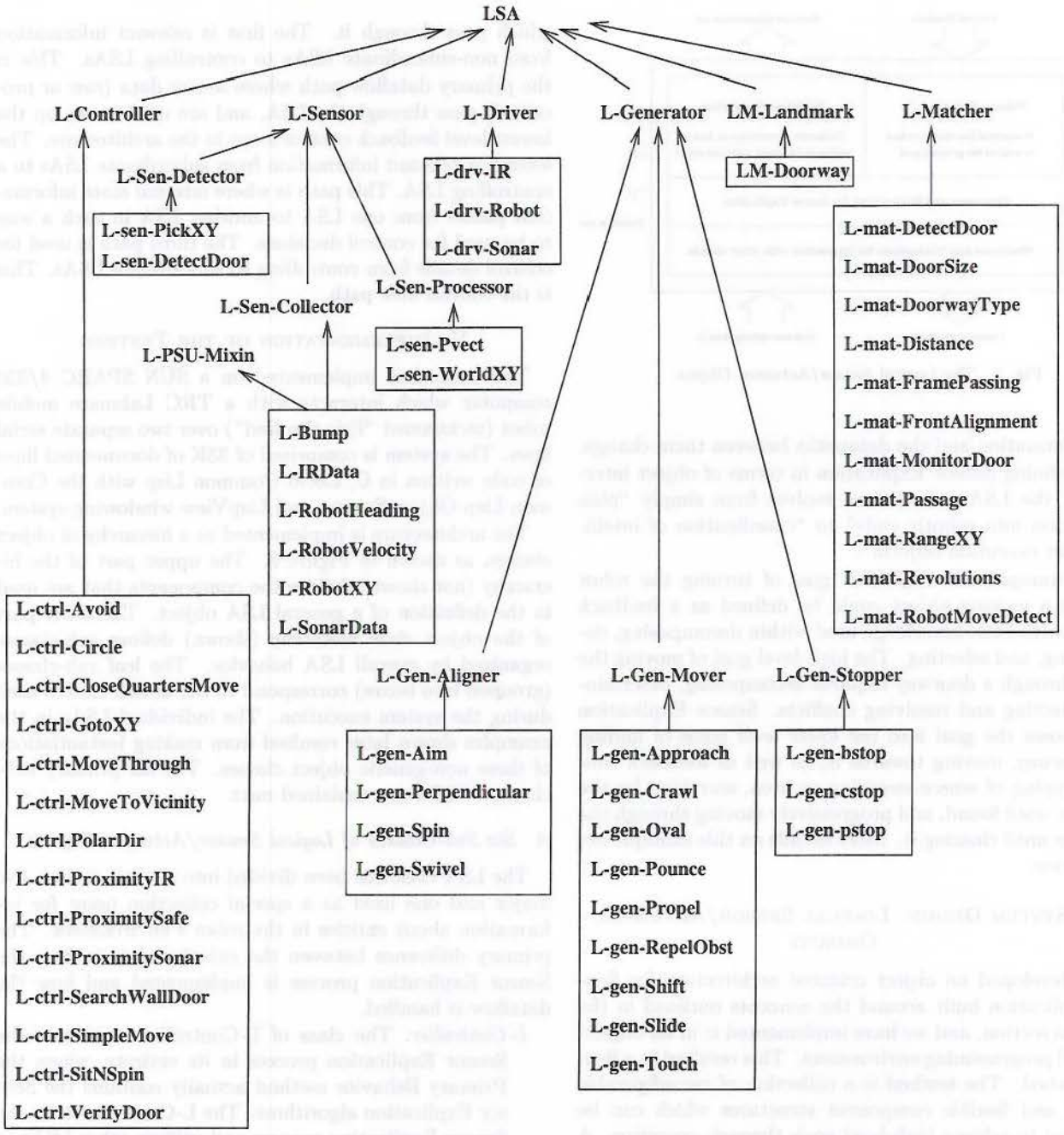


Fig. 8. The LSA object inheritance hierarchy. Arrows point to superclasses, boxes contain primitive LSAs.

L-Sensor. This is an object which produces "data" based on sensing. It can be raw data from actual hardware or it can be the result of combining the hardware with software to produce processed data. This entity is treated like a sensor, its output treated like sensor data (primarily as dataflow to satisfy another LSA's Relevant-Information-Need). L-Sensor is currently divided into three sub-classes for: (1) ordinary collection/processing of raw data; (2) classification of entity types; and (3) detection of world entities.

L-Driver. L-Driver accepts as input multiple commands for individual hardware drivers. It acts as an interface to the hardware, and performs command scheduling. It also resolves minor command conflicts, and routes major conflicts to its controlling LSA. L-Driver differs from the other classes in that its dataflow input is robot actuator commands. We use only one L-Driver per actuator. This allows the L-Driver to monitor the commands passing through it and attempt to detect conflicts, or perform command fusion and mediation.

L-Generator. The L-Generator class accepts sensor data as input and outputs a command meant for a L-Driver. This class can be viewed as a low level, feedback control, looping mechanism between a sensor and the ac-

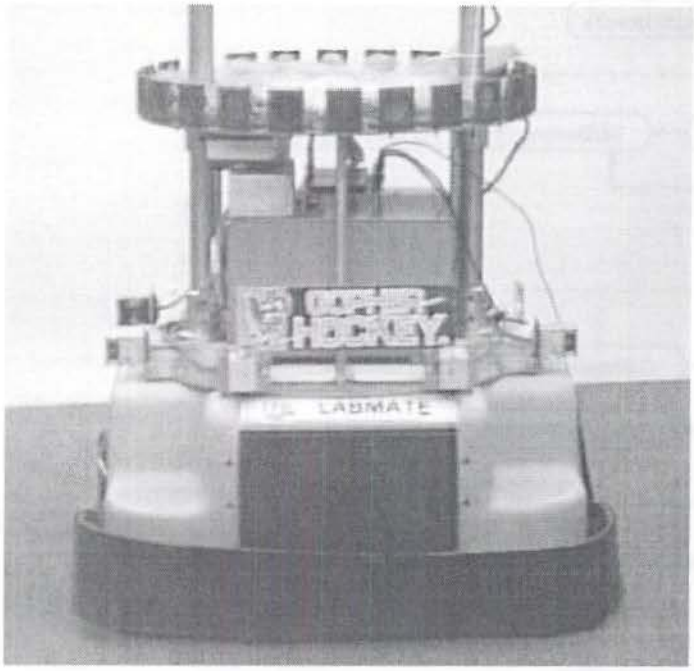


Fig. 9. Eric the Red

tuator. The Primary Behavior for most L-Generators is simply a mapping from a set of possible input values to the desired actuator command.

L-Matcher. L-Matcher is much like L-Sensor, only it also takes as control input a description of a goal or error situation. The class of L-Matcher represents the Sensor Explication sub-process of monitoring relevant information to detect goal accomplishment and error occurrence. L-Matchers use their knowledge of a particular goal or error to interpret incoming sensor data (most likely from a L-Sensor) as to whether that goal or error has occurred.

LM-Landmark. LM-Landmark is used to represent collections of knowledge about physical entities in the robot's environment. It contains information on the expected characteristics of the real entity such as expected location in the world, size, etc. Any information which might be needed by the system or which is derivable about this entity by other LSAs can be contained in this object. For example, when a LSA for detecting doors detects a specific doorway, the LM-Landmark object corresponding to the physical doorway can be updated with the last known location of the doorway. When the system, later, needs the location of the doorway and the door detection LSA is unable to detect it, the corresponding LM-Landmark object can be accessed for an estimated location to be used until the doorway detection LSA can find again the doorway.

VI. The Robot

Eric the Red is an indoor, battery powered platform, which can carry up to 200 pounds of payload (see Figure 9). The base is 70 cm wide, 75 cm long, and 28 cm high. We have outfitted the base with an ultrasonic sonar ring, an infra-red proximity ring, and an aluminum structure, which fits on top of the base, to house the sensor systems.

Twenty-four ultrasonic sensors are mounted at intervals of 15° around a 56 cm diameter ring that is 60 cm above the floor. Each of them emits an ultrasonic pulse whose cone has an angle of approximatively 30°. This arrangement provides an overlapped field coverage of the sonar beams.

Eight infra-red sensors are mounted on the robot's lower corners and can be used to determine the alignment of the robot's physical edges to objects. These sensors are most useful in determining the robot's alignment to doorways.

Eric the Red has back and front bumpers and can provide status information on the current driving mode, heading, x and y location relative to the last reset, left and right wheel velocities and accelerations, bumper contact, and motor fuses. The robot is controlled via a serial connection to a host computer which sends the robot commands and receives status information.

The shape and size of Eric the Red and the limitations of its sensors make the task of passing through doorways challenging. The majority of related research in which a robot freely passed through doorways had the luxury of wide doorways when compared to the width of the robot (often multiples of the robot's width) and, often, of using circular robots that have an easier time maneuvering through a doorway [15]..

VII. An Example of Sensor Explication

To illustrate our approach, we will use an example derived from our lab experiments [16]. When presented with the goal, "MoveThrough Door1", the system first determines what information is initially needed. The goal contains a movement command so the LSA MoveThrough is selected. MoveThrough coordinates among the three main phases illustrated in Figure 10.

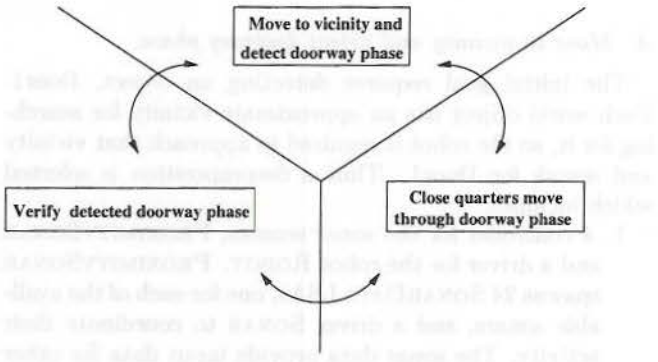


Fig. 10. The knowledge of how to move through a door guides the accomplishment of the goal by coordinating the different phases.

Details on the process are shown in Figures 11, 12, and 13. In the figures the letters at the front of each LSA oval correspond to the first letter of the subclass of that LSA.

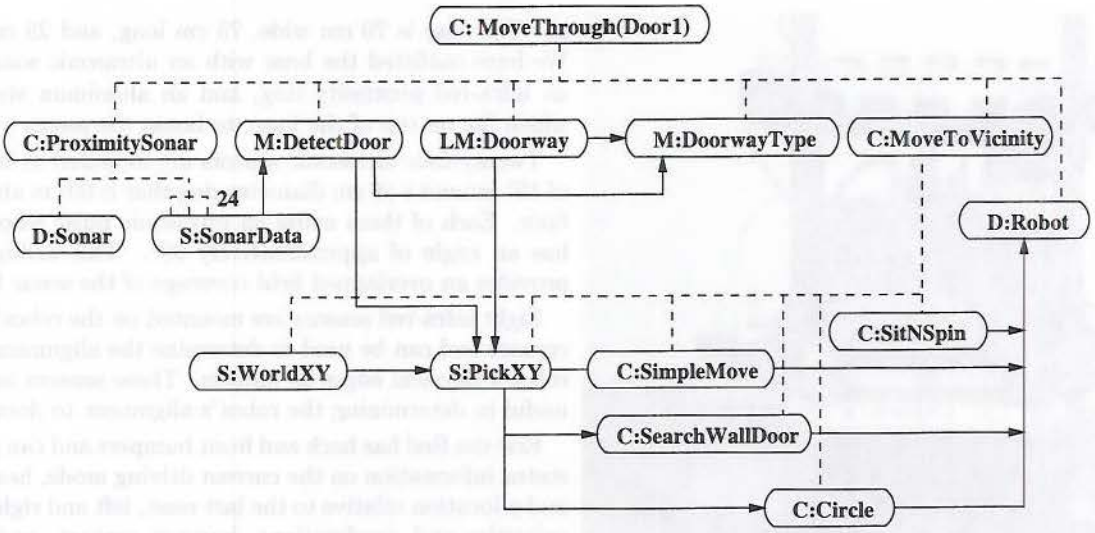


Fig. 11. First phase of moving through a door: move to vicinity and detect doorway. Continuous lines with arrows indicate the primary dataflow, dashed lines indicate the control flow.

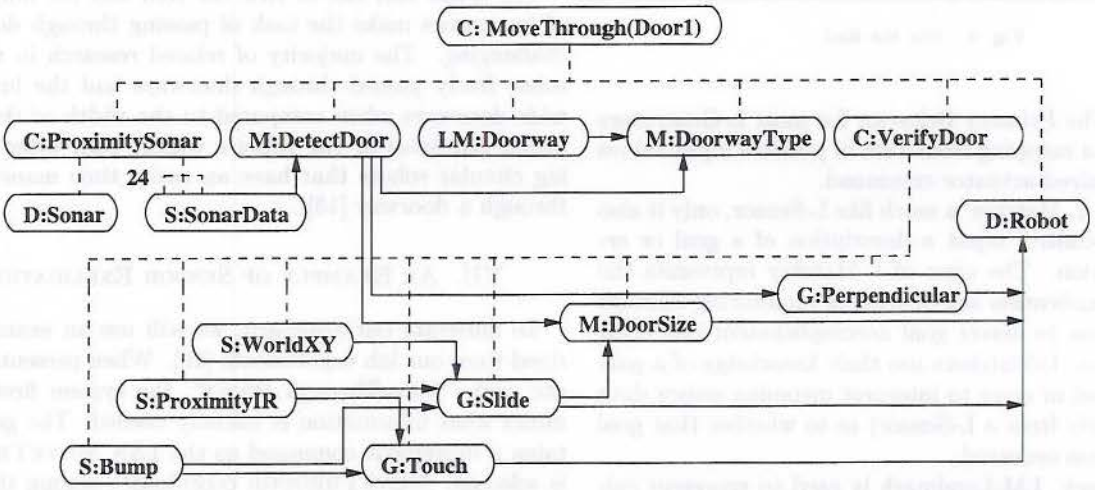


Fig. 12. Second phase of moving through a door: verify detected doorway.

A. Move to vicinity and detect doorway phase.

The initial goal requires detecting an object, Door1. Each world object has an approximate vicinity for searching for it, so the robot is required to approach that vicinity and search for Door1. Thus a decomposition is selected which includes:

1. a controller for the sonar sensors, PROXIMITYSONAR and a driver for the robot ROBOT. PROXIMITYSONAR spawns 24 SONARDATA LSAs, one for each of the available sonars, and a driver SONAR to coordinate their activity. The sonar data provide input data for other LSAs.
2. a matcher DETECTDOOR, that attempts to detect a doorway and, upon detection, returns its location relative to the robot. The output of DETECTDOOR is piped to DOORWAYTYPE. If the type of door detected is different than expected, further investigation would be needed to determine if the doorway found is the right one and how the door is positioned in it.
3. a landmark object DOORWAY for Door1 and a matcher DOORWAYTYPE. The doorway type is important in the selection of other LSAs and their control parameters. Doors that open in vs doors that open out appear differently, and doors in corners require different strategies.
4. a controller MOVETOVICINITY, that moves the robot into a designated vicinity of the doorway, using lower level LSAs. Among the LSAs that are activated by MOVETOVICINITY are: WORLDXY, which passes the robot position to other LSAs, PICKXY, which picks the likely location for the doorway, SEARCHWALLDOOR. which searches the wall looking for doors. SIMPLEMOVE, which moves the robot from one position to another utilizing lower level LSAs; SITNSPIN, which

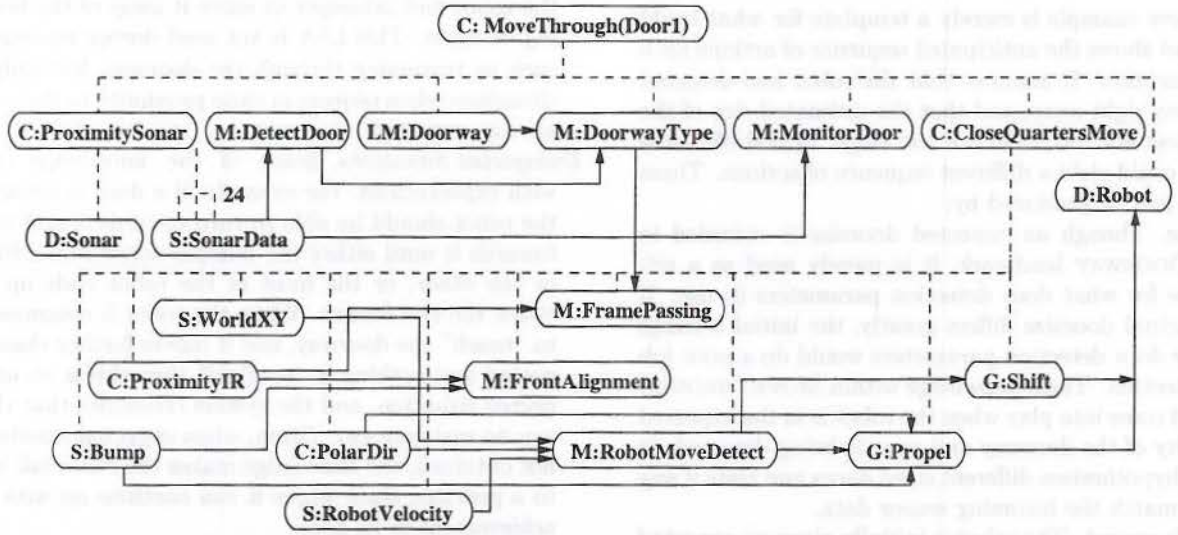


Fig. 13. Third phase of moving through a door: close quarters move through doorway.

spins the robot around in one place; CIRCLE, which circles the robot around a designated position at a designated radius.

Once a door-like structure is found, MOVETOVICINITY and its sub-LSAs are deactivated.

B. *Verify detected doorway phase.*

In the second phase, VERIFYDOOR is activated to make sure that the doorway is truly a doorway, it is open, and the robot could fit through it. This initially sets up DOORSIZE to monitor the size calculations from other LSAs to determine how much clearance there is for the robot to pass through. If the clearance is large, the robot would engage simple, yet imprecise, LSAs for propelling itself through the doorway. If it is a close fit, additional investigation must occur to verify the size and to better align the robot to the opening. The process of verifying the doorway involves three sub-phases:

1. move the robot next to the doorway, within the range of its proximity sensors (TOUCH and PERPENDICULAR),
2. move the robot back and forth in front of the opening to verify its existence and size (SLIDE),
3. center the robot to the doorway (SLIDE and PERPENDICULAR).

Both proximity and sonar sensors are used. Based on the movements within SLIDE, a more precise measurement of the doorway size is obtained, and the position of the door frame determined. This is then used to line the robot up to the doorway.

It is important to note that the switch to VERIFYDOOR is completely controlled by MOVETHROUGH. Within this LSA, the execution of the lower LSAs is monitored and upon completion of that task phase, actions are taken by MOVETHROUGH to switch to, and monitor the next phase.

C. *Close quarters move through doorway phase.*

Once the door frame is verified (assuming the passage is a close fit), VERIFYDOOR is deactivated and two new LSAs are activated. The first is MONITORDOOR, which uses the sonar sensors to monitor the robot's placement and movement through the doorway. It also detects any new obstacles that may appear on the other side of the passage.

This serves as a check on the progress of the second new LSA, CLOSEQUARTERSMOVE, which guides the robot through the opening. Upon activation, CLOSEQUARTERSMOVE also activates and controls a number of sub-LSAs that accomplish these sub-goals:

1. propel the robot through the doorway to a predicted "other side" (PROPEL);
2. keep the robot in line with the door opening as it moves through the doorway (FRONTALIGNMENT), and monitor for obstacles in the forward path using BUMP and PROXIMITYIR;
3. monitor the detection of the doorway frame (FRAMEPASSING) passing each of the side-mounted proximity sensors (PROXIMITYIR);
4. resolve any mis-alignment of the robot to the door frame (SHIFT);
5. monitor the movement of the robot for differentiating between pauses (used by SHIFT and PROPEL) and low velocity stagnation (ROBOTMOVEDETECT). Often during low velocities, the wheel motor drives will freeze up and require a reset of the velocity register.

As the robot moves through the doorway, PROPEL and FRAMEPASSING monitor for evidence of success. Once enough evidence is accumulated, CLOSEQUARTERSMOVE declares success and this prompts MOVETHROUGH to also declare the successful achievement of its goal.

VIII. Variations, Knowledge, Strategies

The above example is merely a template for what could happen and shows the anticipated sequence of actions for a typical execution. It assumes that the robot had detected the doorway right away, and that the estimated size of the doorway was not too small nor too large. In real life, each execution could yield a different sequence of actions. These variations can be produced by:

Doorsize. Though an expected doorsize is recorded in the DOORWAY landmark, it is merely used as a reference for what door detection parameters to use. If the actual doorsize differs greatly, the initial settings of the door detection parameters would do a poor job of detection. Thus, knowledge within MOVETHROUGH would come into play when the robot is in the expected vicinity of the doorway and none is being detected. It then hypothesizes different sized doors and tests if any sizes match the incoming sensor data.

Door placement. The robot is initially given an expected placement of the doorway via the DOORWAY landmark. The actual location varies each run by as much as 1.5m. The strategy is to bring the robot to the expected location and search for the doorway. If it is not initially found, the door detection parameters are "loosened up" and various maneuvers are employed to aid in positioning the robot to detect the doorway.

Door Type and State. Another environmental unknown is the type and state of the door. The type of door is how the door opens (e.g. if there is a door, it opens in/out, to right/left) and how the physical structures around the door encompass it (e.g. is the door at the end of a hallway). The type has a great effect on how the robot would attempt to achieve the goal. In the experiments, we tested only doors of type "straight" and "wall-out-right" . The state of the door indicates whether the door is open or closed.

Ghost Doors. One of the environment interactions which often occur is the temporary appearance of a "ghost door" or the false detection of a doorway which does not exist. Initially, the higher level LSAs perform filtering via pausing the movement of the robot to see if the doorway disappears after a few sensor readings. Sometimes, the structure of the environment will cause fairly persistent ghost doors, and they are not truly discovered until the robot enters the verify detected doorway phase of the execution. Such ghosts can occur in room corners from certain angles, along the cabinets in the room (see Figures 2, 4, and 6), and at times in the middle of the room for no apparent reason (possibly the communication cable to the robot, which hangs from the ceiling, will be detected and appear as a door or obstacle).

Obstacles. Knowledge on how to avoid obstacles is basically included in two LSAs. For movement over large areas, we have an LSA that detects obstacles with the ultrasonic sensors and provides motion commands to steer the robot away from the obstacles. For close obstacles, there is another LSA that uses bumpers and infrared sensors. Upon detection, it takes control of the robot and attempts to move it away of the invading obstacle. This LSA is not used during maneuvers such as traversing through the doorway, but only in situations when objects in close proximity to the robot are not expected.

Unexpected Situations. Some of the knowledge deals with expectations. For example, if a door is detected, the robot should be able to turn towards it and move towards it until either the bumper touches one frame or the other, or the front of the robot ends up between the two frames. When the robot is commanded to "touch" the doorway, and it moves farther than expected and nothing is "touched" then this is an unexpected situation, and the system concludes that there was no real doorway. Often, when expected results are not obtained, the knowledge makes the robot fall back to a previous state where it can continue on with the achievement of its goal.

Sensor Noise. A number of strategies can be taken to reduce sensor noise. One strategy we use is to filter out the noise over a sequence of readings. Filtering works well when the majority of the values are statistically close to the true value, but that may not always be the case. If the robot happens to be sitting in a dead spot (i.e. a point where the sonar echo does not return directly back at the sensor's receiver and is thus undetectable), or an echo point (i.e. a point where multiple returns occur due to echoing) then most to all of the readings are statistically far from the true value. Filtering will not yield an acceptable value. Another problem is caused by moving objects passing near the robot that produce temporary disturbances.

Ways to handle these problems are encoded as domain knowledge within the LSAs. The robot uses its knowledge to decide what to monitor and, upon detection, how to respond with the best strategy.

IX. The Experiments

The laboratory experiments were conducted in a typical indoor open-office environment, simulating an office building with cubicles, for instance, or a warehouse for delivering parts. The environment contains furniture, allows human traffic through it, and hosts multiple doors of various sizes (see Figures 2, 4, and 6). The ceiling is constructed of a sound absorbing material, and neither it nor the floor cause specular noise to the ultrasonic sensors. The walls readily reflect both the ultrasonic and infra-red beams. The furniture reflects ultrasonic beams, but not infra-red. The laboratory floor is carpeted which results in wheel slippage and sometimes noise. Boxes were often used to create makeshift doorways of similar size to the laboratory's doorways. This avoided potential damage to the robot due to collisions. The boxes were primarily white and readily reflected both the ultrasonic and infra-red beams.

The laboratory experiments consisted of three parts: (1) development of the initial implementation on a real robot; (2) knowledge acquisition and refinement through *guided*

TABLE I
RESULTS OF CAPABILITIES EXPERIMENTS OBTAINED WITH FIVE RUNS FOR EACH TEST SET.

Test Set	Door Type	Door Size	Obstacles	Doorway	#goal	#fail	%goal
1	straight	100 cm	none	wide	4	1	80%
2	straight	100 cm	none	wide	4	1	80%
3	straight	81 cm	none	average	5	0	100%
4	straight	81 cm	none	average	4	1	80%
5	straight	70 cm	none	too small	4	1	80%
6	straight	60 cm	none	too small	5	0	100%
7	wall-out-right	81 cm	none	actual	3	2	60%
8	wall-out-right	81 cm	none	actual	4	1	80%
9	wall-out-right	81 cm	2	actual	3	2	60%
10	wall-out-right	81 cm	2	actual	3	2	60%
Totals	2 door types	4 sizes	none or 2	50 runs	39	11	78%

TABLE II
SUMMARY OF EXPERIMENTS.

Characteristics	No. of runs	No. of goal	% goal	Error		
				false-goal	intervention	timeout
straight door	30	26	86.7%	2	1	1
wall-out-right	20	13	65%	0	7	0
not too small	40	30	75%	1	8	1
too small	10	9	90%	1	0	0
no obstacles	40	33	82.5%	0	4	1
obstacles	10	6	60%	0	4	0
Total	50	39	78%	2	8	1

experiments; and (3) final *capabilities experiments*.

The guided experiments were devised to improve the knowledge we originally embedded into the system. Over forty guided runs were executed. New knowledge was encoded back into the system. For example, if during the guided experimentation, it was determined that the robot was prone to become misaligned with the doorway passage upon entering it, additional knowledge (and the necessary supporting LSAs) would be introduced to monitor for the occurrence of misalignment and to correct for it.

To validate the robustness of the approach when faced with previously unencountered situations, we did not use in the guided experiments all the conditions we expected to face later in the capabilities experiments. None of the guided experiments were conducted with doors too small, nor with the door type "wall-out-right". Door sizes ranged from 81 cm to 95 cm and obstacles rarely came into play.

The primary difference between the guided and the capabilities experiments is that during the capabilities experiments no change of the code was allowed and all experiments had to run to completion.

Completion is defined as the robot entering one of these states:

goal: the robot successfully and knowingly passes through the doorway;

false-goal: the robot fails by declaring it has passed through the doorway when it actually has not;

intervention: The robot fails due to human intervention to avoid damage (e.g. the robot runs into an obstacle, but does not detect it and continues to push into it);

timeout: the robot's execution has surpassed a time limit set for the task.

We could think of two other states of completion, in which the robot fails in passing through the doorway, but acknowledges the failure, or passes through the doorway without knowing it succeeded. Actually, the first is never a final experiment result, because the robot would realize that it is failing and initiate a maneuver which brings it back near its starting point to start all over. The robot will try, try again, until one of the other results occurs. The second happened once in the guided experiments but never in the capabilities experiments.

A total of 50 capabilities experiments were run with 10 different scenarios of five runs each. The scenarios varied in distance traveled to goal, doorway sizes, doorway types, and obstacles. Test sets 1 through 6 were with a doorway constructed of cardboard boxes, which allowed changes in its position and size. Sets 7 through 10 were with a real door that opens out and to the right.

Of the 50 capabilities experimental runs, 78% were com-

pleted successfully. The results are shown in Table I. The results from the experiment runs which did not have obstacles were surprisingly good. The robot succeeded 90% of the time in determining that the doorway was too small, which is fairly good considering the lack of guided experiments in this area. The same can be said for the results of the "wall-out-right" runs, where walls are to the immediate right of the doorway. The clearance between the robot and the door frame in Test Sets 7 through 10 is very limited, and requires precision in sensing and control beyond the individual sensors available on our robot. The experiments provide evidence of the robustness of the approach and of transferability of knowledge to never-seen-before scenarios.

In the eleven cases that did not succeed, as shown in Table II, eight were caused by the robot getting wedged in the doorway (seven of those where with the wall-out-right door type), two were caused by the robot declaring it had successfully traversed the doorway when it fact it did not, and one was caused by the robot working itself into a corner and never escaping (and so failing because of timeout). A detailed analysis of the experiments is in [17].

The robot's performance in dealing with ghost doors was extremely good. The door detection LSA often detected ghost doors in the middle of the room or on doorway-less walls. The robot would approach the ghost door and start its verification procedures and always determined it to be a false door.

One unexpected observation was that the robot had problems maneuvering near the edges of the doorway. If the robot ever became misaligned with the doorway so that it was facing towards one frame with the other frame against its side, it would be in serious trouble. Most of the error-intervention runs resulted from this situation. The development of specific domain knowledge and a more sophisticated proximity detection method could help in this situation.

Another unexpected observation was that the robot had difficulty with the widest doorway (100 cm). The door detection LSA could not quickly nor consistently detect the wide opening. This became especially true at very close ranges, such as at the points in the Verify phase where the robot would attempt to become perpendicular/parallel to the opening. At that close range, DETECTDOOR would loose the tracking of the door and eventually determine that it must have been a ghost door. Upon moving back to the middle of the room, the robot would again detect the doorway and approach to verify. It would take a number of repeated tries at this until, by chance, the verify phase occurred far enough away from the opening to allow DETECTDOOR to continue tracking it. Persistence does pay off, and the knowledge encoded within MOVETHROUGH performed well at re-attempting to verify what appeared to be the correct door. This is a good example of proper application of knowledge to improve overall performance.

X. The Use of Knowledge

The role or importance of the domain knowledge is extensive. The questions are: what knowledge is encoded? how is it encoded? how was it acquired? and what are its limitations? These issues are discussed next.

Domain Dependent vs. Domain Independent Knowledge. One of the goals of this research was to be able to separate the domain dependent knowledge from the domain independent knowledge. The Sensor Explication process implemented in the L-Controller class methods is domain independent, while the knowledge encoded in rules that is used by Sensor Explication is primarily domain dependent. The advantage of this division is that it separates the Sensor Explication process from the rest of the computation. This simplifies extending the architecture to other domains, tasks, and environments.

Proper Application of Knowledge Increases Performance. Failure is mostly due to limitations and weaknesses in the LSA's knowledge. Many LSAs have simple algorithms. Poorly designed LSAs resulted in a higher need for knowledge to compensate for the lack of performance. A good example is PROXIMITYSAFE. Its poor performance is evident by the amount of difficulty the robot experiences in moving away from obstacles. One of the capabilities experiments failed due to its inability to retreat. Another example is DETECTDOOR. The algorithm had troubles in detecting larger doorways. If not for the knowledge encoded in its controlling LSA, this weakness could have caused additional failures. The knowledge in the LSA dealt with monitoring for large doorways and applied various maneuvering strategies to facilitate the detection of the doorway as problems occurred. Because of this, the process of detecting and tracking the doorway through the entire execution of the goal succeeded, though progress was occasionally hampered by noise.

This truly emphasizes the importance for higher levels of control to have knowledge about the weakness and limitations of the lower levels. Areas containing knowledge performed very well compared to those lacking knowledge. The laboratory experiments showed that even simple algorithms can be combined with knowledge into an overall synergistic algorithm which is far more reliable than any of the individual components.

Knowledge Acquisition Through Experimentation. The success of the laboratory experiments showed that knowledge can be extracted from the domain and implemented in the system. The experiments also showed the difficulty in extracting the knowledge and structuring it to be properly utilized. At times a great deal of investigation would be needed to determine what aspect of the environment was causing a failure of a LSA or a poor coordination within a dataflow. Knowledge acquisition was the largest bottleneck during the experiments.

Limits of Domain Knowledge. The primary problem is the dependence of the system on the domain knowledge. Though the building of the domain knowledge seems straightforward, there is still a kind of "black magic" aspect about it. The main problem that

has plagued domain knowledge all along still remains. There must be domain knowledge for each determined scenario or discovered weakness. The resulting system is only as powerful as the scenarios the knowledge covers, and there is no guarantee that the knowledge encoded will work beyond the scenarios examined during knowledge acquisition. Thus, the amount of domain knowledge can be endless, and the question remains of "how much is enough". We have observed, as expected, that initially each increment of knowledge improved the performance greatly. However, as the total amount of knowledge increased, the performance did not increase by a comparable amount. The returns were diminishing. At some point the performance improvement is no longer worth the cost. The issue then becomes one of desired performance vs. cost. The issue can only be answered within each individual application domain (i.e. how many dollars is robustness worth to this application?).

XI. Related Research

There has been a large amount of research in the area of sensor-based control of a mobile robot. The most popular and successful approach is the subsumption architecture [8] which avoids planning altogether by using layers of behaviors. Behaviors run in parallel and interact when needed. Each individual layer corresponds to a level of behavioral competence. Though this method of control allows for many behaviors, it disallows central control and so it makes it difficult to assign to the robot different goals to be achieved.

Our method of "intelligent plan execution objects" blends well into the "behaviors" paradigm which dominates the autonomous robotics research today. Using LSAs, a behavior can be constructed from a set of objects, and controlled by a single object which corresponds to that behavior. Each of the sub-objects can attend to a different aspect of that behavior, such as producing information from the sensor data, or deciding what distance to move or turn, or deciding when the goal has been achieved and it is time to stop. In this sense our work is similar to the Perception and Motor Schemas by Arkin [9]. The main difference is in the way LSAa are activated. Control in Arkin's approach is achieved by a single control module that selects from a single layer of motor schemas. In our system knowledge stored in LSAs controls how LSAs are combined together. Different arbitration and task decomposition methods can be programmed in the LSAs, providing for more flexibility.

Many architectures have been proposed that combine the ability to react to the environment with planning. Georgeff [18] has implemented a planning and control system based on a functional decomposition of high-level control into primitives, and that integrates some reactive behaviors into the control structure. The Rex/Gapps system [19] decomposes high-level goals into mobile robot commands, and then converts them into hardware logic designs. The result is a large reactive system that has goal-oriented behavior but cannot plan or adapt itself to new goals.

The 3T architecture [20] has three levels: deliberative, sequencing, and reacting. The top level performs activities such as planning and world modeling, the middle level draws directly from the RAP system [21], while the lower level is a stateless reactive control mechanism which controls activities with no decision-making.

Firby's most recent work [1] extends his original RAP architecture by allowing the creation of independent reactive agents for different tasks, and the run-time selection of which one to use to achieve the next subtask. His "skills" correspond to our low level LSAa, and "skill networks" to our high-level LSAs. The primary difference is that our architecture is homogeneous. Everything is a Logical Sensor/Actuator, from the low-level drivers for sensors and actuators to the high-level LSAs for abstract goals. Common methods are shared by objects through inheritance. The ability to activate/deactivate objects and to change the dataflow among them in a dynamic way depending on the situation gives us a flexibility that other architectures do not have.

The Task Control Architecture (TCA) of Simmons [3] is another successful example of combining deliberative and reactive control. The TCA consists of a set of task-specific modules and of a central control module that coordinates their activities. The central control module routes messages dynamically to the other modules. The TCA supports the creation and execution of hierarchical plans, and facilitates incremental building of systems. The LSA architecture is more distributed and has no central controller.

A large number of experimental results obtained with various robots and tasks using the architectures described above and others not mentioned attests to the validity of developing flexible and reconfigurable architectures.

The LSA architecture borrows heavily from the idea of Logical Sensors [14], [22]. Logical Sensors are abstracted views of sensors and sensor processing, much like how logical I/O is used to insulate the user from the differences of I/O devices and operating systems. We have expanded the concept of Logical Sensors to include Logical Actuators, incorporated it into a comprehensive framework, and demonstrated experimentally the viability of the idea

There are also similarities between LSA and blackboard architectures [4]. Both hierarchically organize different components. The blackboard agent architecture's components are: the perception processes (similar in functionality to L-Sensors); the action systems (similar to L-Drivers); the reflex-arcs and perception-action coordination processes (similar to L-Generators) and the cognition system (similar to L-Controllers and L-Matchers). The main differences hinge on centralized vs distributed control and data storage. The blackboard architecture has a single cognition system containing multiple knowledge sources. The LSA distributes the control across multiple L-Controllers. The storage of data is centralized on the blackboard, while the LSA architecture requires data produced by a LSA to be stored within that LSA.

The CIRCA system [5] integrates planning with real-time control. Most of the work has focused on generating

plans in which the control actions are guaranteed to be executed within the real-time constraints. Our research has not addressed the issue of how to guarantee real-time performance. Our architecture separates planning from execution, and simplifies planning by increasing the level of intelligence of the execution system. The knowledge encoded in LSAs enables the execution system to be flexible and capable of taking advantage of the resources and situations available in achieving its goal.

XII. Conclusions

The Logical Sensor/Actuator architecture was developed as an approach to robotic execution of classical planning goals. This architecture addresses the need to handle noisy sensors and actuators, the need to remain reactive to changes in the environment while accomplishing goals, and, most important, the application of knowledge to deal with resolving failures.

The LSA architecture provides evidence that a goal can be achieved through proper application of knowledge. Specifically, this entails identifying (1) what information is relevant to achieving the goal; and (2) what are the sources of this information (i.e. sensors, actuators and processes), and how to control them. Extensive lab experiments demonstrate that the architecture is reliable and robust, despite sensor noise, actuator errors, and even limits in the knowledge encoded in the LSA objects.

Acknowledgments

We would like to thank Scott Brandt, John Fischer, Chris Smith, Karen Sutherland, and Eric the Red for their help, and the anonymous reviewers for their suggestions that greatly improved this paper. All research described here was performed at the Artificial Intelligence, Robotics and Vision Laboratory of the Department of Computer Science, University of Minnesota.

References

[1] R. James Firby, "Task networks for controlling continuous actions," in *Proc. Artificial Intelligence Planning Systems*, 1994.

[2] D. Lyons, "Representing and analyzing action plans as networks of concurrent processes," *IEEE Transactions on Robotics and Automation*, vol. RA-9, no. 3, pp. 241–256, June 1993.

[3] R. Simmons, "Structured control for autonomous robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 1, pp. 34–43, February 1994.

[4] Barbara Hayes-Roth, "An architecture for adaptive intelligent systems," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 329–365, 1995.

[5] David Musliner, Edmund Durfee, and Kang Shin, "World modeling for the dynamic construction of real-time control plans," *Artificial Intelligence*, vol. 74, no. 1, pp. 83–127, 1995.

[6] D. E Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley, "Planning and reacting in uncertain and dynamic environments," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 6, pp. 197–227, 1994.

[7] J. S. Albus, "Hierarchical interaction between sensory processing and world modeling in intelligent systems," in *5th IEEE Symp. on Intelligent Control*, 1990, pp. 53–59.

[8] Rodney Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, March 1986.

[9] R. C. Arkin, "Motor schema-based robot navigation," *International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, Aug. 1989.

[10] D. W. Payton, J. Kenneth Rosenblatt, and David M. Keirsey, "Plan guided reaction," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 6, pp. 1370–1382, 1990.

[11] E. Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots," in *Proc. Nat'l Conf. on Artificial Intelligence*, 1992, pp. 809–815.

[12] J. Budenske and M. Gini, "Why is it so difficult for a robot to pass through a doorway using ultrasonic sensors?," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 3124–3129.

[13] E. Gat, "Robot navigation by conditional sequencing," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 1293–1299.

[14] T. Henderson and E. Shilcrat, "Logical sensor systems," *Journal of Robotics*, vol. 1, no. 2, pp. 169–193, 1984.

[15] E. Gat, "Robust low-computation sensor-driven control for task-directed navigation," in *IEEE International Conference on Robotics and Automation*, 1991, pp. 2484–2489.

[16] John Budenske and Maria Gini, "Knowledge, execution, and what sufficiently describes sensors and actuators," in *SPIE OE/Technology '92*, 1992.

[17] John Budenske, *Robotic Plan Execution in Dynamic and Unpredictable Environments*, Ph.D. thesis, University of Minnesota, 1993.

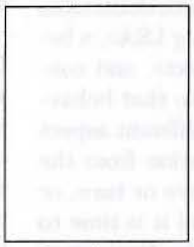
[18] Michael P. Georgeff and Amy L. Lansky, "Reactive reasoning and planning," in *Proc. Nat'l Conf. on Artificial Intelligence*, Seattle, WA, 1987, pp. 677–682.

[19] Leslie P. Kaelbling and Stan Rosenschein, "Acting and planning in embedded agents," *Robotics and Autonomous Systems*, vol. 6, pp. 35–48, 1990.

[20] R. P. Bonasso, H. Antoinisse, and M. G. Slack, "A reactive robot system to find and fetch tasks in an outdoor environment," in *Proc. Nat'l Conf. on Artificial Intelligence*, 1992.

[21] R. James Firby, "An investigation into reactive planning in complex domains," in *Proc. Nat'l Conf. on Artificial Intelligence*, Seattle, 1987, pp. 202–206.

[22] T. Henderson, C. Hansen, and B. Bhanu, "The specification of distributed sensing and control," *Journal of Robotic Systems*, vol. 2, no. 4, pp. 387–396, April 1985.



John Budenske received his B.S., M.S, and Ph.D. degrees in Computer Science from the University of Minnesota in 1983, 1987, and 1993, respectively. He worked at the Honeywell Systems and Research Center on artificial intelligence, signal image processing, robotics, simulation, and database systems. He then worked on advanced methods for software reengineering and object oriented programming at Loral Defense Systems. Currently, he is performing research in intelligent control, software engineering, electronic collaboration, and intelligent agents for Architecture Technology Corporation in Minneapolis, Minnesota.

Maria Gini is an Associate Professor in the Department of Computer Science at the University of Minnesota, in Minneapolis. She has been a Research Associate in the School of Engineering, Politecnico of Milan, Italy and a Visiting Research Associate at the Artificial Intelligence Laboratory at Stanford University. Her research interests are in integrating Artificial Intelligence with robotics for navigation, exploration, learning of mobile robots, parallel algorithms for real-time path planning for articulated robots, task planning with incomplete or uncertain information, detection and recovery from robot errors, object-oriented programming for robotics.