SEEK: ACCOMPLISHING ENTERPRISE INFORMATION INTEGRATION ACROSS HETEROGENEOUS SOURCES

SUBMITTED: January 2002
REVISED: July 2002.
PUBLISHED: August 2002 at http://www.itcon.org/2002/7
EDITOR: Abdul Samad Kazi

William J. O'Brien, Assistant Professor,
M.E Rinker, Sr. School of Building Construction, University of Florida,
email: wjob@ufl.edu, http://www.bcn.ufl.edu

Raja R.A. Issa, Professor,
M.E. Rinker, Sr. School of Building Construction, University of Florida
email: raymond-issa@ufl.edu, http://www.bcn.ufl.edu

Joachim Hammer, Assistant Professor
Dept. of Computer & Information Science & Engineering, University of Florida
email: jhammer@cise.ufl.edu, http://www.cise.ufl.edu

Mark S. Schmalz, Research Professor
Dept. of Computer & Information Science & Engineering, University of Florida
email: mssz@cise.ufl.edu, http://www.cise.ufl.edu

Joseph Geunes, Assistant Professor
Dept. of Industrial & Systems Engineering, University of Florida
email: geunes@ise.ufl.edu, http://www.ise.ufl.edu

Sherman X. Bai, Associate Professor
Dept. of Industrial & Systems Engineering, University of Florida
email: bai@ise.ufl.edu, http://www.ise.ufl.edu

SUMMARY*: This paper describes ongoing research on the Scalable Extraction of Enterprise Knowledge (SEEK) project. The SEEK toolkit is a collection of modular components. The components enable rapid instantiation of connections to firms' legacy information sources, (semi)-automatically integrating knowledge in the firm with knowledge needed as input to decision support tools. SEEK is not a general-purpose toolkit. Rather, it allows extraction of knowledge required by specific types of decision support applications. Thus SEEK enables scalable implementation of computerized decision and negotiation support across a network of firms. Current development is directed towards support for construction supply chain applications.*
SEEK represents a departure from research and development in shared data standards. Instead, SEEK embraces heterogeneity in firms' information systems, providing the ability to extract and compose knowledge resident in sources that vary in the way data is represented and how it can be queried and accessed. This paper outlines the business needs for such capabilities, the SEEK information architecture, and reviews the underlying technologies (principally, Data Reverse Engineering) supporting SEEK.

KEYWORDS*: legacy system integration, knowledge capture, knowledge composition, data reverse engineering, supply chain management, process models.*

1. INTRODUCTION

Our vision is to enable computerized decision and negotiation support among the extended network of firms composing the construction supply chain. Recent research has led to an increased understanding of the importance of coordination among subcontractors and suppliers (Howell and Ballard 1997; Vrijhoef and Koskela 1999). There is a role for decision or negotiation support tools to improve supply chain performance, particularly with regard to the user’s ability to coordinate pre-planning and responses to changed conditions (O'Brien et al. 1995).

Deployment of these tools requires integration of data and knowledge across the supply chain. Due to the heterogeneity of legacy systems, current integration techniques are manual, requiring significant programmatic set-up with only limited reusability of code. The time and investment needed to establish connections to sources has acted as a significant barrier to adoption of sophisticated decision support tools and, more generally, as a barrier to information integration in construction. By enabling (semi-)automatic connection to legacy sources, the SEEK (Scalable Extraction of Enterprise Knowledge) project and associated toolkit is directed at overcoming the problems of integrating legacy data and knowledge in the construction supply chain.

An important assumption underlying development of SEEK is that there is and will continue to be significant heterogeneity in legacy sources. Thus while the SEEK project does support a long-held goal for computer-integrated construction (e.g., Brandon et al. 1998; Teicholz and Fischer 1994), it represents a significant departure from much current work in developing shared data standards and information models. Rather, the SEEK approach embraces a world where there are numerous data models that coexist to support the differing applications and views of project participants. The SEEK approach is much in the spirit of development foreseen by Turk (2001), who notes that it may be easier to develop translators between data models than it is to develop a unifying data model.

2. MOTIVATION AND CONTEXT

The SEEK information architecture has the dual abilities to (1) securely extract data and knowledge resident in physically and semantically heterogeneous legacy sources and (2) compose that knowledge to support queries not natively supported by the sources. This section describes the motivation for these capabilities and the context within which SEEK will operate. This motivation and context has driven development of SEEK capabilities and choices about implementation. Subsequent sections detail the SEEK architecture and underlying methodologies.

2.1 Business needs for information integration and SEEK

Motivation for SEEK stems from the need for coordination of processes across multiple firms (and hence, from a technical need to integrate distributed process related information). Consider that any given project may have dozens of subcontractors; in turn, each subcontractor may have several suppliers. The large number of firms involved in a project requires continual coordination and re-coordination of processes to ensure timely production and adequate allocation of resources. Problems due to poor coordination (such as scheduling conflicts, materials shortages, etc.) are well documented in the construction literature (e.g., Ballard and Howell 1998; Bell and Stukhart 1987; Halpin 1993; O'Brien 1997; Wegelius-Lehtonen and Pahkala 1998).

It is the goal of the SEEK project to automate assembly of process related information for subsequent analysis by decision support tools (and hence, facilitate improved processes on projects). Figure 1 depicts the information environment of SEEK. There are many firms (principally, subcontractors and suppliers), and each firm contains legacy data used to manage internal processes. This data is also useful as input to a project level decision support tool. However, the large number of firms on a project makes it likely that there will be a high degree of physical and semantic heterogeneity in their legacy systems, making it difficult to connect firms’ data and systems with enterprise level decision support tools. It is the role of the SEEK system to act as an intermediary between firms’ legacy data and the decision support tool. Note that Figure 1 shows the tool linked to a coordinating firm such as a general contractor. This may be appropriate for a scheduling decision support tool. In other applications such as detailed production analysis (e.g., Tommelein and Ballard 1997), firms such as larger subcontractors may play a coordinating role and host the tool.

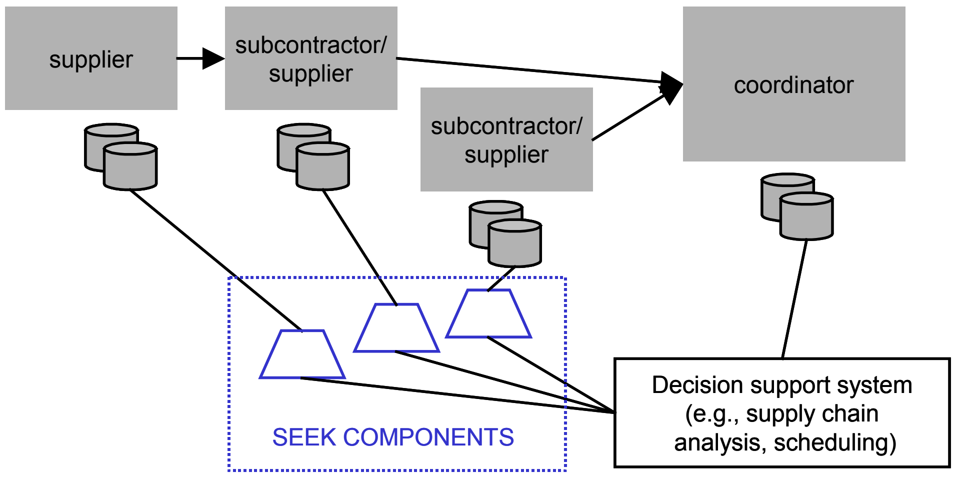


FIG.1: Information environment of SEEK

2.1.1 SEEK as a narrow data extraction and composition tool for decision support applications

SEEK is not intended to be a general-purpose data extraction tool. SEEK does extract a narrow range of data and knowledge from heterogeneous sources to support a class of decision support applications. For example, the construction literature and related literature in manufacturing describe a growing number of process models to improve coordination among firms in the construction supply chain. These models range from scheduling and supply extensions (e.g., Shtub 1988) to analytic models of supply chains (e.g., Thomas and Griffin 1996). All of these models need similar data (largely resource scheduling, productivity, and cost data) to operate as a useful decision or negotiation support tool. Current instantiations of SEEK are built to extract the limited range of information needed by these process models to support their use as tools.

Beyond extraction of data, SEEK is intended to perform knowledge composition. Consider that much of the data used for operations in firms is detailed in nature, often mimicking accounting details or a detailed work breakdown structure (Barrie and Paulson 1992). This data is too detailed for most decision support models (see for example, the supply chain models in Tayur et al. (1999)). SEEK composes the data needed as input for analysis tools from data used by applications with other purposes in mind. (Data composition tasks are reviewed in section 5 of this paper and a related example is presented in O'Brien and Hammer (2001)).

2.1.2 High-level functional requirements

The information environment of Figure 1 and capabilities reviewed above suggest several challenges that translate to high-level functional requirements for SEEK:

- Rapid deployment: The project supply chain comprises a large number of firms and is assembled quickly. The system must be deployed swiftly with limited human interaction.
- Connect to heterogeneous sources: The large number of firms on a project suggests that associated legacy sources will not subscribe to uniform data standards but rather present a high degree of heterogeneity both physically and semantically. SEEK must accept a wide range of source types.
- Composition of data: Concomitant with heterogeneous information representation, there is a need to compose or mediate data stored in a firm's legacy sources to support the information needs of decision support applications.
- Security: Firms will generally be unwilling to make their legacy systems open for general examination by other firms. The system must filter the data extracted from the underlying sources.

These functional requirements are reflected in the design of the SEEK architecture presented in section 3.

2.2 SEEK and its relationship to efforts in data standards

The SEEK approach to integrating extended enterprise information differs from the approach of recent academic and commercial work developing data standards such as the Industry Foundations Classes (IFC) (IAI 1996) and aecXML (aecXML 1999). A core assumption driving development of SEEK is that the multiple firms composing a project will not subscribe to a common data standard for process related data. The large number of firms in the construction supply chain (easily hundreds and perhaps thousands on large projects) makes it implausible that all firms will uniformly subscribe to a common standard. As argued in O'Brien and Hammer (2001), it seems more likely that firms will maintain use of legacy applications for process data, selectively transitioning to new applications. Our view is similar to those of Amor and Faraj (2001), Turk (2001), and Zamanian and Pittman (1999). These authors argue that a single integrated project database and supporting data standard will be not be able to contain or reconcile the multiple views and needs of all project participants. They predict that rather than a single standard, multiple protocols will evolve over time. Each protocol will be suitable for use by different disciplines at a specific phase of the project. The development of trade group standards such as the CIMsteel Integration Standards (http://www.cis2.org/) supports their views. It should be noted that CIMsteel and the IFC, while each developed from STEP constructs, are not interoperable. Considerable future development work is required to make them so (Crowley 1999).

SEEK also differs from existing efforts as it is focused on process as opposed to product. Zamanian and Pittman (1999) note that process models are not well integrated with much of the research in data standards as that work has focussed on product models. While limited process extensions have been developed for the IFC, current tests suggest that these extensions do not adequately address the needs of the process modelling community (Froese et al. 1999; Staub-French and Fischer 2000). We do not argue that the IFC and related standards are not extensible to process modelling, but rather that the current limitations and traditional divisions between the process and product modelling communities suggest that there will continue to be heterogeneity of applications and data formats in practice. The Process Specification Language (PSL) (http://www.mel.nist.gov/psl/) developed by NIST also suggests that there will be a range of process models. While PSL is highly descriptive, it is not envisaged that it will become a standard but rather it will be used as a kind of process lingua franca for translation between models developed in heterogeneous applications (i.e., rather than application one translated directly to application two, the translation would be application one to PSL to application two). In much the same sense, SEEK will translate process information from one source and translate to another.[1]

With a focus on processes, SEEK is not meant to be a replacement for product model data standards. Nor are SEEK and specifications like the IFC mutually exclusive; in a world with multiple protocols for different applications, many different applications and languages can coexist. But SEEK does represent a paradigm shift from a single data model for a single application (and, more broadly, a shift from a single data model for the project). SEEK is designed to operate in a world where there is heterogeneity of data models that store data related to a class of business/decision support problems. SEEK provides abilities to extract and compose the narrow range of data and knowledge related to that class, overcoming the problems imposed by heterogeneity in information representation.

3. SEEK ARCHITECTURE

In this section a high-level architectural view of SEEK is presented, relating functional capabilities to the business context described in section 2. Sections 4 & 5 describe the theory and methods underlying SEEK components and a sample application of SEEK data extraction and composition.

3.1 SEEK functional components

A high-level view of the SEEK architecture is shown in Figure 2. SEEK follows established mediation/wrapper methodologies (e.g., TSIMMIS (Chawathe et al. 1994), InfoSleuth (Bayardo et al. 1996)) and provides a software middleware layer that bridges the gap between legacy information sources and decision makers/decision support applications. This is seen in Figure 2, where there are three distinct, intercommunicating layers: (1) an information Hub, that provides decision support and that mediates communication with multiple

[1] It is important to note the PSL cannot automatically discover the information model in a source and develop the necessary translations. All translations between PSL and a source language must be developed by experts. SEEK, on the other hand, provides for automatic discovery of source data formats. Future developments of SEEK may make use of PSL in its internal data format.

firms; (2) the SEEK connection tools consisting of the Analysis Module (AM), Knowledge Extraction Module (KEM), and Wrapper; and (3) the Firm containing legacy data and systems.

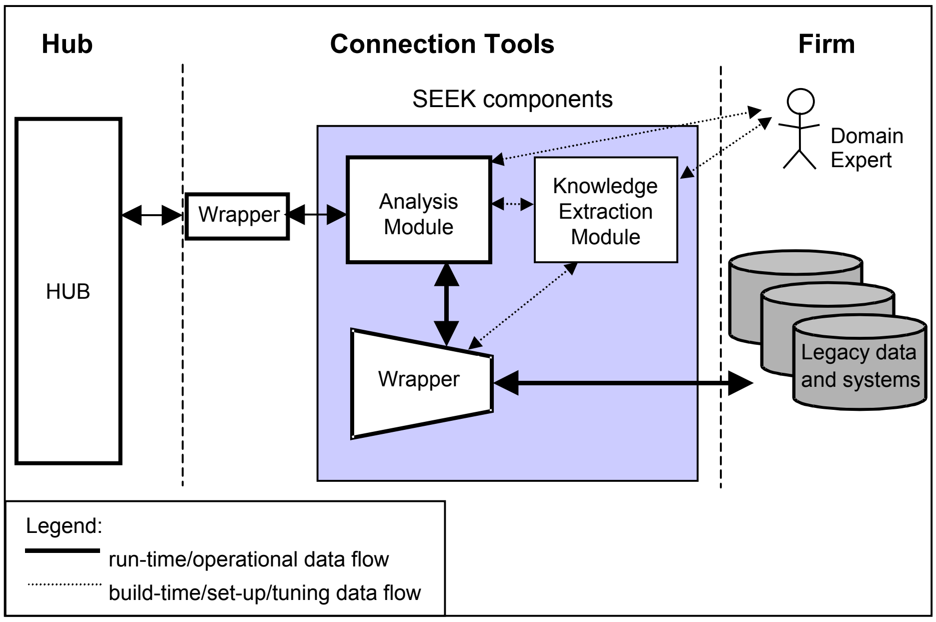


FIG.2: Schematic diagram of the conceptual architecture of the SEEK system and related components

The Hub provides decision support for the extended enterprise of firms (e.g., the construction supply chain containing contractors and suppliers). Information for decision support is gathered from firms through the SEEK connection tools. Specifically, the Hub connects to the Analysis Module, which performs knowledge composition or mediation tasks (Wiederhold 1998) on legacy data extracted from the Firm. As the AM does not cache data, it maintains a real-time connection with the Wrapper, which translates between the data representation and query formalism of the Hub/AM and the underlying source(s).

It is important to note the function of the wrapper between the Hub and the Analysis Module (Figures 2 & 3). There may be multiple Hubs with associated diversity among their internal data formats and languages. The wrapper enables translation between the data format of the Hub and the AM. As SEEK is limited to specific forms of data extraction derived from the decision support capabilities provided by a class (e.g., supply chain analysis) of Hubs, it is not envisioned that wrappers between the Hub and AM will be complex or difficult to implement. Indeed, provision for a wrapper allows support for multiple Hubs, increasing the scalability of the SEEK components.

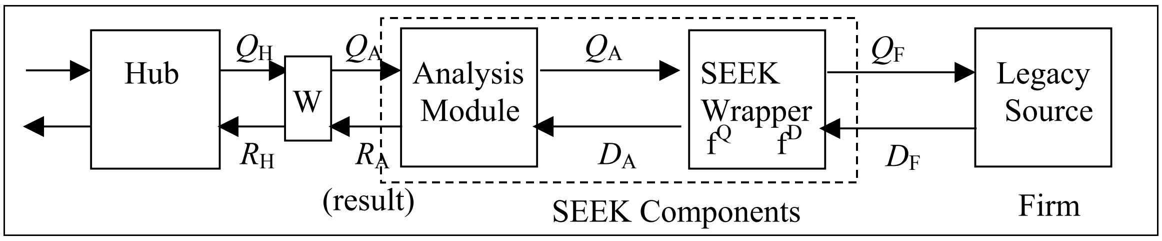


FIG.3: Overview of the interactions between Hub, SEEK components and Firm

The interactions between the Hub, SEEK components and Firm are summarized in Figure 3. At runtime, (i.e., after the SEEK wrapper and analysis module have been configured), the AM accepts a query issued by the Hub (Q_H in Figure 3) that has been converted into a query (Q_A) that the AM can understand using the wrapper W. The AM processes the Hub request and issues one or more queries (Q_A) to the SEEK wrapper to obtain the relevant legacy data needed to satisfy the Hub's request. The SEEK wrapper produces one or more queries (Q_L) in a format that the legacy source can understand. The legacy source processes Q_L and returns legacy data (D_L) that is transformed by the SEEK wrapper into data (D_A) that is tractable to the AM. The AM then processes this data and returns a result (R_H) via the wrapper W to the Hub that fulfils the original Hub query (Q_H).

As SEEK tools must be instantiated for each firm, it is important to provide rapid configuration with minimal human support. Instantiation is accomplished semi-automatically during build-time by the knowledge extraction module (KEM) that directs wrapper and analysis module configuration. The SEEK wrapper must be configured with information regarding communication protocols between SEEK and legacy sources, access mechanisms, and underlying source schemas. The analysis module must be configured with information about source capabilities and available knowledge and its representation. To produce a SEEK-specific representation of the operational knowledge in the sources, domain specific templates are used to describe the semantics of commonly used structures and schemas. The KEM queries the legacy source using the initial instantiation of a simple (generic) SEEK wrapper. Using data reverse engineering (DRE) techniques, the KEM constructs a representation of the legacy schema. The representation includes the semantics of the legacy data determined, for example, from queries, data samples, application code, and user input (as required). Using an iterative, step-wise refinement process the KEM constructs the mappings (f^Q and f^D) which extend the access and extraction capabilities of the initial wrapper to perform the aforementioned query and data translations, as shown schematically in Figure 3. A wrapper generation toolkit (Hammer et al. 1997a) is used to implement the customized wrappers.

Configuration of SEEK components is assisted by a domain expert (i.e., a manager familiar with the data in and use of the firm's legacy systems) to extend the capabilities of the initial, automatic configuration directed by the templates in the knowledge extraction module. Use of domain experts in template configuration is particularly necessary for poorly documented database specifications often found in older legacy systems. Input from a domain expert need not be performed for initial configuration of SEEK components, nor is configuration limited to a single set-up period. Domain experts can refine the quality of the data mappings over time, effectively expanding the scope and quality of data extraction and knowledge composition as needed.

3.2 SEEK and security of legacy data

The SEEK architecture allows secure, privacy constrained filtering of legacy data in two ways. First, as an intermediary between Hub and Firm, it restricts access to firm data and answers only specific queries. For example, the analysis module may respond to a query about resource availability on given dates with a simple affirmative or negative answer. The Hub (and whoever operates the Hub such as a general contractor) need never see the firm's raw data concerning resource allocation to other projects. As firms are justifiably reluctant to share details concerning operations, costs, etc., the existence of an intermediary between their legacy information and requester is both an important source of security and is an enabler for information integration in practice.[2]

A second aspect of security is that the firm can limit access to the legacy data available to the SEEK wrapper component. This is shown in Figure 4, which depicts an access layer between legacy sources and the wrapper. The access layer controls the overall permissions and availability of data. It is likely that the access layer will not be SEEK specific but be a general directory services platform that controls both internal and external access to a firm's information systems (Economist 2001). As access levels can change, for each change it will be necessary to use SEEK's KEM to reconfigure the wrapper and AM. Reconfiguration can use knowledge generated during previous configuration, speeding setup tasks and increasing accuracy.

Based on discussion in section 2.2, it is useful to note that Figure 4 depicts a range of legacy sources in the firm, including both applications built on industry standards such as the Industry Foundation Classes and other legacy applications (e.g., built in-house, based on proprietary software, etc.). SEEK can query both types of legacy sources, making use of data standards where available. Moreover, as the access layer allows multiple classes of applications to query firm data, SEEK can coexist with other forms of applications that make use of data standards (such as shared CAD/product modelling applications).

[2] This suggests two ways SEEK components may be implemented: First, by firms themselves as their primary query mechanism/interface to information hubs. Second, by a trusted third-party provided operating in an application service provider (ASP) model.

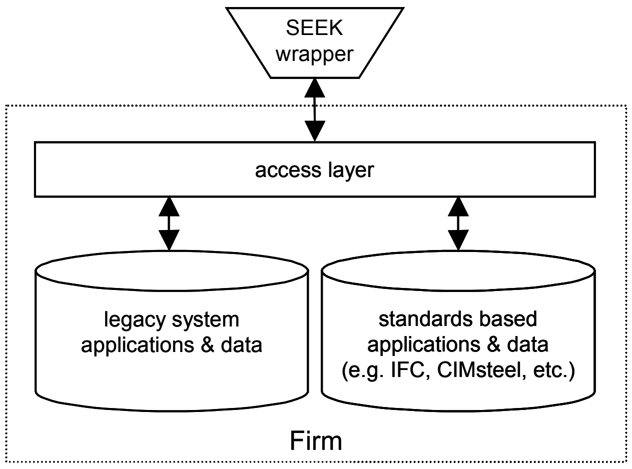

SEEK wrapper
access layer
legacy system applications & data
standards based applications & data (e.g. IFC, CIMsteel, etc.)
Firm

FIG.4: SEEK wrapper and relationship to firm's legacy sources

4. DATA REVERSE ENGINEERING AND WRAPPER TECHNOLOGIES IN SEEK

Development of SEEK is based on two major areas of research: data reverse engineering and source wrapper configuration. This section reviews the past and current research in this area.

4.1 Data Reverse Engineering

Software reverse engineering (SRE), a well-known practice in computer science, has as its goal the inference of program structure and functionality from compiled source code. Similarly, *data reverse engineering* (DRE) refers to the inference of structure and meaning (e.g., schema, relations, and semantics) from databases. As SRE is primarily applied to legacy source code, DRE techniques have been applied to legacy source data. For example, Aiken (1996) aptly defines DRE as "…the use of structured techniques to reconstitute the data assets of an existing system." The emphasis on structured techniques is key to economically feasible DRE implementations, which attempt to add value to existing data assets by increasing their consistency and ease of use within or across organizations (Davis and Aiken 2000).

Industrial legacy database applications (LDAs) often evolve over several generations of developers, have hundreds of thousands of lines of associated application code, and maintain vast amounts of data. In many cases, the documentation has become obsolete and the original developers have left the project. In SEEK, a major task of DRE is recovery of LDA conceptual structure that is often based on a relational database. Unfortunately, the simplicity of the relational model does not support direct description of the underlying semantics, nor does it support inheritance, aggregation, *n*-ary relationships, or time dependencies including design modification history. However, relevant information about concepts and their meaning is distributed throughout an LDA. For example, procedural code, database schema, parameter values can be extracted from data or obsolete documentation, and expertise of the system users or designers (where available) can be collected.

Davis and Aiken (2000) partition a major portion of the DRE literature into three areas: translation algorithms and methodologies, tools, and application-specific experiences. Translation algorithm development in early DRE efforts involved manual rearrangement or reformatting of data fields, which was inefficient and error-prone (Davis and Aiken 2000). Publication of the relational data model (Codd 1970) provided theoretical support for research in automated discovery of relational dependencies (Casanova and Sa 1983; Silva and Melkanoff 1979). In the early 1980s, focus shifted to translation of relations to E/R diagrams (Dumpala and Arora 1981; Melkanoff and Zaniolo 1980). Given the early successes with translation using the relational data model, DRE translation techniques were applied to flat file databases (Casanova and Sa 1983; Davis and Arora 1985) within domains such as enterprise schemas (Klug 1980). The aforementioned problem of re-engineering legacy code to reveal data relationships and database schema was discussed by Nilsson in the context of COBOL code (Nilsson 1985).

Although DRE translation algorithms were usually based on a homogeneous data model, DRE methodologies were applied to heterogeneous data models, for example, in the context of a semantic data model that involves simple transformation of the data without revealing its high-level semantics (Navathe and Awong 1988).

Alternatively, the relational model only could be used as input (Johannesson and Kalman 1989). Due to previous establishment of the E/R model as a conceptual tool, reengineering of legacy relational database systems to produce E/R models became a focus of DRE in the late 1980s (Davis and Arora 1987). Additionally, information content analysis was applied to databases, allowing more effective access to gather higher-level information from data (Boulanger and March 1989).

DRE in the 1990s was enhanced by cross-fertilization with software engineering. Chikofsky (1990) developed a taxonomy for reverse engineering that includes DRE methodologies and also highlights available DRE tools. DRE formalisms were better defined, and the focus began to shift toward DRE's interaction with the human user (Hainaut 1991). The emphasis continued to be on the relational data model, in particular, extraction of E/R and schema from relational databases (Chiang et al. 1994; Markowitz and Makowsky 1990; Song and Froehlich 1995). Applications focus continued to be placed on legacy systems, including DoD applications (Aiken et al. 1994). Research in DRE tools proliferated, resulting in systems such as August-II (data model reverse engineering (Davis 1995)), DB-MAIN (programmable CASE tool for database engineering (Englebert and Hainaut 1999)), and tools for translation of relational databases (Chiang et al. 1994).

An important trend in software engineering research is the use of program analysis or program comprehension. The original goal was to help programmers understand how existing programs work and how they would be affected by proposed modifications, but applications to reverse engineering are straightforward. Several approaches have been proposed for program comprehension and in the last few years there has been a considerable effort to automate the same. The most important techniques include *program slicing* (Horwitz and Reps 1992), *cliché recognition* (Wills 1994), and *pattern matching* (Paul and Prakash 1994), besides the more conventional approaches of lexical and syntactic analysis. Slicing is a data flow analysis derivative that helps understand what an existing program does and how it works by reducing the available code to only those lines that manipulate a certain set of program points of interest (e.g., input and output variables and their dependants).

Clichés are commonly used computational structures in programs. Examples of clichés include: list enumeration, binary search, and common data structures such as hash table and priority queues. Since clichés have well–known properties and behaviour, cliché recognition allows an experienced programmer to reconstruct the program's design and comprehend it. Commonly occurring programming patterns are encoded in terms of data and control flow constraints and stored in a cliché library. Recognition is achieved by parsing the flow graph of the program in accordance to the language grammar. Attribute and constraint checking with the cliché library is interleaved with the parsing.

Pattern matching is a technique that identifies interesting patterns and their dependencies in the code. For example, conditional control structures such as `if..then..else`, or `case` statements may encode business knowledge, whereas data type declarations and class or structure definitions can provide valuable information about the names, data types, and structure of concepts represented in an underlying database. Interesting programming patterns are stored in templates using a so-called 'pattern language'. Pattern matching works by transforming both the source code and the pattern templates into syntax trees. A code pattern recognizer performs the matching of patterns from the templates against patterns in the source code. Coupled with *program dependence graphs*, a language independent program representation, slicing, cliché recognition, and pattern matching are valuable tools for extracting semantic information from application code.

In the late 1990s, object-oriented DRE was explored in terms of discovering objects in legacy systems using function-, data-, and object-driven objectification (Wiggerts et al. 1997). Applications of DRE continued to grow, particularly in identification and remediation of the Y2K bug. The recent focus of DRE is more applicative, for example, mining of large data repositories (Dayani-Fard and Jurisica 1998), analysis of legacy systems (Hensley and Davis 2000) or network databases (Moh 2000), and extraction of business rules hidden within legacy systems (Shao and Pound 1999). Current research in the area focuses on developing powerful DRE tools, refining heuristics to yield fewer missing constructs, and developing techniques for reengineering legacy systems into distributed applications. Development of SEEK emphasizes understanding of legacy semantics, as discussed in Section 5.2.

4.2 Wrapper Technology

The SEEK *wrapper* (Papakonstantinou et al. 1995; Rashid 1997) accepts queries expressed in the legacy source language and schema and converts them into queries or requests understood by the source. When a result is being returned by the source, the wrapper converts the data from the source format into a representation consistent with the schema corresponding to the Analysis Module information model, which is expressed using XML. The mappings that underlie query and data transformations are determined by the SEEK DRE module.

One can identify several important commonalties among wrappers for different data sources, which make wrapper development more efficient and allow the data management architecture to be modular and highly scalable. These are important prerequisites for supporting numerous legacy sources, many of which have parameters or structure that could initially be unknown. Thus, the wrapper development process must be partially guided by human expertise, especially for non-relational legacy sources.

A naïve approach involves hard-coding wrappers to effect a pre-wired configuration, thus optimizing code for these modules with respect to the specifics of the underlying source. However, this yields inefficient development, with poor extensibility and maintainability. Instead, the SEEK development effort employs an implementational toolkit such as Stanford University's TSIMMIS Wrapper Development Toolkit (Hammer et al. 1995; Papakonstantinou et al. 1995), which is based on translation templates written in a high-level specification language. Using the TSIMMIS toolkit, Hammer has developed value-added wrappers for sources such as DBMS, online libraries, and the Web (Hammer et al. 1997a; Hammer et al. 1997b; Hammer et al. 1997c).

Existing wrapper development technologies exploit the fact that wrappers share a basic set of source-independent functions that are provided by their toolkits. For example, in TSIMMIS, all wrappers share a parser for incoming queries, a query processor for post-processing of results, and a component for composing the result. Source-specific information is expressed as templates written in a high-level specification language. Templates are parameterized queries together with their translations, including a specification of the format of the result. Thus, the TSIMMIS researchers have isolated the only component of the wrapper that requires human development assistance, namely, the connection between the wrapper and the source, which is highly specialized yet requires relatively little coding effort.

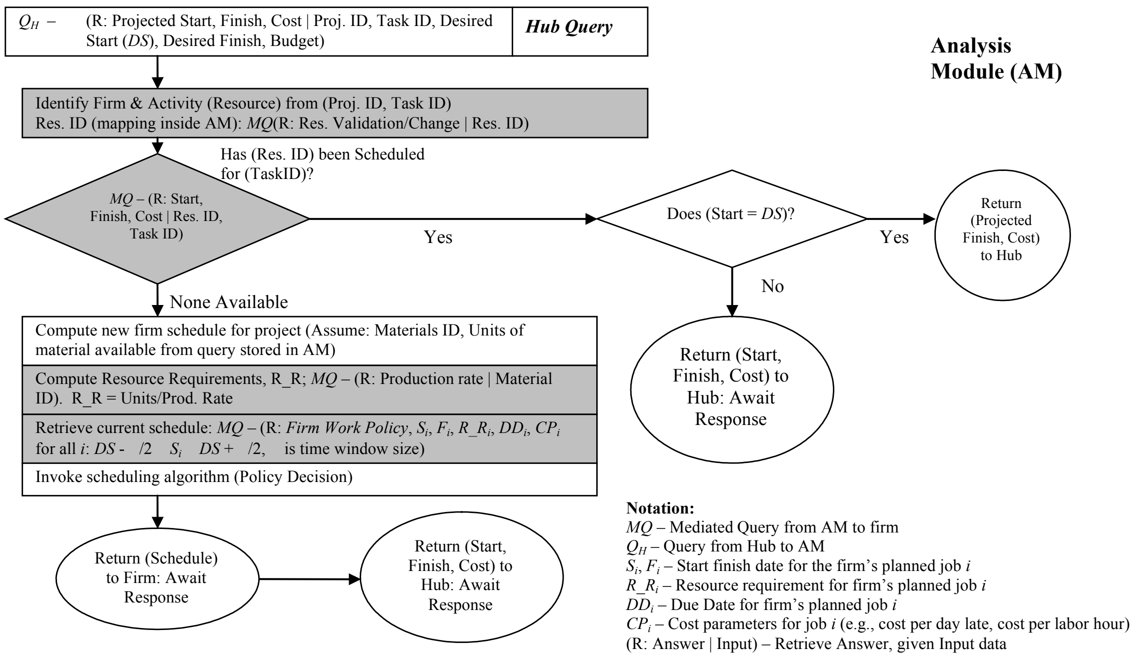
In addition to the TSIMMIS-based wrapper development, numerous other projects have been investigating tools for wrapper generation and content extraction including researchers at the University of Maryland (Gruser et al. 1998; Rashid 1997) (funded under DARPA's I3 program), USC/ISI (Ashish and Knoblock 1997; Gruser et al. 1998; Sahuguet and Azavant 1998), and University of Pennsylvania (Sahuguet and Azavant 1998). In addition, artificial intelligence (Kushmerick et al. 1997), machine learning, and natural language processing communities (Califf and Mooney 1998; Mooney 1999) have developed methodologies that can be applied in wrapper development toolkits to infer and learn structural information from legacy sources.

5. EXAMPLE AND DESCRIPTION OF SEEK OPERATION

In this section, SEEK operations to extract legacy data and compose a result in response to a query from the Hub are described in detail.

5.1 Example: Data Extraction and Composition for Scheduling

In addition to extracting data from legacy systems and drawing inferences about relationships between and among the data, SEEK adds value through project scheduling decision support. This decision support functionality takes place in the Analysis Module (AM), which takes the data extracted from legacy systems as input to optimization-based scheduling algorithms. This section describes the how the AM interacts with the Hub and legacy systems via a set of queries in order to provide interactive decision support capabilities between the Hub and various firms involved in a project.



Q_H – (R: Projected Start, Finish, Cost | Proj. ID, Task ID, Desired Start (DS), Desired Finish, Budget)
Hub Query
Analysis Module (AM)
Identify Firm & Activity (Resource) from (Proj. ID, Task ID)
Res. ID (mapping inside AM): MQ(R: Res. Validation/Change | Res. ID)
Has (Res. ID) been Scheduled for (TaskID)?
MQ – (R: Start, Finish, Cost | Res. ID, Task ID)
Yes
Does (Start = DS)?
Yes
Return (Projected Finish, Cost) to Hub
No
Return (Start, Finish, Cost) to Hub: Await Response
None Available
Compute new firm schedule for project (Assume: Materials ID, Units of material available from query stored in AM)
Compute Resource Requirements, R_R; MQ – (R: Production rate | Material ID). R_R = Units/Prod. Rate
Retrieve current schedule: MQ – (R: Firm Work Policy, S_i, F_i, R_R_i, DD_i, CP_i for all i: DS - /2 S_i DS + /2, is time window size)
Invoke scheduling algorithm (Policy Decision)
Return (Schedule) to Firm: Await Response
Return (Start, Finish, Cost) to Hub: Await Response
Notation:
MQ – Mediated Query from AM to firm
Q_H – Query from Hub to AM
S_i, F_i – Start finish date for the firm's planned job i
R_R_i – Resource requirement for firm's planned job i
DD_i – Due Date for firm's planned job i
CP_i – Cost parameters for job i (e.g., cost per day late, cost per labor hour)
(R: Answer | Input) – Retrieve Answer, given Input data

FIG.5: Schematic of query in Analysis Module

To illustrate the types of automated decision support SEEK can provide, consider a scenario in which a decision support tool (i.e., the Hub) is directed to construct a project schedule based on its knowledge of the various activities required by the project as well as the project's desired start and finish dates and budget. Within this scenario, a schematic of the flow of information and sequence of events between a query by the Hub to the Firm is provided in Figure 5. Each project is assigned a unique project identification number (PID). Associated with a PID is a set of tasks required for completion of the project, and each task contains a unique task identification number (TID). In order to initiate project scheduling at the Hub, for each TID, the Hub must first determine the scope of possible start and finish times and the costs associated with any unique start and finish time pair. The scope of possible start/finish times and task costs is ascertained through a request from the Hub to the AM of the form

(Request: Start, Finish, Cost | PID, TID, Desired Start, Desired Finish, Budget).

In this hub query, the parameters PID, TID, Desired Start, Desired Finish and Budget are provided by the hub (input parameters); the parameters Start, Finish, and Cost must be computed by the AM (output parameters). (For clarity the actual query, which is represented as an XML document, is not shown in Figure 5). The above query triggers several actions in the AM, beginning with the identification of the unique firm and activity (or resource) associated with the unique (PID, TID) pair. The unique firm and activity are mapped to a unique resource identification number (RID) in the AM. The first thing the AM does is validates whether the RID is indeed correct using a mediated query (MQ) to the firm that owns the RID. The firm either confirms that the RID is correct or it responds with a different, appropriate RID. Upon validating the proper RID, the AM then queries the firm as to whether the RID has been scheduled to complete the TID. If the firm response indicates that the TID has been scheduled, the AM then queries the firms as to whether the scheduled start date equals the desired start requested by the hub. If it does, the AM responds to the Hub with the currently planned start/finish pair and projected task. If the projected start date of the TID does not match the desired start requested by the firm, the firm returns the projected start/finish and cost, and waits for a response from the Hub as to whether the projected finish and cost are acceptable, or whether further analysis and/or options are required.

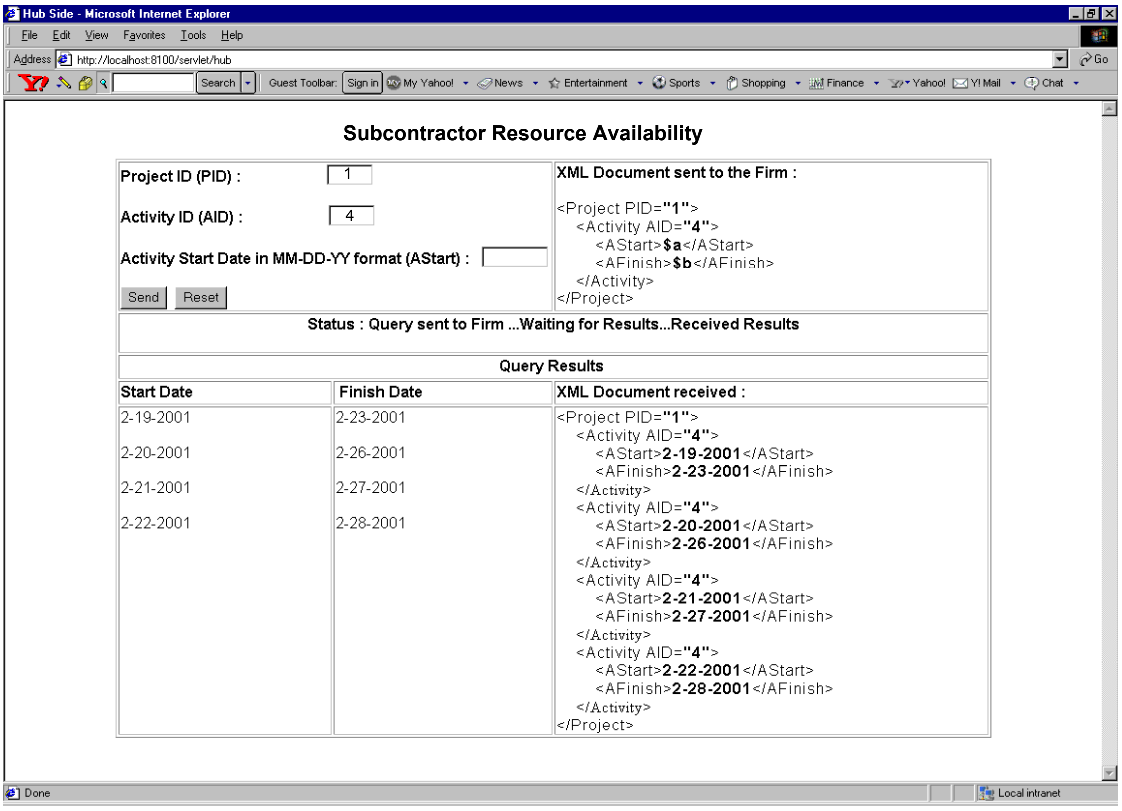
If the TID has not yet been scheduled for the RID, the AM then sends a mediated query to the firm of the form

(Request: Start, Finish, Cost | RID, TID)

As before, Stat and Finish are the expected output parameters, RID and TID are the input parameters. The end result of this query is a response to the Hub with an array of potential start/finish and cost combinations for the

TID. The way in which this end result is accomplished depends on the level of sophistication of the firm's legacy systems. If the legacy system has a sufficient degree of sophistication, the characterization of possible start/finish and cost combinations is done entirely within the firm's systems and is simply returned to the Hub through the AM. In many cases, however, the legacy system will not have the decision support capabilities required to provide such analyses, and so the AM will have the capability to perform this analysis on behalf of the firm. In such cases, the AM task scheduling sub-module executes the creation of schedule options for the TID.

The AM task scheduling sub-module will have the capability to extract information regarding the availability of the relevant RID over time, or equivalently, the firm's future working schedule for the RID. Associated with each TID, in addition to the RID, is (possibly) a required material (or material group) and the number of units required of the material (we use the term "material" loosely. This material may not in all cases consist of a physical material in the traditional sense; rather it may imply some required activity such as physical labor). In order for the AM to determine the time required by the RID to complete the TID, the AM first queries the firm to determine the output rate associated with the RID and material ID combination. The AM then computes the RID's resource requirement for the TID by dividing the number of units of material output required by the output rate. Once the AM has determined both the firm's planned work schedule for the RID and the resource requirements for performing the TID, the AM then invokes a scheduling algorithm based on predetermined policy specifications (a policy specification may be as simple as "schedule earliest due date first," for example) to determine a feasible start/finish cost combination for the TID. The scheduling algorithm can be re-run several times with different policy rules or parameters, in order to provide an array of potential start/finish and cost combination options from which the Hub may select. Before returning the array of start/finish and cost combinations to the Hub, the AM communicates this information to the firm for validation, if so desired by the firm. The firm is then free to return the proposed array back to the Hub (through the AM) or to modify the recommendations before returning them to the Hub.


Hub Side - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8100/servlet/hub
Go
Search
Guest Toolbar: Sign in
My Yahoo! News Entertainment Sports Shopping Finance Yahoo! Y! Mail Chat
Send Reset
Done
Local intranet

Subcontractor Resource Availability

Project ID (PID) : 1

Activity ID (AID) : 4

Activity Start Date in MM-DD-YY format (AStart) :

XML Document sent to the Firm :

```
<Project PID="1">
   <Activity AID="4">
      <AStart>$a</AStart>
      <AFinish>$b</AFinish>
   </Activity>
</Project>
```

Status : Query sent to Firm ...Waiting for Results...Received Results

Query Results

Start Date	Finish Date
2-19-2001	2-23-2001
2-20-2001	2-26-2001
2-21-2001	2-27-2001
2-22-2001	2-28-2001

XML Document received :

```
<Project PID="1">
   <Activity AID="4">
      <AStart>2-19-2001</AStart>
      <AFinish>2-23-2001</AFinish>
   </Activity>
   <Activity AID="4">
      <AStart>2-20-2001</AStart>
      <AFinish>2-26-2001</AFinish>
   </Activity>
   <Activity AID="4">
      <AStart>2-21-2001</AStart>
      <AFinish>2-27-2001</AFinish>
   </Activity>
   <Activity AID="4">
      <AStart>2-22-2001</AStart>
      <AFinish>2-28-2001</AFinish>
   </Activity>
</Project>
```

FIG.6: Snapshot of our SEEK prototype, illustrating a query about resource availability. Query results reflect data extracted and composed following analysis in Analysis Module.

Once a menu of possible start/finish and cost combinations is returned to the Hub for each TID associated with a PID, the Hub then invokes a project scheduling algorithm to determine which start/finish pairs it will select for each TID. This information is then finally transmitted back to the AM, which returns the final schedule information back to the firms. It is, of course, possible that the collection of options returned to the Hub for all TIDs is incompatible and that no feasible schedule exists that satisfies project budget and/or precedence constraints. Such occurrences will necessitate a schedule reconciliation procedure at the Hub, in which the Hub must first determine which TIDs led to the constraint violations, and must then initiate a set of queries to the affected firms requesting additional schedule options (the Hub may at the same time "suggest" or request certain time windows for performing the affected TIDs, which are compatible with the other project tasks). Figure 6 shows a snapshot of a sample hub-AM interaction using the SEEK prototype. In this particular interaction, the hub is requesting information about the availability of a resource belonging to a particular project (PID) and Activity (AID). For debugging purposes, the hub query and AM result are also shown in their native XML representation.

5.2 Data Reverse Engineering in the Context of a Project Management Example

This section describes the DRE activities that occur inside the KEM in order to configure the SEEK wrapper and Analysis Module to support the Hub query and data processing activities described immediately above.

Continuing the preceding example, the AM depends on KEM for the following two important items: (1) information about which data are available in the legacy systems in the firm - this will determine the algorithm used by the AM to satisfy a request from the Hub; and (2) a configured wrapper that can understand the mediated queries (MQ) and convert them into requests against the underlying legacy source, for further processing. The mediated queries are expressed in terms of the conceptual AM information model shown in Figure 7, and must be converted into equivalent queries represented by the information model of the legacy system. When the result is produced in the legacy source, it must be converted back into a representation that is consistent with the information model used by the AM. For the example above, a sample information model used by the AM is shown in Figure 7.

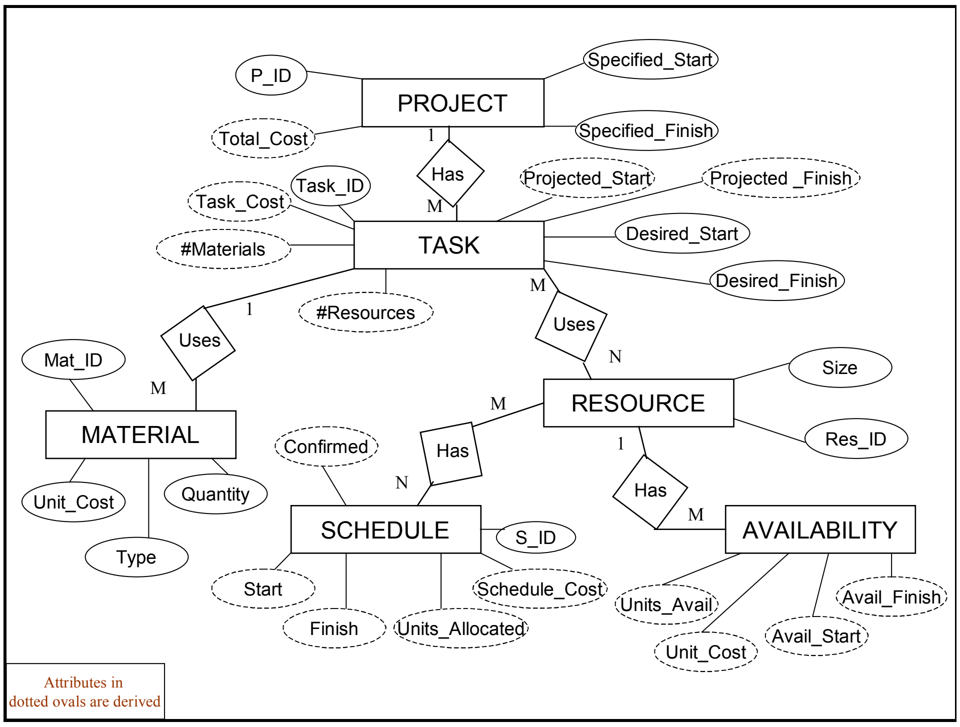


FIG.7: Information Model for AM for the example in Sec. 5a.

Figure 7 contains the entity-relationship (E/R) diagram representation of the Analysis Module information model. The purpose of this model is to provide a rigorous formalism for describing all interactions between objects in the project management schema (S_A), because there is a fixed reference for extracting data from legacy sources. The entities in Figure 7 are Project, Task, Material, Schedule, Resource, and Availability. Attributes of entities are specified within ovals whose outline is solid (dotted) if the attribute is time-invariant (resp. is derived or varies during a given project timeframe). Diamonds denote relations whose arity (e.g., 1-to-N, M-to-N) is specified on the links that the relation uses to connect two entities. For example, Resource is linked to Availability by the relation Has, whose arity is 1-to-M. This means that a given resource can have one or more availability intervals with associated Units Available and Unit Cost of the resource. A construction Material (e.g., brick or tile) is assumed to be constantly available, and has cost that does not vary within the scope of a project. In contrast, a Resource denotes human effort and thus has a Schedule to satisfy Task (and, therefore, Project) requirements, as well as a list of Availability intervals with associated cost. In this initial model, simplifying assumptions are made about Materials in order to constrain the analysis complexity. When modelling a situation where material availability is restricted and cost varies, the Resource entity can be used to describe a given Material. Thus, Figure 7 depicts an information model that accounts for all practical eventualities in construction project management.

To satisfy both requirements, the KEM employs a DRE methodology to extract schema information about the legacy source. The resulting schema as well as any available application code is processed by a semantic analyser to elicit the necessary semantics to be able to relate the source schema to the information model used by AM. In order to gain access to the firm and perform DRE, the KEM needs a simple wrapper to jumpstart the extraction process. A generic wrapper that can access the source (e.g., a wrapper using the JDBC call-level interface to connect to a MS Project application) is assumed. The generic wrapper is later configured by the KEM to be used by the AM at run-time.

5.2.1 Data Reverse Engineering in SEEK

This section describes how DRE is conducted in the context of the example in section 5.1, and continues with an outline of the semantic analysis procedure. The first step of DRE is the extraction of schema information from the legacy source, which in this example is a Microsoft Project application. The outcome of this step is a syntactic description of the relational database schema used by the Microsoft Project application for persistent storage of the project information. The full schema for the MS Project Database is accessible at: http://www.microsoft.com/office/project/prk/2000/Download/VisioHTM/P9_dbd_frame.htm

The SEEK approach to schema inferencing is based on the algorithms described by Chiang (Chiang 1995; Chiang et al. 1994) and Petit (Petit et al. 1996). The implementation of Chiang's algorithm is relatively straightforward, despite its tendency to redefine standard terms in nonstandard ways. Petit's algorithm is less completely reported than Chiang's, and uses Markowitz and Makowsky's (1990) method for extracting an extended entity-relationship (EER) diagram, but was found to be useful as a check on Chiang's algorithm. For purposes of brevity, only the schema extraction process is exemplified.

Our schema inferencing algorithm was applied to a Microsoft Project schema, as shown in the following stepwise description. Three assumptions are required, namely:

- The relational schema is in 3NF;
- Key attributes are consistently named; and
- One must be able to run queries on the legacy database. Hence, the database must be available and capable of processing queries. This implies knowledge of Q_L in Figure 3.

Step 1: Extracting Schema Information from the Legacy Source

The data dictionary, which is stored in the underlying database as a relational table, is queried to obtain *relation* and *attribute names* used by the application. DRE then finds all attributes that have non-null constraints and whose values are unique, which are called *candidate keys*. The *primary keys* are discovered via *user input*, as follows. If there is only one candidate key per entity, then that candidate key is the primary key. Otherwise, all candidate keys are presented to the user, who can (a) indicate the primary keys, or (b) include additional keys.

As a result, the following relations were obtained from the MS-Project schema:

MSP-Project [<u>PROJ_ID</u>, ...]

MSP-Availability [<u>PROJ_ID, AVAIL_UID</u>, ...]

MSP-Resources [PROJ_ID, RES_UID, ...]

MSP-Tasks [PROJ_ID, TASK_UID, ...]

MSP-Assignment [PROJ_ID, ASSN_UID, ...]

Step 2: Classification of the Relations

In this step, the primary key (PK) of each of the relations obtained in Step 1 is examined and compared with the primary keys (PK) of other relations to identify strong and weak entities, as well as specific and regular relations. According to Chiang's definition, a strong entity's PK does not properly contain a key of any other relation, whereas weak entities and specific relationships are defined by the following multiple conditions.

A Weak Entity (WE) relation ρ has multiple attributes in its PK and satisfies the following three conditions:

- A proper subset of ρ's PK contains keys of other strong or weak entities;
- The remaining attributes of ρ's PK do not contain a key of any other relation; and
- ρ has an identifying owner and properly contains the PK of its owner relation. *User input* is required to confirm these relationships.

If conditions 1) and 2) are satisfied, but condition 3) is not satisfied, then a Specific Relationship (SP) exists. A Regular Relation (RR) has a PK that is formed completely by concatenating the PKs of other entity relations. As a result, the following classification is obtained:

- Strong Entities: MSP_Project.
- Weak Entities: MSP-Resources, MSP-Tasks, and MSP_Availability.
- Specific Relationship: MSP-Assignment.

For efficiency, one can first use the PK to classify those strong and regular relations that have the PK of strong entities. Then, if the PK attributes appear as component(s) of the candidate keys for some other strong entities, this is a regular relationship.

Step 3: Classification of the Attributes.

Attributes are classified first if they are the attributes of the relation's primary key. Depending on the type of relation, the attribute classification can be further refined. We distinguish between the following attribute types:

Primary key attributes (PKA) are the attributes of strong entities' primary keys, or a weak entity relation's primary key that are also keys of some other strong or weak entity type and those attributes of relationship relation's primary keys that are also components of the key of other entity relations. *Dangling key attributes* (DKA) are attributes of the primary key of a weak entity relation that does not appear as a key of other relations. *Foreign key attributes* (FKA) occur when a non-primary key attribute (probably composite) appears as the key of another (strong or weak) relation. *Non-key attributes* (NKA) are those attributes that cannot be classified as PKA, DKA, or FKA. We obtain the following attribute classification for our example, summarized in Table 1.

Table 1: Attribute classification.

	PKA	DKA	GKA	FKA	NKA
MS-Project	Proj_ID				All the remaining attributes
MS-Resources	Proj_ID	Res_uid			
MS-Tasks	Proj_ID	Task_uid			
MS- Availability	Proj_ID	Avail_uid		Res_uid+Proj_ID	
MS-Assignment	Proj_ID		Assn_uid	Res_uid+Proj_ID, Task_uid+Proj_ID	

Step 4: Inclusion Dependencies (Referential Integrity Constraints)

If A and B are two relations and if X and Y are attributes or a set of attributes of A and B respectively, then the number of attributes in X should be equal to the number of attributes in Y. Here, the notation A.X << B.Y denotes that a set of values appearing in A.X is a subset of B.Y.

Inclusion dependencies are discovered by proposing all possible inclusion dependencies, assuming that A and B are two strong entity relations:

- If A and B have the same key X, then A.X << B.X or B.X << A.X is possible. This is used to identify is-a relationships and generalization hierarchies.
- If the key X of a strong or weak entity appears as a foreign key x of another relation B, there may be an inclusion dependency between B.x and A.X (i.e., B.x << A.X). This is used to identify binary relations represented by foreign keys.
- If the primary key attributes (X) of a relationship relation or a weak entity (S), appear as a key (X) of a strong/weak entity (A), then S.X << A.X may hold. This determines the owner of entities of weak entities, and determines the participating entity types for relationship types corresponding to relationship relations.

Also, if a set of non-key attributes appears in two distinct relations, then there may be an inclusion dependency representing a binary relationship. User input is required to resolve this.

After inclusion dependencies are discovered, invalid inclusion dependencies must be rejected. For each proposed inclusion dependency, one formulates two SQL queries that are issued on the target DBMS. Query results help reject invalid inclusion dependencies. Foreign key attributes that do not appear in any valid inclusion dependencies are classified as NKA. Projection and transitivity are used to remove redundant inclusion dependencies. This step obviates mistaken identification of participating entity types and identification of redundant IS-A relationships. For example, a << b and b << c implies that a << c.

The inclusion dependencies discovered in the example are listed as follows:

MSP_Assignment [Task_uid, Proj_ID] << MSP_Tasks [Task_uid, Proj_ID]

MSP_Assignment [Res_uid, Proj_ID] << MSP_Resources [Res_uid, Proj_ID]

MSP_Availability [Res_uid, Proj_ID] << MSP_Resources [Res_uid, Proj_ID]

MSP_Resources [Proj_ID] << MSP_Project [Proj_ID]

MSP_Tasks [Proj_ID] << MSP_Project [Proj_ID]

MSP_Assignment [Proj_ID] << MSP_Project [Proj_ID]

MSP_Availability [Proj_ID] << MSP_Project [Proj_ID]

The latter two inclusion dependencies are removed on the basis of transitivity.

Step 5: Identify Entity Types

Strong and weak entities, together with their owners, are identified by the following rules, given that strong (weak) entity relations associate strong (weak) entities:

- For each weak entity relation, a weak entity type is identified with the Dangling Key Attribute (DKA) as the key attribute.
- The owner of a weak entity type is determined by referring to inclusion dependencies.
- A weak entity and its identifying owner are related through an identifying relationship and always labelled dependent.

As a result, the Entity types in the ongoing example were identified as follows:

Strong entities: MSP_Project with Proj_ID as its key.

Weak entities: MSP_Tasks with Task_uid as its key and MSP_Project as its owner.

MSP_Resources with Res_uid as its key and MSP_Project as its owner

MSP_Availability with Avail_uid as key and MSP_Resources as owner.

Step 6: Identify Relationship Types

Default cardinality ratios are assigned as follows:

- One-to-one (1:1) for a IS-A and Inclusion relationships
- One-to-many (1:N) for relationships identified through FKs, and inclusion dependencies between NKAs.
- One-to-many (1:N) binary relationships between weak entities and their owner.
- Many-to-many (N:M) for relations that were specific or regular relationships.

The following relationship types were identified in the example:

- 1:N relationships for weak entity types: Between MSP_Project and MSP_Tasks, between MSP_Project and MSP_Resources, and between MSP_Resources and MSP_Availabilty
- Relationships represented by a Specific relationship relation: Since two inclusion dependencies involving MSP_Assignment exist (i.e., between Task and Assignment and between Resource and Assignment), there is no need to define a new entity. Thus, MSP_Assignment becomes a N:M relationship between MSP_Tasks and MSP_Resources.

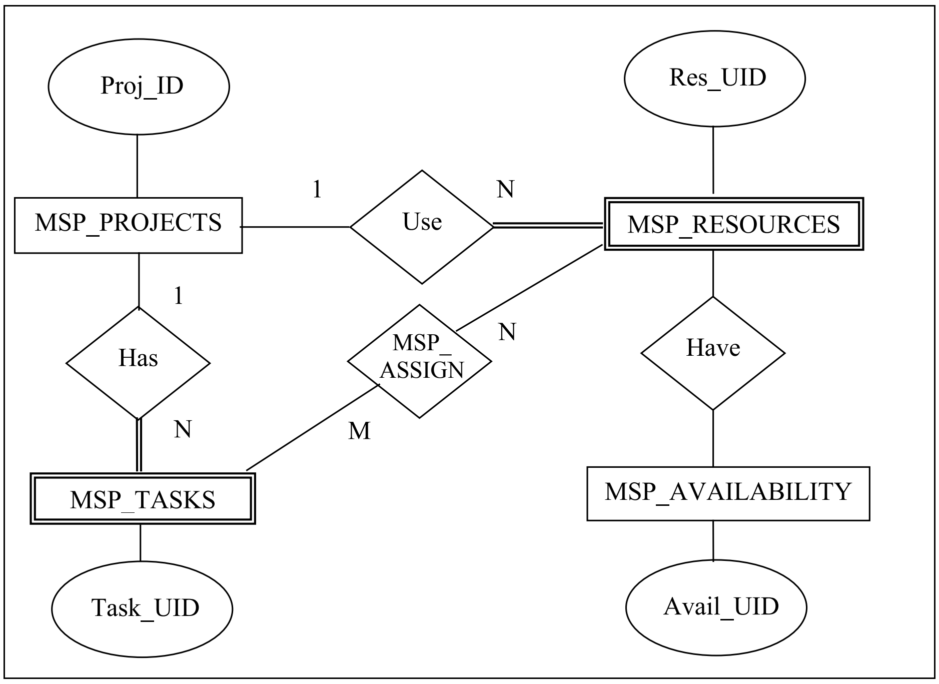


FIG.8: E/R diagram obtained by applying our DRE algorithm to a Microsoft Project implementation.

Step 7: Assign NKA

Non-key attributes are assigned on the basis of (a) relation type, (b) whether or not the relation contains FK's, and (c) the meaning of NKA(s) in the relations. All NKAs except those containing FK are assigned automatically. In the case of relations that contain FKs, *user input* is required to confirm the assignment.

As a result, all the Non Key Attributes can be attached to particular entities. NKAs are not shown here for the sake of legibility and simplicity.

Step 8: Generate the E/R Diagram

The resulting E/R diagram is shown in Figure 8. This diagram is isomorphic to the Analysis Module information model of Figure 7. Thus, the schema extraction algorithm exemplified herein produces the required schema for the legacy source.

5.2.2 Semantic Analysis

The goal of semantic analysis is to identify existing correspondences between the legacy schema obtained by DRE and the Analysis Module's information model and identify the meaning of the entities in the extracted schema. This process is discussed in detail below and is diagrammed schematically in Figure 9.

The SEEK approach to DRE represents significant improvement on prior methods due to the use of semantic analysis in conjunction with schema extraction. In the past, semantic analysis has been performed manually, with high cost in relationship to the resultant benefit. This traditional method is unacceptable in SEEK, due to (a) the need for efficiency given the constantly changing nature of planned SEEK applications, (b) the tendency of legacy systems to be modified unpredictably, and (c) the expected approach of iteratively discovering new material in legacy sources. Thus, in the high-level view of SEEK's complete DRE process shown in Figure 9, there are three primary sub-processes – DRE Schema Extraction, Semantic Analyzer, and User Input. DRE is the data reverse engineering process whose underlying algorithm was discussed and exemplified in the preceding section. Semantic Analysis is a procedure that takes as input the Analysis Module schema S_A and Legacy Source schema S_L, then examines each object in S_L in terms of each object in S_A to determine correspondences and meaning of each correspondence and extract business knowledge from the legacy system. Prohibitive complexity of correspondence matching is avoided at the outset by exploiting naming clues (e.g., *Resource* in SA corresponds to *MSP_Resource* in MS-Project).

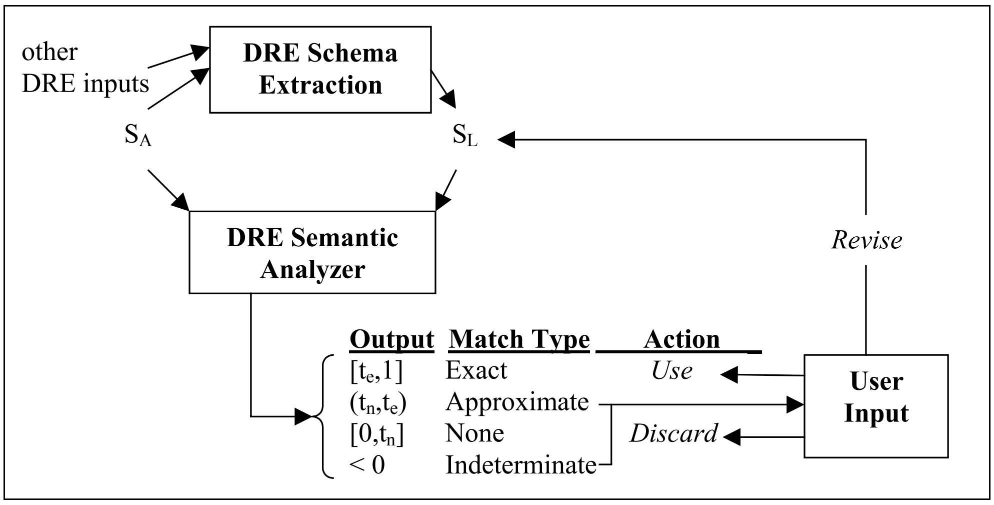


*FIG. 9: Notional diagram of semantic matching between analysis model schema S_A and legacy system schema S_L using a correspondence engine and three resultant actions (*Use*,* Discard*, or* Revise*) keyed to correspondence engine output.*

In order to determine correspondences between elements in S_A and S_L, a matching engine used to produce an interval representation of the match (e.g., a score ranging from zero to one). By thresholding the score, regions of *exact match*, *approximate match*, *indeterminate match*, and *no match* can be segmented. The corresponding thresholds in Figure 9 for exact match and no match are t_e and t_n, respectively. An indeterminate match results from the correspondence engine not being able to determine enough of the object parameters to compute a matching score. The matching engine uses a hierarchical clustering model for clustering objects in the multi-dimensional metric space. Equivalence of data objects within the individual clusters is determined using a number of distance functions that calculate the semantic distances among the objects based on their attribute values. The matching engine has been developed and tested in a separated Integration Project called IWIZ (Hammer and Pluempitiwiriyawej 2001; Pluempitiwiriyawej and Hammer 2001) and is being configured for use in SEEK.

In Figure 9, User Input is invoked for resolving indeterminate matches via one of three possible actions – *use* (exact match or good-enough approximate match clarified by user input), *discard* (no match or approximate match that is too ambiguous to be clarified by user input), and *revise* (approximate match that user doesn't know how to dispose of, but will resolve at a later time). This user input will be mediated via a simple GUI that will maintain an audit trail of correspondence resolution decisions. In future development, this audit trail will be used as input to a machine-learning algorithm that will help improve the accuracy of the correspondence resolution engine.

Additional semantic information about the extracted schema is obtained through code mining. The SEEK approach to mining semantic information from source code is based on the following premise: somewhere in the application code there are output statements that are related to report generation or display of query results. With the help of the output message string, which usually describes the variable being displayed, semantic information about that variable can be obtained. The task that remains is to trace this particular variable and find the statement in the program responsible for giving a value to this variable. Further, such statements could be part of extracting data sets from the result set of a query. Thus semantic information could be associated with a particular column of the database table being accessed in the query.

Program slicing (first developed by Horwitz and Reps (1992)) is used to prune the size of the code under consideration. In particular, by first identifying each output statement and the variable being displayed, backward slicing is used to produce only those program statements that affect the value of the output variable. Traversal of the program dependence graph then allows us to associate the output variable and its semantic meaning with the corresponding database query. This association between output variable and query allows us augment the semantic of the extracted source schema and to formulate a unique mapping between the source schema and the schema used by the analysis module. User input provides guidance and final validation for correctness.

Future extensions to SEEK DRE are to extract business rules. Sneed and Erdos (1996) define a business rule as an axiom by which a business decision is made or through which a business result is calculated. Since the outcome of a business decision usually affects one or more variables, a business rule can be thought of as a function that generates these values. One simple heuristic to identify all the business rules embedded in source code is to slice the code based on all input and output variables. Other possible heuristics for business rule extraction include inserting points in the code where the input data is delegated to different processing units and using the end point of a procedure (see, for example, Huang et al. (1996)). The problem of extracting business rules is challenging, but represents a significant extension to SEEK capabilities by allowing richer processing of the legacy information.

6. CONCLUSIONS

SEEK provides a structured approach using data reverse engineering (DRE) and wrapper development technologies to overcome the challenges of integrating (a narrow range of) information resident in heterogeneous legacy sources across the construction supply chain. Its modular architecture provides several important capabilities:

- Rapid configuration with limited set-up: SEEK can be rapidly configured to query a wide variety of legacy systems, removing the burden of set-up and integration that exists with current, manual techniques.
- Connection to physically and semantically heterogeneous sources: SEEK can automatically discover data and knowledge in a wide variety of legacy sources.
- Composition of knowledge: Via the Analysis Module, SEEK can compose answers to queries from the hub, extending the capabilities of the underlying legacy sources.
- Protection of source-specific, proprietary knowledge: SEEK establishes a layer between the source and end user, protecting details of source representation. Access is also set at the source level, further protecting privacy.

Developed under an on-going project, SEEK currently exists as a functional prototype with abilities to discover source schema in heterogeneous scheduling applications. It is being extended using sample data from a project on the University of Florida campus in cooperation with the construction manager Centex Rooney and several subcontractors and suppliers. The data testbed developed with these firms will allow rigorous testing of the capabilities of the SEEK toolkit.

SEEK provides important contributions to the theory of knowledge capture and integration. The difficulty of (semi-)automatically linking salient knowledge resident in legacy systems with decision-support systems is well known. The differences in schema and data representation both across legacy systems and between legacy systems and decision support tools have been difficult to resolve in theory given the complexity of the problem. Since data modelling is inherently subjective and guided by many different requirements, there are innumerable ways to represent the same data (Kent 1989). Thus knowledge capture and information integration methodologies have to-date relied heavily on human input. Current contributions to theory lie in our development of new algorithms for extraction of semantic knowledge and business rules from application code that go beyond current practices such as pattern matching and code slicing.

SEEK also represents a contribution to the practice of enterprise information integration. Prior to SEEK, integration in practice was achievable only through significant programmatic effort with little reusability of code. By providing an automated approach, SEEK represents an important step towards building scalable sharing architectures that can be configured with less human effort, fewer errors, and in shorter time than with current manual integration techniques. Further, by operating with heterogeneous legacy sources, SEEK provides a basis for rapid deployment of decision support tools for use by business networks of varying size, composition, and sophistication.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under grant numbers CMS-0075407 and CMS-0122193. The authors also thank graduate students Sangeetha Shekar, Nikhil Haldavnekar, Huanqing Lu, and Bibo Yang for their work on this project and contributions to the prototype.

8. REFERENCES

aecXML. (1999). "A framework for electronic communications for the AEC industries." IAI aecXML Domain Committee, http://www.aecxml.org/docs/aecwhite.doc, 9 pages.

Aiken, P. (1996). *Data Reverse Engineering: Slaying the Legacy Dragon*, McGraw-Hill.

Aiken, P., Muntz, A., and Richards, R. (1994). "DoD legacy systems: Reverse engineering data requirements." *Communications of the ACM*, 37(5), 26-41.

Amor, R., and Faraj, I. (2001). "Misconceptions about integrated project databases." *ITCON - Electronic Journal of Information Technology in Construction*, 6, 57-66.

Ashish, N., and Knoblock, C. (1997). "Wrapper generation for semi-structured internet sources." *Workshop on Management of Semistructured Data*, Ventana Canyon Resort, Tucson, Arizona, May 16, 1997.

Ballard, G., and Howell, G. (1998). "Shielding production: essential step in production control." *ASCE Journal of Construction Engineering and Management*, 124(1), 11-17.

Barrie, D. S., and Paulson, B. C. (1992). *Professional Construction Management*, McGraw-Hill, New York.

Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, G., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., and Woelk, D. (1996). "Semantic integration of information in open and dynamic environments." *MCC-INSL-088-96*, MCC.

Bell, L. C., and Stukhart, G. (1987). "Costs and benefits of materials management systems." *ASCE Journal of Construction Engineering and Management*, 113(2), 222-234.

Boulanger, D., and March, S. T. (1989). "An approach to analyzing the information content of existing databases." *Database*, 20(2), 1-8.

Brandon, P., Betts, M., and Wamelink, H. (1998). "Information technology to support construction design and production." Computers in Industry, 35, 1-12.

Califf, M. E., and Mooney, R. J. (1998). "Relational learning of pattern-match rules for information extraction." Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing, Stanford, CA, March 1998, AAAI Press, 6-8.

Casanova, M. A., and Sa, J. E. A. d. (1983). "Designing entity-relationship schemas for conventional information systems." Third International Conference on Entity-Relationship Approach, August 1983.

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. (1994). "The TSIMMIS project: Integration of heterogeneous information sources." Tenth Anniversary Meeting of the Information Processing Society of Japan, Tokyo, Japan,October 1994, 7-18.

Chiang, R. H. (1995). "A knowledge-based system for performing reverse engineering of relational database." Decision Support Systems, 13, 295-312.

Chiang, R. H. L., Barron, T. M., and Storey, V. C. (1994). "Reverse engineering of relational databases: Extraction of an EER model from a relational database." Data and Knowledge Engineering, 12(1), 107-142.

Chikofsky, E. a. J. C. (1990). "Reverse engineering and design recovery: A taxonomy." IEEE Software, 7(1), 13-17.

Codd, E. F. (1970). "A relational model for large shared data banks." Communications of the ACM, 13(6), 377-387.

Crowley, A. (1999). "The development of data exchange standards: the legacy of CIMsteel." The CIMsteel Collaborators, http://www.cis2.org/faq/Crowley1999.pdf, 6 pages.

Davis, K. H. (1995). "August-II: A tool for step-by-step data model reverse engineering." IEEE Second Working Conference on Reverse Engineering, May 1995, 146-155.

Davis, K. H., and Aiken, P. (2000). "Data reverse engineering: A historical survey." IEEE Seventh Working Conference on Reverse Engineering, November 2000, 70-78.

Davis, K. H., and Arora, A. K. (1985). "Methodology for translating a conventional file system into an entity-relationship model." Fourth International Conference on Entity-Relationship Approach, November 1985, 148-159.

Davis, K. H., and Arora, A. K. (1987). "Converting a relational database model into an entity-relationship model." Sixth International Conference on Entity-Relationship Approach, October 1987, 271-285.

Dayani-Fard, H., and Jurisica, I. (1998). "Reverse engineering: A history - where we've been and what we've done." IEEE Fifth Working Conference on Reverse Engineering, April 1998, 174-182.

Dumpala, S. R., and Arora, S. K. (1981). "Schema translation using the entity-relationship approach." Second International Conference on the Entity-Relationship Approach, August 1981, 337-356.

Economist. (2001). "List makers take control." The Economist, 360(8240), September 22, 2001, S27-29.

Englebert, V., and Hainaut, J.-L. (1999). "DB-MAIN: A next generation Meta-CASE." Information Systems Journal, 24(2), 99-112.

Froese, T., Fischer, M., Grobler, F., Ritzenthaler, J., Yu, K., Sutherland, S., Staub, S., Akinci, B., Akbas, R., Koo, B., Barron, A., and Kunz, J. (1999). "Industry foundation classes for project management - a trial implementation." ITCON - Electronic Journal of Information Technology in Construction, 4, 17-36.

Gruser, J.-R., Raschid, L., Vidal, M. E., and Bright, L. (1998). "Wrapper generation for web accessible data sources." 3rd IFCIS International Conference on Cooperative Information Systems, New York City, New York, USA, August 20-22, 1998, 14-23.

Hainaut, J.-L. (1991). "Database reverse engineering: Models, techniques, and strategies." 10th International Conference on Entity-Relationship Approach,November 1991, 729-741.

Halpin, D. W. (1993). "Process-based research to meet the international challenge." ASCE Journal of Construction Engineering and Management, 119(3), 417-425.

Hammer, J., Breunig, M., Garcia-Molina, H., Nestorov, S., Vassalos, V., and Yerneni, R. (1997a). "Template-based wrappers in the TSIMMIS system." Twenty-Third ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 23-25, 1997, 532.

Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., and Crespo, A. (1997b). "Extracting Semistructured Information from the web." Workshop on Management of Semistructured Data, Tucson, Arizona, May 17, 1997, 18-25.

Hammer, J., Garcia-Molina, H., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. (1995). "Integrating and accessing heterogeneous information sources in TSIMMIS." *AAAI Symposium on Information Gathering*, March 1995, 61-64.

Hammer, J., McHugh, J., and Garcia-Molina, H. (1997c). "Semistructured data: The TSIMMIS experience." *First East-European Symposium on Advances in Databases and Information Systems (ADBIS '97)*, St. Petersburg, Russia, September 1997, 1-8.

Hammer, J., and Pluempitiwiriyawej, C. (2001). "Overview of the integration wizard project for querying and managing semistructured data in heterogeneous sources." *The Fifth National Computer Science and Engineering Conference (NSEC 2001)*, Chiang Mai University, Chiang Mai, Thailand, 7 - 9 November 2001.

Hensley, J., and Davis, K. H. (2000). "Gaining domain knowledge while data reverse engineering: An experience report." *Data Reverse Engineering Workshop, EuroRef Seventh Reengineering Forum*, February 29-March 3, 2000.

Horwitz, S., and Reps, T. (1992). "The use of program dependence graphs in software engineering." *Fourteenth International Conference on Software Engineering*, Melbourne, Australia, May 1992.

Howell, G., and Ballard, G. (1997). "Factors affecting project success in the piping function." *Lean Construction*, L. Alarcón, ed., A. A. Balkema, Rotterdam, The Netherlands, 161-185.

Huang, H., Tsai, W. T., Bhattacharya, S., Chen, X. P., Wang, Y., and Sun, J. (1996). "Business rule extraction from legacy code." *20th International Computer Software and Applications Conference (COMPSAC '96)*, May 1996, 162 -167.

IAI. (1996). "End user guide to Industry Foundation Classes, enabling interoperability in the AEC/FM industry." International Alliance for Interoperability (IAI).

Johannesson, P., and Kalman, K. (1989). "A method for translating relational schemas into conceptual schemas." *Eigth International Conference on the Entity-Relationship Approach*, November 1989, 271-285.

Kent, W. (1989). "The many forms of a single fact." *Proceedings of the IEEE Spring Compcon*, San Francisco, CA, February, 1989, 26-36.

Klug, A. C. (1980). "Entity-relationship views over uninterpreted enterprise schemas." *First International Conference on the Entity-Relationship Approach*, August 1980, 39-60.

Kushmerick, N., Weld, D. S., and Doorenbos, R. B. (1997). "Wrapper induction for information extraction." *IJCAI*, 1, 729-737.

Markowitz, V. M., and Makowsky, J. A. (1990). "Identifying extended entity-relationship object structures in relational schemas." *IEEE Transactions on Software Engineering*, 16(8), 777-790.

Melkanoff, M. A., and Zaniolo, C. (1980). "Decomposition of relations and synthesis of entity-relationship diagrams." *First International Conference on the Entity-Relationship Approach*, August 1980, 277-294.

Moh, C.-H., E-P. Lim, and W-K. Ng. (2000). "Re-engineering structures from Web documents." *ACM International Conference on Digital Libraries 2000,* 67-76.

Mooney, R. J. (1999). "Learning for semantic interpretation: Scaling up without dumbing down." *Workshop on Learning Language in Logic*, Bled, Slovenia, June 1999, 7-14.

Navathe, S. B., and Awong, A. M. (1988). "Abstracting relational and hierarchical data with a semantic data model." *Entity-Relationship Approach,* 305-333.

Nilsson, E. G. (1985). "The translation of COBOL data structures to an entity-relationship type conceptual schema." *Fourth International Conference on the Entity-Relationship Approach*, November 1985, 170-177.

O'Brien, W., and Hammer, J. (2001). "Robust mediation of supply chain information." *ASCE Specialty Conference on Fully Integrated and Automated Project Processes (FIAPP) in Civil Engineering*, Blacksburg, VA, September 2001 (rescheduled to January 2002), 415-425.

O'Brien, W. J. (1997). "Construction supply-chains: case study, integrated cost and performance analysis." *Lean Construction*, L. Alarcón, ed., A. A. Balkema, Rotterdam, The Netherlands, 187-222.

O'Brien, W. J., Fischer, M. A., and Jucker, J. V. (1995). "An economic view of project coordination." *Construction Management and Economics*, 13(5), 393-400.

Papakonstantinou, Y., Gupta, A., Garcia-Molina, H., and Ullman, J. (1995). "A query translation scheme for rapid implementation of wrappers." *Fourth International Conference on Deductive and Object-Oriented Databases*, Singapore, December 1995.

Paul, S., and Prakash, A. (1994). "A framework for source code search using program patterns." *Software Engineering*, 20(6), 463-475.

Petit, J.-M., Toumani, F., Boulicaut, J.-F., and Kouloumdjian, J. (1996). "Towards the reverse engineering of denormalized relational databases." *Twelfth International Conference on Data Engineering (ICDE)*, New Orleans, LA, February 1996, 218-227.

Pluempitiwiriyawej, C., and Hammer, J. (2001). "A hierarchical clustering model to support automatic reconciliation of semistructured data." *TR01-015*, University of Florida, Gainesville, FL, 35 pages.

Rashid, L. (1997). "University of Maryland Wrapper Generation Project." http://www.umiacs.umd.edu/labs/CLIP/DARPA/ww97.html.

Sahuguet, A., and Azavant, F. (1998). "W4F: a WysiWyg web wrapper factory." University of Pennsylvania, Philadelphia, PA.

Shao, J., and Pound, C. (1999). "Reverse engineering business rules from legacy system." *BT Journal*, 17(4).

Shtub, A. (1988). "The integration of CPM and material management in project management." *Construction Management and Economics*, 6, 261-272.

Silva, A. M., and Melkanoff, A. (1979). "A method for helping discover the dependencies of a relation." *Conference on Advances in Database Theory*, Toulouse, France, December 1979, 115-133.

Sneed, H. M., and Erdos, K. (1996). "Extracting business rules from source code." *Fourth Workshop on Program Comprehension*,August 1996, 240 -247.

Song, I.-Y., and Froehlich, K. (1995). "Entity-relationship modeling." *IEEE Potentials*, 13(5), 29-34.

Staub-French, S., and Fischer, M. A. (2000). "Practical and research issues in using Industry Foundation Classes for construction cost estimating." *CIFE Working Paper No. 56*, Stanford University, Stanford, CA, 45 pages.

Tayur, S., Ganeshan, R., and Magazine, M. (eds.) (1999). *Quantitative Models for Supply Chain Management*, Kluwer Academic Publishers, Boston/Dordecht/London.

Teicholz, P., and Fischer, M. A. (1994). "A strategy for computer integrated construction technology." *ASCE Journal of Construction Engineering and Management*, 120(1), 117-131.

Thomas, D. J., and Griffin, P. M. (1996). "Coordinated supply chain management." *European Journal of Operations Research*, 94, 1-15.

Tommelein, I. D., and Ballard, G. (1997). "Coordinating specialists." *Technical Report 97-8*, Construction Engineering and Management Program, Civil and Environmental Engineering Department, University of California, Berkeley, Berkeley, CA,

Turk, Z. (2001). "Phenomenological foundations of conceptual product modelling in architecture, engineering, and construction." *Artificial Intelligence in Engineering*, 15(2), 83-92.

Vrijhoef, R., and Koskela, L. (1999). "Roles of supply chain management in construction." *Proceedings of IGLC-7*, Berkeley, CA, July 26-28, 1999, 133-146.

Wegelius-Lehtonen, T., and Pahkala, S. (1998). "Developing material delivery processes in cooperation: an application example of the construction industry." *International Journal of Production Economics*, 56-57, 689-698.

Wiederhold, G. (1998). "Weaving data into information." *Database Programming and Design*, 11(9).

Wiggerts, T., Bosma, H., and Fielt, E. (1997). "Scenarios for the identification of objects in legacy systems." *IEEE Fourth Working Conference on Reverse Engineering*, September 1997, 24-32.

Wills, L. M. (1994). "Using attributed flow graph parsing to recognize clichés in programs." *International Workshop on Graph Grammars and Their Application to Computer Science.*, November 1994, 101-106.

Zamanian, M. K., and Pittman, J. H. (1999). "A software industry perspective on AEC information models for distributed collaboration." *Automation in Construction*, 8, 237-248.