CIB W78 Workshop, Cairns, Australia, July 9-11, 1997.

Software Agents and Robots in Construction: An Outlook

v
Ziga Turk,
University of Ljubljana, Faculty of Civil Engineering, Slovenia
http://www.fagg.uni-lj.si/~zturk/

Summary: The rapid growth of the World Wide Web (Web) has spawned some rather new technologies for handling the vast amount of information available there. Services like Yahoo, AltaVista and Virtual Software Library use web crawling robots and agents to gather the information. Construction information processes have many similarities to the Internet - both are fragmented, there are numerous information sources and the information is poorly structured. We have successfully applied similar technologies that have proven themselves on the Internet to handle construction information. Custom robots and an agent-enabled database have been developed and have been used in the on-going research projects. They have proven useful on the global construction scope of a construction virtual library, on a narrower national scope of regulation management and on the project level to manage documents. The paper identifies the problems, introduces the technologies, demonstrates how they were used in construction context and concludes that in a client-server environment such as the Internet or an intranet, agent technology is an important building block of CIC environments.

Keywords: construction information technology, computerization of building regulations, document management, virtual libraries, Internet, intranet, agent, robot, database.

1. Introduction

We claim that information logistics is a vital component of CIC (computer integrated construction) and that technology is required to facilitate not just the just-in-time (search) but the just-in-case (learning) delivery of information as well.

1.1 Motivation

Modern construction1 business, as fragmented and specialized as we know it, was by large enabled by the 18th century invention of descriptive geometry. Before that time, the entire construction process was conducted by a single individual - the master builder. The advances of construction technologies were very slow if present at all. The most daring structures in the world, e.g. world largest domes, would be more than a thousand years old. Not even brilliant individuals were able to finish a construction project during their life time. Finally, the precise construction drawings, using a common set of technical symbols enabled information sharing between different professionals. This enabled the use of advanced building technologies but also fragmentation of the construction business.

Since 1960s, the computers were used in construction to create new information, first by structural analysis software and later by computer aided drafting tools, expert systems etc. The goal was to automate the base processes and reduce human interventions. This was leading to the "islands of automation" (Fenves, 1991) and

[1] In this paper, the term construction is used to denote many disciplines related to the built environment, including building, civil engineering, structural engineering …

ultimately to computer automated construction. Since the late 1980s the computer integrated construction (CIC) research has focused on the finding, moving and re-using the information. Right now, humans are all too often the glue between the applications (Fig. 1). Reducing human intervention from controlling the glue processes leads to computer integrated construction.

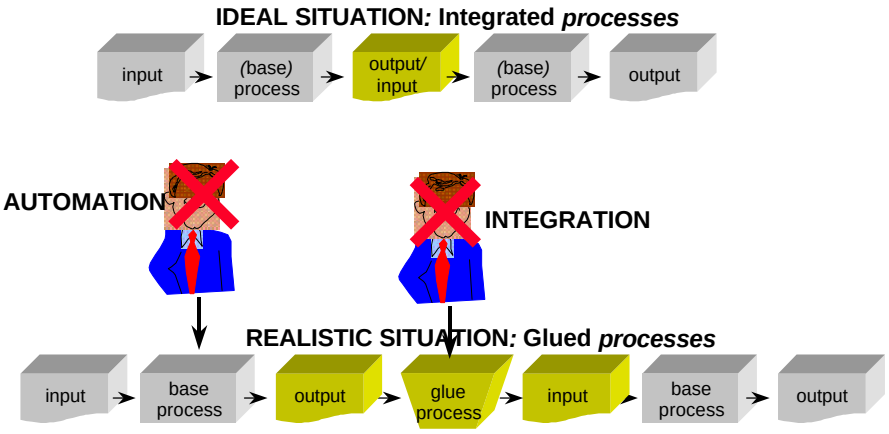

Fig. 1: Integrated and glued processes; integration and automation is achieved by removing human control either from base or from glue processes.

Several aspects of the glue process can be examined (Turk, 1997):

- Information scope. What kind of information is exchanged.
- Actors. Who is involved in information exchange.
- Time. When does the integration take place.
- Technology. What kind of technologies are used to facilitate the information flow.

Looking at the time aspect we notice that information creation and information use may be linked *sequentially* or *concurrently*. Concurrently, the "use" process guesses or approximates the outputs of the "create" process, verifies the correctness of the guess afterwards and attempt to resolve any conflicts. E.g. structural system and load are estimated during the design of the foundations, to speed up the beginning of the construction

Sequentially, the integration may happen (1) *just-in-time*, (2) *just-in-case* and (3) *once-in-time*. Looking up a design part the moment we need to know the detail is an example of just-in-time access. Going to school, reading a journal or visiting conferences is an example of just-in-case activity - we learn *in case* sometime the information might be useful. Watching TV or listening to the radio is an example of the once-in-time activity. If we missed it, it is gone forever.

IT has been applied all four time-related information distribution options as shown in table 1.

Most of the research so far has been directed towards the just-in-time information delivery which lets clients quickly find the information. Particularly when IT was not available, classification systems organized concepts beforehand to speed the search. Product models with extensive data structures use product's structure, layout, geometry, function etc. to explicitly link the related information. Full text indexing techniques are aimed at finding a needle in a haystack and can find information based on the frequency and proximity of the words in the document.

Table 1:

type	traditional	IT supported
concurrent	educated guess, experience, past cases	data warehouse, fast tracking, concurrent engineering, conflict management
just-in-time	book look-up, library look-up, phone call to expert ...	database lookup, Internet search, discovery and search agents ...
just-in-case	reading books, magazines, journals, school, visiting conferences ...	subscriptions to customized content, distant learning ...
once-in-time	watching TV, listening to radio ...	not-archived discussion systems, push and -cast services ...

1.2 Problem statement

It has been estimated that the amount of data available on line is doubling every 6 months. According to Moor's law, computational power of computers is doubling only every 18 months. There are several consequences of this widening gap which have become apparent with the increasing dependence on online information sources:

1. It is difficult to keep an overview of relevant construction information which is available on-line. This holds true both for general construction information, as well as for county or project specific information and may cause that we will not be able to find things just-in-time.
2. Using the just-in-time information delivery we must know what we are looking for. However, we do not learn new things. IT has not been doing much for the just-in-case delivery.
3. We should be trying to minimize the amount of information we are supposed to be dealing with and have machines assist more efficiently.

General search engines such as AltaVista or Lycos return a very high percentage of noise when searched for construction specific information. The other alternative are the registration based libraries, such as Yahoo, Virtual Library of Civil Engineering, etc. and hotlists (like VTT's) which lack completeness and convey a highly subjective view on the subject. Management of such lists is laborious and error prone. It was suggested before (Amor et al, 1996) that an answer to these problems are not sophisticated search engines, such as AltaVista, or universal cataloguing systems, such as Yahoo, but topical virtual libraries. In this paper we claim that agents and robots are enabling technology for such libraries. They do information mining, provide automatic updates and avoid subjective preferences of the librarian. Additionally, agent technology built right into the library, can be used to improve the way end users communicate with the library and learn about news in their field.

Internet is a very poor media for discovering important information by browsing and for the just-in-case delivery. The just-in-case is implemented in the form of mailing lists which hardly contain personalized information we are looking for when we search. Periodic manual searching is one of the ways to keep in touch with new items on line. But again, we could have someone do a just-in-time search for us, examine the results, and present only things which are new and which we would find useful.

Similarly to what such tools would do for information on the Internet, software could monitor the intranet for project specific information.

2. Technologies

We propose that agents, robots and web oriented databases are the key information technologies which can be applied to address the problems enumerated above.

2.1 Robots

Koster (1995) defines a web robot as a "program that traverses the Web's hypertext structure by retrieving a document, and recursively retrieving all documents that are referenced". These programs are sometimes called "spiders", "web wanderers", or "web worms", or shortly "bots". Robots are used by WWW search engines, such as Lycos, HotBot or AltaVista to locally store every web page for further processing which is normally limited to the generation of a full text and keyword indexes. These would then be stored in a database which could be searched by the users over the WWW. Koster estimates that more than 100 such robots are gathering data on the web and that a handful keeps visiting each site every week.

Since only a fraction of the Web is related to construction, searches using this technology only, return a very noisy results sets - each really relevant item found is accompanied by several which are not interesting at all.

We have been involved in Internet robot development since 1994 when we developed the Virtual Software Library. Its back-end robot was scouting the Internet for new public domain software (Turk, 1995). Still used by shareware.com, it manages one of the largest collections on the Internet.

2.2 Agents

AI (artificial intelligence) research seems to be shifting its attention from making computers intelligent towards researching methods which would prevent computers from generally being so very stupid (Riesbeck, 1996). The recent interest in (intelligent) agent technology is aimed at creating assistants which make sure computer programs do not act so very stupidly, that they learn from the users' requests and, in their own initiative, attempt to assist them.

According to Webster's dictionary (Webster, 1996), an agent is "One that acts for or as the representative of another." A software agent is a piece of software which acts to accomplish tasks on behalf of its user. Many agents are based on the idea that the user needs only to specify a high-level goal instead of issuing explicit instructions, leaving the `how' and `when' decisions to the agent. Etzioni and Weld (1995) defined a list of qualities of software agents:

- Autonomous: an agent is able to take initiative and exercise a non-trivial degree of control over its own actions
- Goal-oriented: an agent accepts high-level requests indicating what a human wants and is responsible for deciding how and where to satisfy the request.
- Collaborative: an agent does not blindly obey commands, but has the ability to modify requests, ask clarification questions, or even refuse to satisfy certain requests.
- Flexible: the agent's actions are not scripted; it is able to dynamically choose which actions to invoke, and in what sequence, in response to the state of its external environment.
- Self-starting: unlike standard programs which are directly invoked by the user, an agent can sense changes to its environment and decide when to act.
- Temporal continuity: agents are continuously running processes, not ``one-shot" computations that map a single input to a single output, then terminate.
- Character: an agent has a well-defined, believable "personality" and emotional state.
- Communicative: the agent is able to engage in complex communication with other agents, including people, in order to obtain information or enlist their help in accomplishing its goals.

- Adaptive: the agent automatically customizes itself to the preferences of its user based on previous experience. The agent also automatically adapts to changes in its environment.
- Mobile: an agent is able to transport itself from one machine to another and across different system architectures and platforms.

Software having at least some of these features has been around for years and knowing the internal workings of that software and of some we have created makes us reluctant to add the word "intelligent".

In addition to creating few custom agents and robots, we have also built agent functionality into our Web oriented database.

2.3 Web Oriented Databases

Crafting individual Web pages is tedious, slow, difficult to keep consistent and to maintain by people not fluent in hypertext markup language (HTML). Published information usually follows the same form which makes many Web publishing efforts in fact a database application. The so called “web database connectivity” has been expensive and quite complicated to use. It required commercial database systems such as Sybase, Oracle or Access, special servers and has not been very open towards the inclusion of multimedia data.

We define main features of WWW oriented database as:

- Web browser is the main interface for creating, using and managing the database. All basic database functions, such as adding, editing, deleting, querying, if information can be done using a web browser. Database administration and maintenance is also performed through a browser.
- Database engine does independent user authentication or uses server-side authentication. It is important to allow for anonymous users. Even if users are anonymous, they should be able to protect their data from being edited by someone else.
- Export and import of data to and from PC applications such as Excel or Access should be enabled.
- The database should have similar user interface and search syntax as other Web services in order to get quickly accepted by the visitors on the WWW.

Advanced features of a web oriented database include:

- Support for multimedia information, such as pictures, drawings, printable documents and other binary data.
- Include software agent technology which can learn the needs of the user.
- Tie in with other Internet technologies, such as email, chat and on-line conferencing.
- Enable rapid development of database application and include report and form generation.
- Support internationalization, national character sets, collate sequences and multi-lingual user interface.
- Support SQL and ODBC data storage.

Since 1995 we have been developing a “web oriented database system” called WODA (Turk, 1995-97). WODA is a CGI application written in Perl. Its design goal was to create a smart and simple tool which would allow very rapid creation of small to medium size database applications which could be used and managed using Web tools. WODA is tightly integrated with Web technology, supports multimedia contents (such as full text articles) file uploads, full text searches and includes software agent technology. WODA can talk to the user in English, German, French, Spanish and Slovene. It has been used locally for the Faculty information system, nationally to support Building Center of Slovenia and in several international cooperation projects such as Esprit-SCENIC, Copernicus ATEM, EASY database and in the support of last year's W78 workshop.

Using WODA we have learned that web orientation and rapid prototyping tools can outweigh database features of commercial systems.

WODA dynamically generates all Web pages based on the information schema, responds to end user queries, allows adding new library items and takes care of the search-agent requests.

3. Case studies

The case studies where the above technology has been applied are presented. The scope of the information ranges from construction subset of the Internet down to project specific information.

3.1 Construction Virtual Library

A library is an ordered and managed collection of information sources, such as books, journals, CD-ROMs, and other kind of media. Minimal ordering is ensured by library classification systems. Minimal managing ensures that only selected material is available in a library and that items in the library can be found and used. Since the invention of writing, libraries have been the main warehouses of knowledge and information.

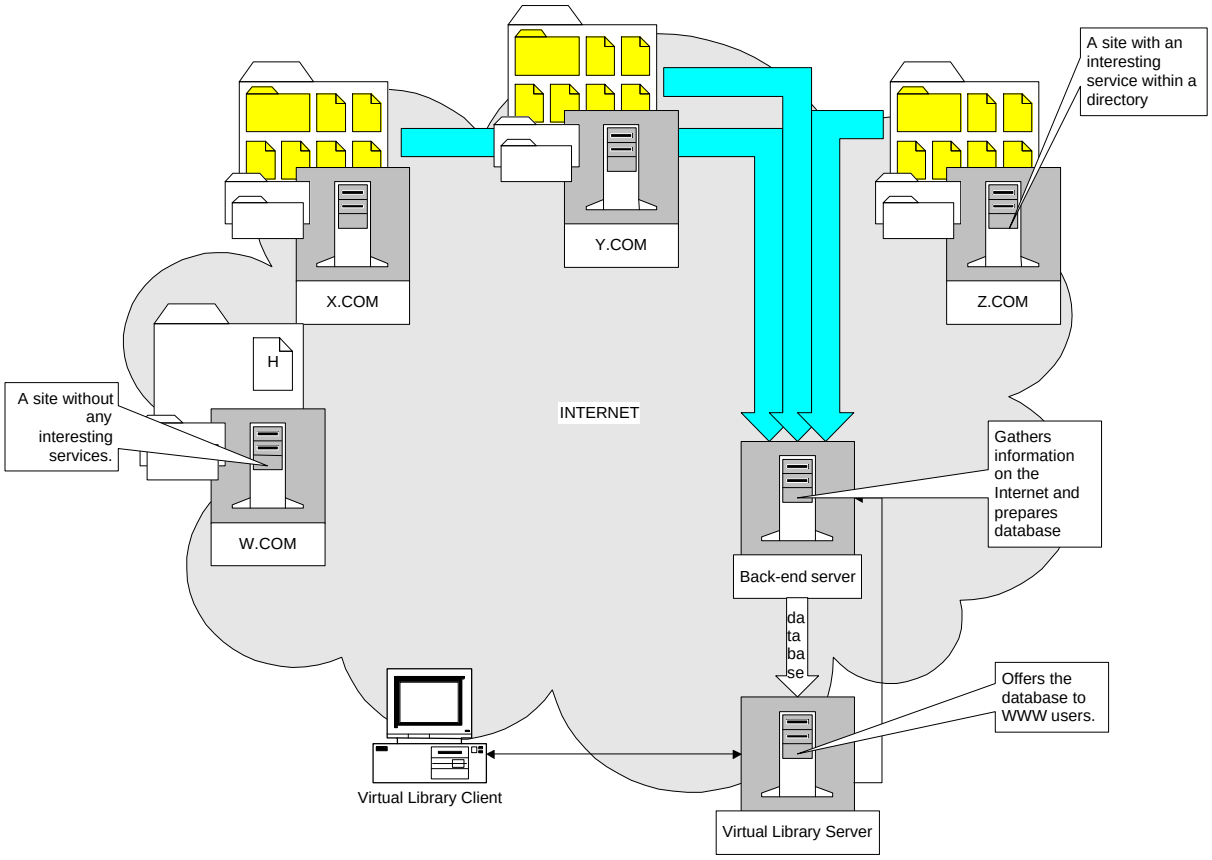

Fig. 2: Overall architecture of a virtual library.

Electronic media in combination with the computer networks separate the media (paper, magnetic tape, celluloid) from the content (a novel, a symphony, a movie). With the content available on-line, libraries need not store the content - just point to it. This makes them virtual - they are like a library with everything, but the books. They do have the front desk, the catalogue sorted in few ways, a librarian who would keep the catalogue up to date ... But virtual library is only a collection of *pointers* to information sources. In contrast to a traditional library, a virtual library stores only the meta information which includes the source of the information, summary, keywords, classifications and some measure of quality. A library makes sure that its lists are up-to-date and the items it lists are available.

Virtual libraries, hotlists and bookmarks are all examples of resource directories. Creating such resource directories for construction has been quite popular and in the early days the resource directory services outnumbered any other construction service. Examples of construction resource directories are the Civil Engineering Virtual Library, resource list at Curtin University, Lund University, Construction Technology Centre Atlantic, British construction directory at BRE, NICE database in Ljubljana.

On the Internet, there are thousands of information servers. In Fig. 2, they are shown as w.com, x.com, y.com and z.com. In reality these would be something like www.itcon.org, www.asce.org or www.stanford.edu. Servers include several directories depicted as folders. Information in some of these folders may be interesting for inclusion in a virtual library. Such folders are only on servers x, y and z. Such a folder with all its sub folders provides a service on the Internet, for example all information about a research project, a conference or some engineering standard ...

Data in these directories is then examined by the *back-end server*. The back-end server processes the information, extracts keywords, classifies the site etc. and distills the data into a format which can be used by the *front-end* (WWW) server. Based on the information collected, the *front-end server* is generating library's home page, tables of contents and can be queried by the end users and also used by the virtual library staff to manage the library.

The following databases are created by the back-end:

- *Index of hotlists*. This index includes pointers to Web pages which compile lists of pointers, where new services which should be included in this library, are likely to appear. Examples of such pages are Yahoo's Civil Engineering page, VTT's CIC directory, directory of Esprit projects at the Cordis server, etc.
- *Full text index*. This index compiles full text of all documents within the interesting directories. The index is stored in a compressed format.
- *Virtual Library Directory*. This directory stores the meta information about every service found.

Data in these databases is maintained by several software modules two of which (information miner and librarian) have many features of a software agent.

Information mining agent periodically scans the hotlists. If a hotlist page has changed, the agent fetches the page, extracts the links on that page and downloads all pages referenced in that hotlist. For each of the referenced pages it extracts some basic meta-information such as URL, title, date of last change, author, synopsis ... Such information is then offered to the human editor to decide whether to include the service in the Library or not. Alternatively, new services may be suggested by the Internet users using a WWW form.

Full text of all services, which are part of the library, are periodically harvested so that the accessibility of the service is verified and that a full text of a service is available to the front-end WWW search engines.

The front end server is the end-users' and editor's interface to the Virtual Library. This server may present the data in several ways, most likely in various tables of contents and inverted indexes. The most powerful way of using the front-end server, of course, are the on-line queries requested by the users interactively. Two types of searches are supported.

- The search in the meta data finds entire services. E.g. the whole ITcon journal is found as a single entry, ASCE's server as another etc.
- The full text search finds individual documents within the services. For example, a paper within ITcon could be found, or a page describing how to join ASCE.

To keep in touch with what is new on the Internet related to construction, an end user would need to periodically visit the library and perform his particular search, for example, "list new items which include the word *masonry*". Alternatively, an end user is able to ask a friendly librarian to keep him/her updated with the items of his/hers interest. For example, one would wish to receive new items related to masonry delivered by email every Friday or every 15th of the month. This job is taken care of by another agent which is a part of the front-end software.

A system built to the specifications defined above is running at http://www.fagg.uni-lj.si/ICARIS/cr/ as Network of Information servers for Civil Engineering (NICE). The end user interface is publicly available. The back-end editorial interface is password protected.

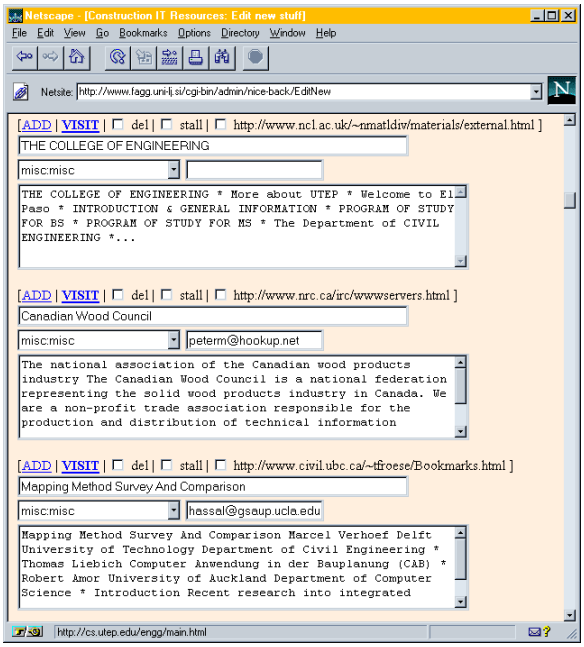

Fig.3: A screen which presents the information mined by the back-end agent to the library editor.

3.2 Distribution of Building Regulations

The overall architecture builds on the three actors which are involved in regulation processing - clients, servers and agents. Fig. 4 depicts one alternative to their relations:

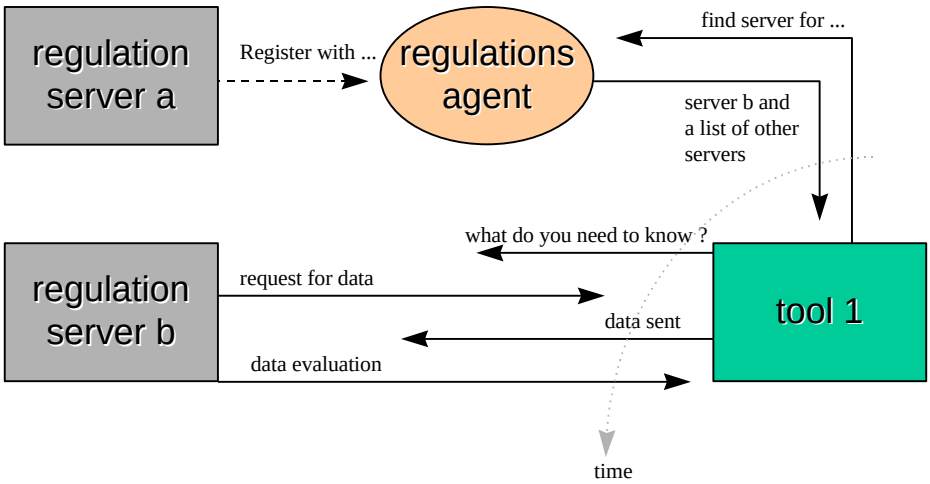

Fig. 4: Regulation servers, regulation agents and regulation clients (tools). An engineering tool needs to consult a regulation. It first asks a regulation agent, who might be able to accomplish that. The agent replies with a list

of servers which can help. The tool than asks the server what kind of information does it need. Server replies with a form a tool must fill in. The tool send the data and the regulation server replies with it's evaluation.

The functions such a system must perform are:

- allowing the registration of a server with an agent;
- finding a regulation server which could help in checking the conformance of a product component;
- establishing the interface between the regulation server and the regulation client;
- mapping of client data into server data.

The components of an requirements server are:

- Regulation specific part includes rule sets, procedures, objects or other components specific to a single regulation.
- Representation specific part are components, which are unique to a representation type, but may be reused with all regulations using a this regulation type. E.g. hypertext browser, rule evaluation engines ...
- Client specific parts are components which are special for each client using the regulation server.
- Agent specific parts are components specific to each agent, which will discover the standard server to the client.
- Application interface. An important feature of a server is that all servers present themselves to the outside world in a uniform way. This means they should have not client or agent specific parts but be compatible with a standardized application interface.

A server should be able to make the following information about itself available to the agents. All this information should be made available in a standardized uniform format:

- access and billing related information;
- server type (hypertext server, checker or checker server);
- server functions (access, calling parameters, bibliographic information). A server may, of course, perform more than one function.

A server should be able to communicate with the outside world on two levels:

- It should be able to publish its schema (e.g. EXPRESS or some easier to parse semantic representation).
- It should be able to exchange product data (e.g. STEP physical files).

Regulation agents should store lists of regulation servers and the uniform descriptions of their functionality. In addition to simply finding a server and passing that information to the client and letting that client arrange a session with the server, agents could be more helpful in storing server details so that the dialogue between server and client could be shorter. Agents could also include other information such as certification data on some regulation servers, performance and reliability evaluation etc.

3.3 Distributed Project Document Management

Similar technology, as applied above to construction pages on the Internet, can be applied to data related to a particular construction project. Construction projects are nowadays handled by several different companies, using different tools, different document management software and organizational approaches. It may well be that in the near future their smallest common denominator will be the same Internet technology. By using the Internet these companies form a virtual enterprise. In the distant future they might share the same project management software or even a common, distributed, object-oriented product model database.

Today, however, they do not use product models, common document management software or document management software at all. Such software induces a certain amount of overhead for checking the documents in- and out of the managed document base and for filling-in the meta-data fields.

We have been working on a simple experiment creates a useful virtual enterprise's document management system by introducing as little overhead and organizational change as possible. The requirements towards the users of the systems are:

- Each company has a Web or FTP server on which it organizes directory structures for each project they are involved with. Within each project they create directories for each of their engineers to publish their work in.
- The directory structure is password protected.
- Directory structure, but not each document, is registered at the central project's database.
- Each company makes a small personnel database available which can map between directory names and people.
- Users of the system publish their files in the respectful directory. They may include some special magic strings anywhere in the file (like !!DRAFT!!) to tag document's state.

A central system running on the computer of the selected "leading" company maintains a full text index of all documents published. It automatically learns meta-data such as author, creation date and state. Special "content" extraction software could be used for each file type, so that, for example, layer names could be extracted out of AutoCAD files, title information from MS Word files etc. A robot scans the databases on an hourly basis and automatically gathers project related documents into the document database. The database is fully searchable by keywords appearing in the files. Also, engineers may ask an agent to let them know about when a particular file would change or when a such and such document would appear in the database.

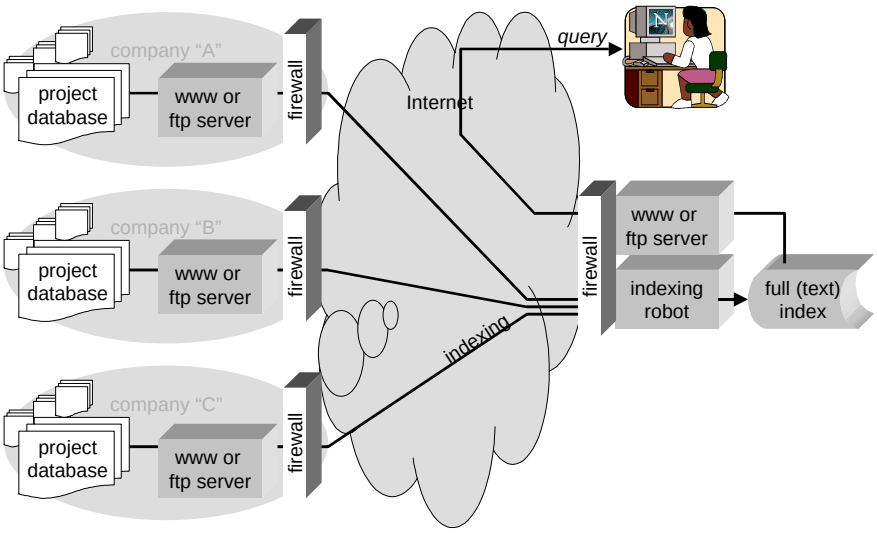

Fig. 5: System architecture of the virtual enterprise's document management system.

4. Conclusions

Technology developed for handling the vast amount of on-line data on the Internet has been modified and applied to the logistics of construction information of global, national and project scopes. We have found out that using smarter IT tools such as full text indexing, agents and robots, we can, to some extent, compensate for the lack of structure and standards for construction information. Even better results can be expected, however, when these techniques are applied to more structured information such as that in the product models or combined with standard classification systems. Within ongoing research projects such as CONDOR and ToCEE we will try to confirm these expectations.

5. Acknowledgments

Some of the presented work has been done within the frames of the Slovenian national research project TIGRA and the European Esprit project ToCEE. The contribution of the partners in these projects as well as that of the funding agencies is gratefully acknowledged.

Most of our Web activities are served off an HP server which was purchased through TEMPUS Joint European Project 3008: Integrated CAD of Earthquake Resistant Buildings and Civil Engineering Structures.

6. References

Amor, R., Bloomfield, D., Langham, M., Formann, J., Jarrett, N. and Goodwin, P. (1996). "The UK industry knowledge base feasibility study", in Z. Turk (editor), Construction on the Information highway, CIB Proceedings, University of Ljubljana.

Etzioni O. and Weld, D. (1995). "Intelligent Agents on the Internet: Fact, Fiction, and Forecast," IEEE Expert, 44-49, August 1995.

Fenves, S.J. (1991). Status, Needs and Opportunities in Computer Assisted Design and Analysis, Structural Engineering International, 2.

Koster, M. (1995). "Robots in the Web: threat or treat?", ConneXions, Volume 9, No. 4.

Riesbeck, C.K. (1996). What next? The Future of Case-Based Reasoning in Post Modern AI, in D.B. Lake (editor), Case-Based Reasoning: Experience, Lessons & Future Directions, pp. 371-388, AAAI Press / MIT Press, USA.

Turk, Z. (1995). Virtual Shareware Library - A WWW based system for cataloging software on the Internet, in R. Holzapfel (editor), Third International WWW Conference Darmstadt'95, Frauenhofer Institute for Computer Graphics, Darmstadt, Germany, 1995.

Turk, Z. (1995-97). WODA-Web Oriented Database; http://www.fagg.uni-lj.si/~zturk/works/wb/.

Turk, Z. (1997). Overview of Information Technologies for the Construction Industry, invited lecture, Icelandic Construction IT Conference, Reykjavik, Iceland, May 9th, at http://www.fagg.uni-lj.si/~zturk/papers/iceland.97/

Webster (1996). "Hypertext Webster 's interface", http://gs213.sp.cs.cmu.edu/prog/webster.