CRC for Construction Innovation (2004) *CPW prolog-based intelligence server*. CRC for Construction Innovation, Brisbane.

CPW Prolog-based Intelligence Server

Technical Report 2002-056-C-TR02

Gerardo Santos Trinidad
CSIRO Manufacturing and Infrastructure Technology
PO Box 56 (Graham Road), Highett, Victoria 3190 Australia
Email: Gerardo.Trinidad@csiro.au

Research Program C
Management and Delivery of Built Assets

Project 2002-056-C
Construction Planning Workbench

PREFACE

The Cooperative Research Centre for Construction Innovation (CRC CI) is a national research, development and implementation centre focused on the needs of the property, design, construction and facility management sectors. Established in 2001 and headquartered at Queensland University of Technology as an unincorporated joint venture under the Australian Government's Cooperative Research Program, the CRC CI is developing key technologies, tools and management systems to improve the effectiveness of the construction industry. The CRC CI is a seven-year project funded by a Commonwealth grant and industry, research and other government support. More than 150 researchers and an alliance of 19 leading partner organisations are involved in and support the activities of the CRC CI.

There are three research areas:

- Program A - *Business and Industry Development*
- Program B - *Sustainable Built Assets*
- Program C - *Delivery and Management of Built Assets*

Underpinning these research programs is an *Information Communication Technology* (ICT) Platform.

The Construction Planning Workbench project in the **Management and Delivery of Built Assets** core area is envisioned to demonstrate the feasibility of deriving draft construction schedules from IFC data generated from 3d CAD models. The project is also geared towards the investigation of methodologies in automatically linking construction schedules with 3d CAD models to produced visualisation and simulation of construction schedules.

The project is a collaborative effort between CSIRO Manufacturing and Infrastructure Technology and The School of Architecture and Built Environment at the University of Newcastle, together with the project's industry partners: Woods Bagot and John Holland. The members of the project team are:

Dr Guillermo Aranda-Mena	(Uni Newcastle)
Ms Fanny Boulaire	(CSIRO)
Mr John Crawford	(CSIRO)
Mr Robin Drogemuller	(CSIRO)
Dr Rod Gameson	(Uni Newcastle)
Mr Peter Hoskins	(Woods Bagot)
Ms Cheryl McNamara	(CSIRO)
Mr Willy Sher	(Uni Newcastle)
Dr Gerardo Trinidad	(CSIRO)
Mr Peter Ward	(Uni Newcastle)
Mr Nick Windmeyer	(John Holland)

CONTENTS

INTRODUCTION

Previous works have tried to integrate the building product model with the construction process model *[Fischer and Froese* (1996)]. They presented three important concepts: building product component, construction process component, and the association between the product component and process component. A possible application of this association between building product and construction process is to automatically generate a construction schedule using some sort of knowledge-based reasoning engine [*Chevallier and Russell* (2001)].

The goal of this technical report is to demonstrate how a logic programming language, such as Prolog, with a rule base, in order to generate draft schedules automatically. The rules serve to capture domain knowledge such as basic construction principles and standard practices with the industry.

Forming a construction plan is a backward reasoning exercise where the required steps (i.e. construction activities) are identified to yield the desired result (i.e. completed building structure). The planning process begins with a result (i.e. a building design). Essential aspects of construction planning include the generation of required activities, and the analysis of the implications of these activities.

The programming language Prolog (for Programmation et Logique or Programming in Logic) has its own reasoning engine [*Sterling and Shapiro* (1988)]. Prolog reasoning engine is a backward-chaining procedure using depth-first search algorithm on ordered facts and rules until a solution is found.

The similarity in the reasoning style of Prolog and the required mode of reasoning in construction planning is a strong motivation in investigating the use of Prolog in implementing a software-based workbench for construction planning.

JOB LOGIC

A construction schedule typically represents a sequence of multiple teams (i.e. trades) that perform individual and distinct work while sharing common workspaces and resources. The logic or rationale behind the sequence of activities in a schedule is referred to as job logic. Job logic includes physical relationship between building elements or components (e.g. a column supports a beam), work team interactions (e.g. concrete work team and carpenters), or safety and code regulations (e.g. workers in a lower level should be protected from activities above them).

The start of some activities obviously depends on the completion of other activities. However, some activities may be independent from other set of activities and may proceed concurrently. Much of job logic follows from well-established work sequences that are standard in the trade. Nevertheless, there is generally more than one approach and no unique order of activities in any significant construction project.

The job logic between construction activities can be divided into fixed logic and soft logic. Fixed logic, such as the relation between installation of the reinforcing bars and pouring of concrete, will not change in any sensible construction process. Soft logic, such as at which end of a bridge should construction start may depend on various factors.

The basic goal of using logic programming is to capture both basic construction principles and local industry practices in order to provide a guideline in generating initial construction schedule.

An illustrative example

Consider a trivial example of a structure consisting of 4 pad footings, 8 columns, a stiffened raft (ground slab with 4 edge beams), a suspended beam and slab floor (with 4 beams), 4 beams (with no slabs), a roof and a wall as shown in Figure 1. All elements are of in-situ reinforced concrete construction except for the wall, which is of block work construction.

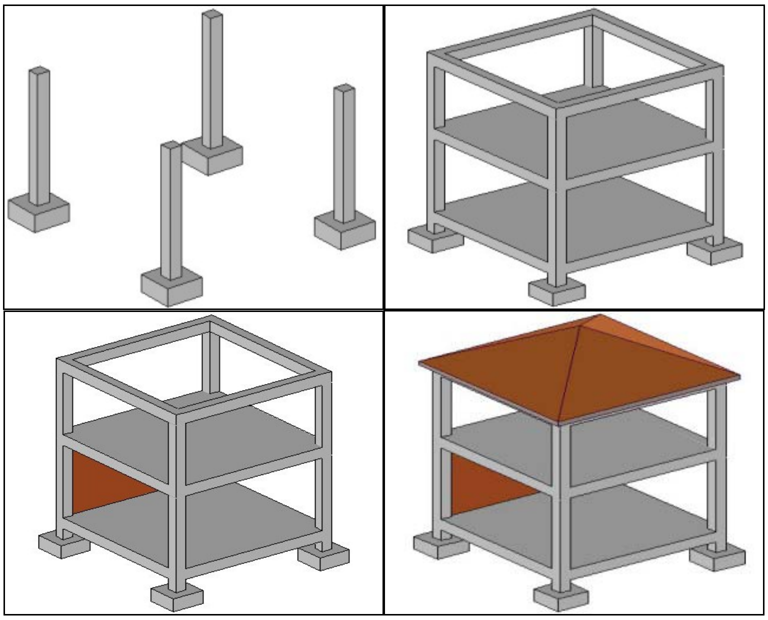

Figure 1: Construction stages of a trivial building as an illustrative example

Observe that the resulting schedule, as shown in Figure 1, allows all in-situ concrete construction to be done continuously. This may be beneficial if a single sub-contractor does all concrete works.

PROLOG STRUCTURES

Prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation. It is a frequently used language in Artificial Intelligence where manipulation of symbols and inference about them is a common task [*Bratko* (1986)]. Prolog consists of a series of rules and facts. Additional facts, called derived facts are computed from rules and known (i.e. given) facts.

A Prolog program execution is basically presenting a query and solving it using known rules and facts. For example, consider the following facts about building elements and their connectivity:

```
% Building Element
% element(id, element-type, storey,…)
element(c201,'column',2, …).          % Element c201 is a column in level 2
element(s301,'slab',3,…).             % Element s301 is a slab in level 3
% Element Connection
% connected(id, d)
connected(c201,s301)                  % Element c201 is connected to s301
```

Consider also the following rule about support relation between elements:

```
% Support Relationship
support(X,Y) :-                      % X support Y if …
        element(X,'column',Lc,…),    % X is a column in level Lc
        Ls is Lc+1,                  % Ls is the level just above Lc
        \+ ground_level(Ls)          % Ls is not the ground level
        element(Y,'slab',Ls,…).      % Y is a slab in level Ls
```

The rule above can be interpreted as *"If a column is connected to a slab, the column is just below the slab, and the slab is not a ground slab, then the column supports the slab"*.

Given a query:

```
support(X,Y)?                        % Find pairs of (X,Y) such that X supports Y
```

Such a query will result to: X = c201 and Y = s301

Other types of relationships between elements can also be defined using rules in Prolog. For instance, a *"constructed together"* relation can be written as:

```
% Together Relationship
together(X,Y) :-                     % X is constructed together with Y if
        connected(X,Y),              % X is connected to Y
        element(X,'beam',…),         % X is a beam
        construction(X,'in-situ RC'),  % X is an in-situ RC beam
        element(Y,'slab',…),         % Y is a slab
        construction(Y,'in-situ RC').  % Y is an in-situ RC slab
```

The rule above can be interpreted as *"Beams and slabs in a reinforced concrete floor are constructed together"*.

Derived relationships such as *"support"* and *'together"* can in turn be use to derived other relationships such as *"constructed before"*.

```
% Precedes Relationship
precedes(X,Y) :-                     % X is constructed before Y if
        support(X,Y),                % X support Y
        \+ together(X,Y).            % X is not constructed together with Y.
```

Elements and Associated Activities

Associating a set of construction activities to a building element or a group of building elements is a critical step in automatically generating workable job logic. The set of activities associated with a particular building element (i.e. column, slab, beam, or wall) depends on the construction method (i.e. reinforced concrete, steel frame, pre-cast concrete, or composite construction).

For example, the following are activities typically associated with reinforced concrete construction written as Prolog facts:

```
% Construction Activities
activity(1,'formwork').
activity(2,'place reinforcement').
activity(3,'pour concrete').
activity(4,'wait and cure concrete').
activity(5,'strip formwork').
```

Activity templates are used to associate the sequence of activities to a particular combination of building element and construction type. For instance, consider the following Prolog rule and facts:

```
% Element-Activity Pair
element_activity(X,A) :-
        element_activity_list(X,L),
        member(A,L).
% Element-Activity List
element_activity_list(X,L) :-          % L is the list of activities associated with X
        element(X,T,…),                % T is the element type of X (i.e. X is a slab)
        construction(X,C),             % C is the construction method of element X
        activity_template(T,C,L).
% Activity Templates
activity_template('slab','rc',[1,2,3,4,5]).
activity_template('column','rc',[2,1,3,4,5]).
….
```

Note that for reinforced concrete columns, the activity template specifies that the reinforcement be put in place before the formwork while for slabs the formwork comes before the placement of reinforcement.

Activities and Resources

Construction resources in traditional construction management views are divided into three categories, namely, labour, material and equipment. These are inadequate for constructability review of the schedule. In recent years, construction space was identified as another important resource type in construction planning. The important role of construction space in construction planning has been illustrated in several studies. Such as: Space occupation as a resource constraint [*Thabet and Beliveau* (1994)], Space scheduling model [*Riley and Sanvido* (1997)], and Time and space conflicts [*Akinci et al.* (2002)].

The definition of construction activities can be extended to include required equipment resources. For example, activity(3,'pour concrete') can be extended to activity(3,'pour concrete','concrete mixer'). Unfortunately, the required equipment may vary depending on the element type and its location. For instance, pouring concrete on a ground slab may also required a trowelling tool/machine while pouring concrete above ground level may require concrete pump and/or crane. Hence, an activity may have two hierarchies of required resources: the minimum resource requirement (i.e. concrete mixer) and the conditional resource requirement (i.e. trowelling tool).

Identifying the required resources for a given element-activity pair is part of the domain knowledge and can be written as Prolog rules as follows:

```
% Element-Activity-Resource Triples
element_activity_resource(X,A,R) :-
        activity(A,_,ResList),
        member(R,ResList).
element_activity_resource(X,A,'trowelling tool') :-
        activity(A,'pour concrete',_),
        element(X,'slab',L),
        ground_level(L).
```

Other traditional resources such as material and labour can be handled in the same way equipment resources are processed as described in the previous section. In fact, even a more abstract concept such as time-space constraints can be dealt with in a similar manner.

INTELLIGENCE SERVER

CPW Prolog-based intelligence server (PINS) is the core of the Construction Planning Workbench (CPW) collection of software tools. The CPW-PINS module is a full-featured Prolog server for applications written in Java. It can potentially support C, C++, Delphi, and Visual Basic applications. It incorporates domain rules in the construction planning area. The rules in PINS are designed to identify:

- A list of building elements supporting a given building element
- A list of building elements that should be constructed together
- Activities & resources associated with a building element
- Precedence relationship between activities (job logic)

Software Installation

The software distribution kit for PINS consist of a single zipped file comprising 18 software components plus a sample MS Access database containing the data associated with the illustrative example presented before. The software distribution kit is available from the CRC for Construction Innovation.

To install PINS simply extract the zipped file into a separate directory (e.g. c:\CPW) as shown in Figure 2.

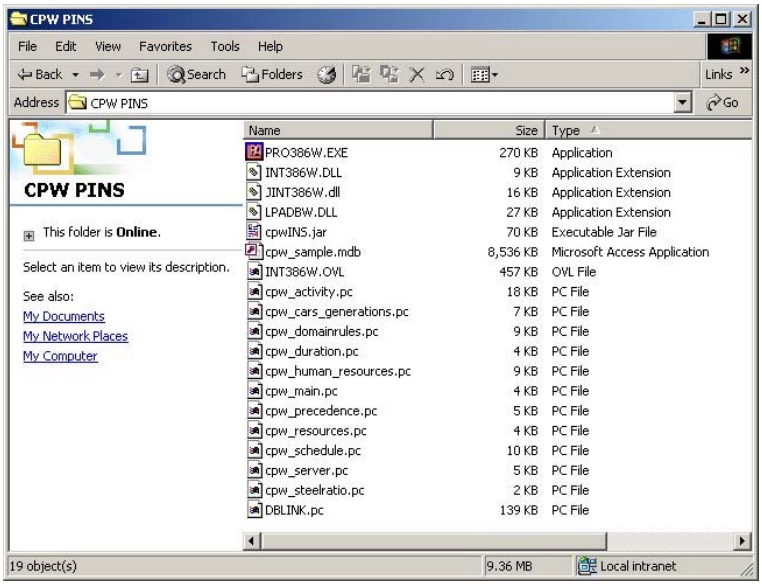

Figure 2: Distribution files for the CPW Prolog-based intelligence server

PINS distribution kit consists of 12 Prolog modules (*.pc) as present. The number of Prolog modules may change as CPW rule base evolves. The kit also contains three application extension modules, which may increase if additional programming languages are supported in the future (e.g. C, C++, Delphi or Visual Basic).

Database Setup

Before PINS can be executed, an ODBC compliant database must be set-up. The CPW PINS software application is designed to work with MS Access, MS SQL Server and Oracle 9i databases.

To setup an ODBC database for CPW PINS, simply run the ODBC Data Source Administrator by going to Setting> Control Panel> Administrative Tools> Data Sources (ODBC). Then go to the System DSN tab (see Figure 3).

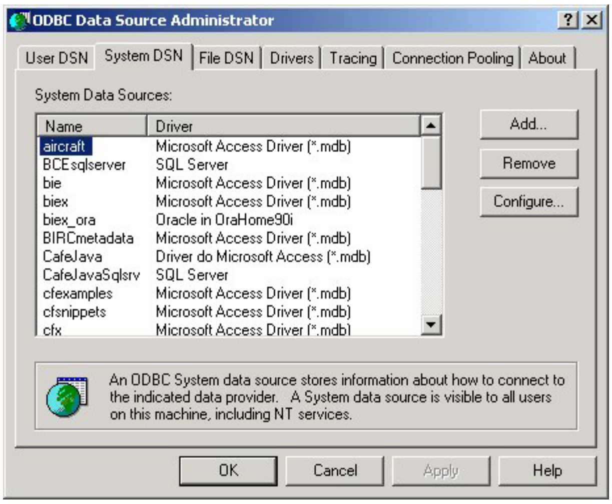

Figure 3: System DSN tab in the ODBC Data Source Administrator

Add a new data source by selecting the appropriate driver for the selected database as shown in Figure 4.

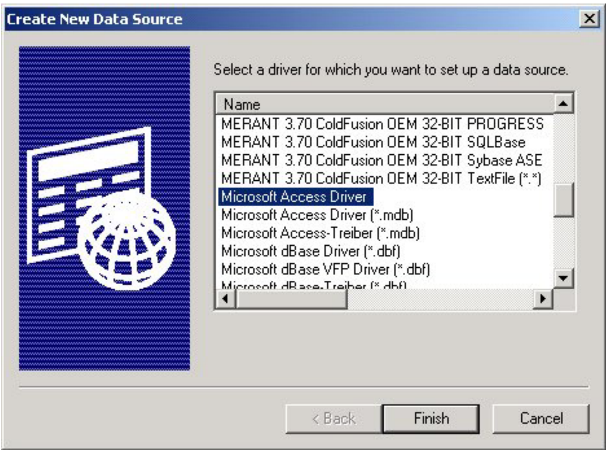

Figure 4: Selecting an appropriate ODBC driver for a selected database

Complete adding the new data source by locating and configuring the selected database. Figure 5, Figure 6 and Figure 7 illustrates how an MS Access database can be configured as an ODBC data source. Note that configuring other databases such as MS SQL Server and Oracle 9i can be slightly different.

To setup the example provided (cpw_sample.mdb), name the ODBC data source as "cpw_store" (see Figure 5), and then select the database location (e.g. C:\CPW\cpw_sample.mdb)

Running CPW PINS

To run CPW PINS simply double-click of the executable Jar file, cpwINS.jar. Alternatively, CPW PINS can be activated using the Start > Run as shown in Figure 8. A shortcut icon can be defined in Windows Active Desktop if CPW PINS is to be executed repeatedly (see Figure 9).

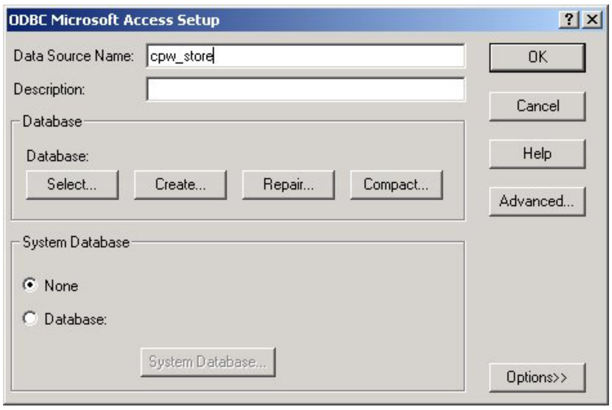

Figure 5: Naming an ODBC data source

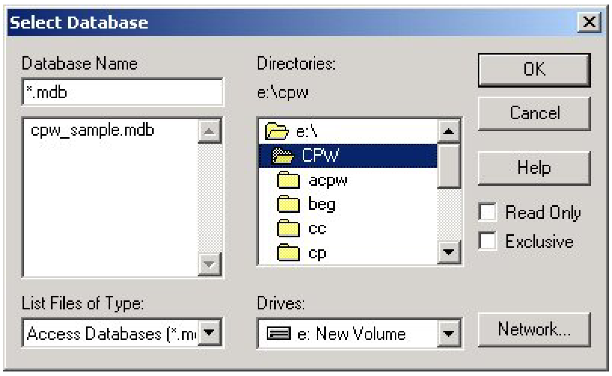

Figure 6: Locating an MS Access database

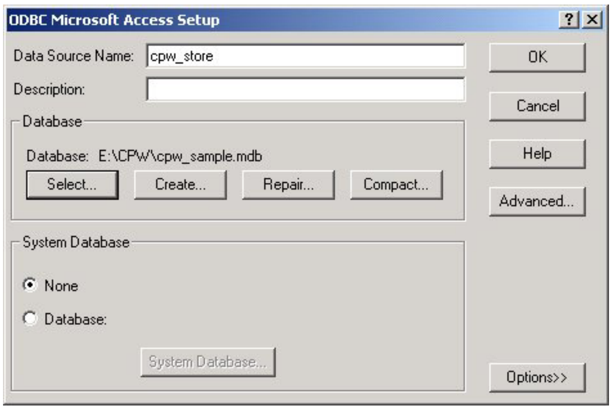

Figure 7: Completing the ODBC MS Access database setup

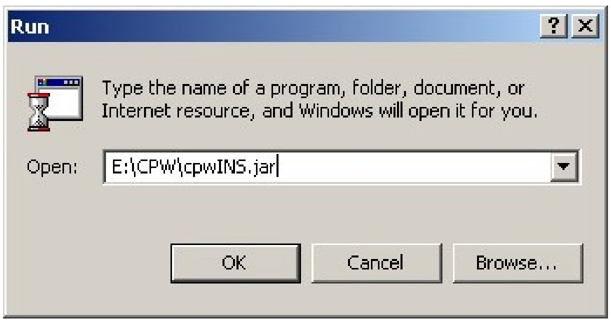

Figure 8: Running CPW PINS from the Start> Run menu

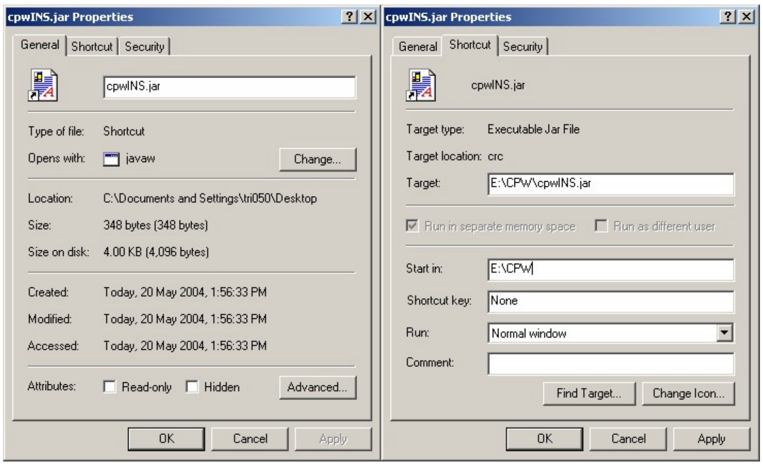

Figure 9: Using a shortcut on the active desktop to run CPW PINS

Using CPW PINS

CPW PINS start with a Java application window shown in Figure 10. The "Load CPW" button instruct the intelligence server to load the rule base and connect to the ODBC data source (see Figure 11). Analysis of the building data can be done once the rule base and database are loaded (see Figure 12). The analysis consists of generating associated construction activities for each building elements and the required resources. The precedence relationships (also called job logic) among the generated activities are also defined during the analysis.

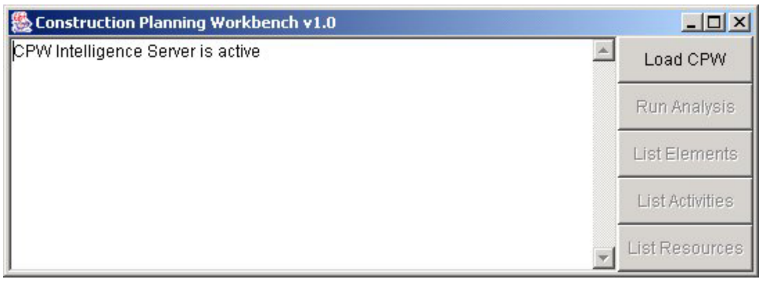

Figure 10: Initial state of the Java application window of CPW PINS

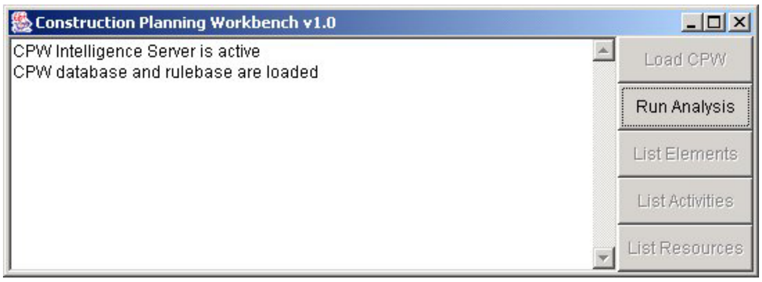

Figure 11: Loading rule base and connecting to the ODBC data source

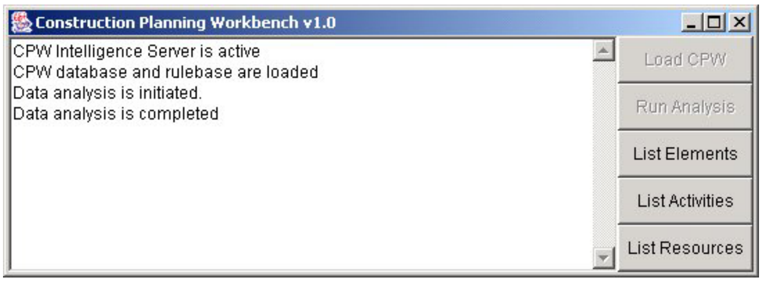

Figure 12: Generating construction activities, resources and job logic

A list of building elements, construction activities and required resources can be obtained after the completion of the data analysis as shown in Figure 13, Figure 14 and Figure 15.

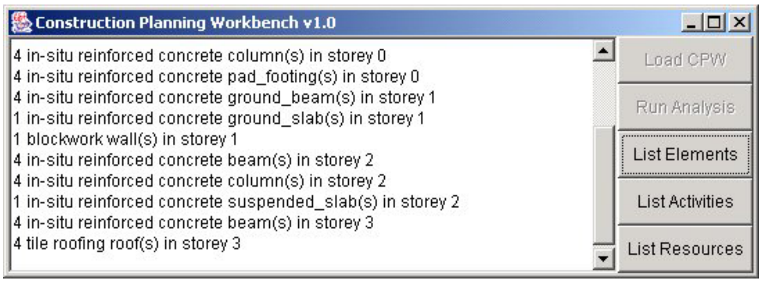

Figure 13: List of building elements

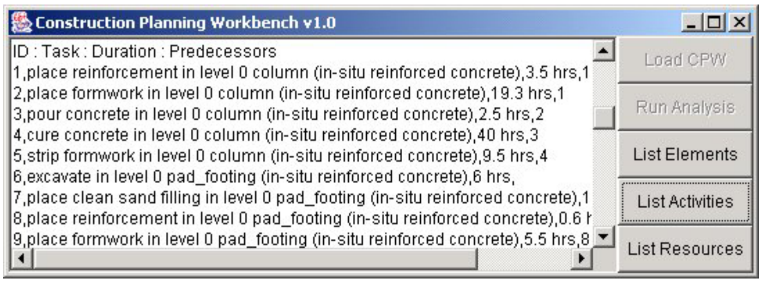

Figure 14: List of construction activities and their predecessors

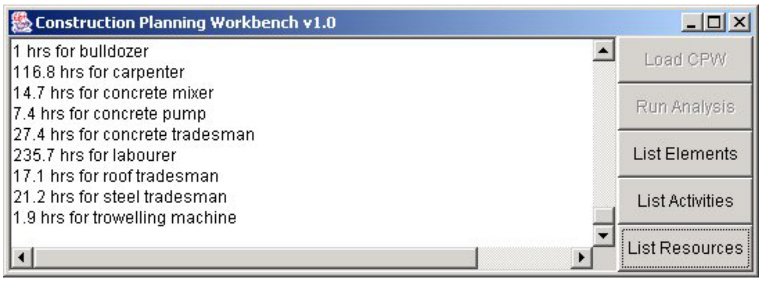

Figure 15: List of required equipment and human resources

MICROSOFT PROJECT

The result of the data analysis from CPW PINS can be imported into Microsoft Project by opening the same ODBC data source used in CPW PINS. The File> Open> ODBC menu is used to open an ODBC data source in MS Project

Pressing the "OK" button in the "Select Data Source" window (shown in Figure 16) activates the MS Project Import Wizard (see Figure 17).

The Import Wizard is a helpful tool for transferring project data between Microsoft Office Project and other programs. It uses import maps, which is a set of instructions that maps out for Project exactly what types of data are to imported, in what order, and their field names in the destination format.

Figure 18, Figure 19, Figure 20 and Figure 21 illustrates the various steps in importing the database table "MSPrj_Tasks" from the ODBC data source "cpw_store".

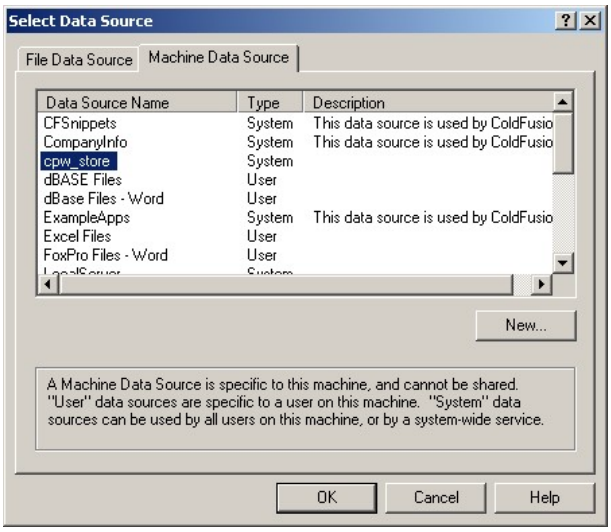

Figure 16: Opening an ODBC data source in MS Project

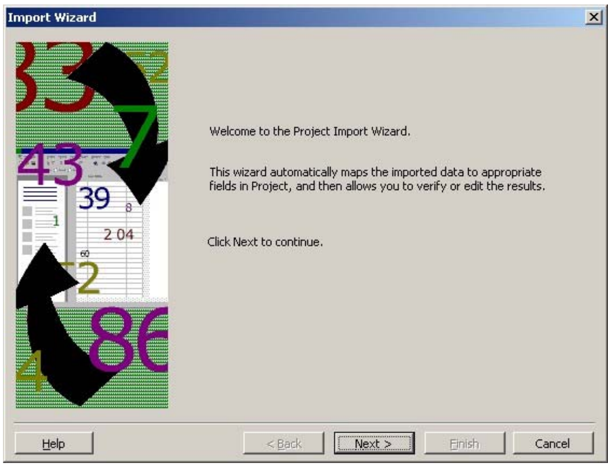

Figure 17: MS Project Import Wizard

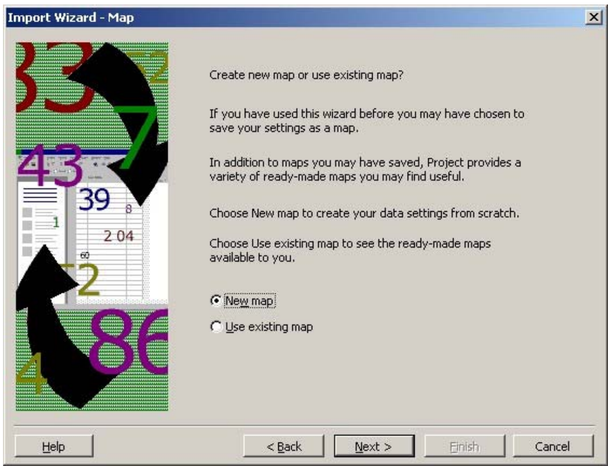

Figure 18: Creating a new data mapping

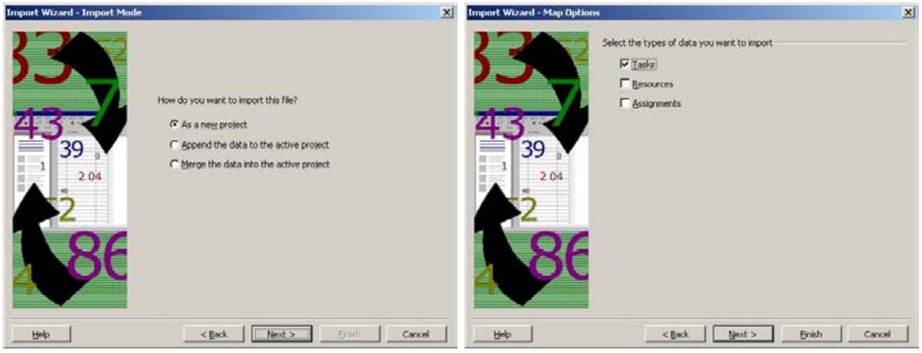

Figure 20: Importing task information into a new project

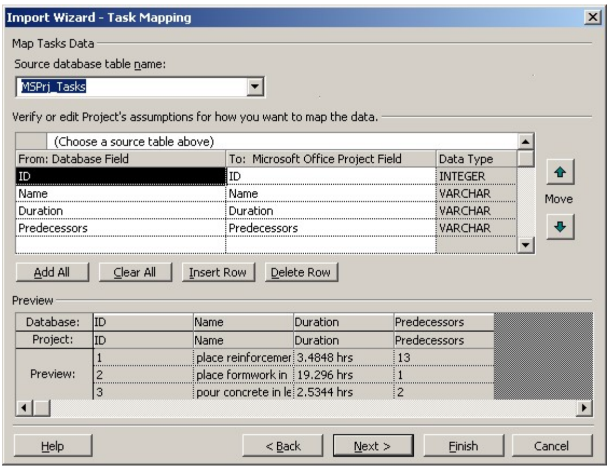

Figure 21: Data mapping for the database table "MSPrj_Tasks"

CONCLUDING REMARKS

This technical report describes CPW PINS, the Prolog-based intelligence server that is the main application in the collection of software modules developed in the Construction Planning Workbench project. The report explains some of the motivation behind the decision to use a logic programming approach in generating draft construction schedule from IFC data.

This technical report describes how CPW PINS can be installed and run on MS Windows 2000 or MS Windows XP. It also describes how the result of the data analysis in CPW PINS can be imported into MS Project for further analysis of the draft construction schedule.

REFERENCES

Akinci B., Fischer M., Levitt R., Carlson R. (2002) "Formalization and automation of time-space conflict analysis", J Comput Civil Engng, Volume: 16, Issue: 2, pp. 124-134

Bratko, I. (1986) Prolog Programming for Artificial Intelligence, Addison-Wesley Publishers Ltd, ISBN 0-201-14224-4

Chevallier, N.J. and Russell, A.D. (2001) “Developing a draft schedule using templates and rules”, Journal of Construction Engineering and Management, September-October, pp. 391-398

Fischer M., Froese T. (1996) "Examples and characteristics of shared project models", Journal of Computing in Civil Engineering, Volume: 10, Issue: 3, pp. 174-182

Riley D.R., Sanvido V.E. (1997) "Space planning method for multistorey building construction", J Construction Engineering Management, Volume: 123, Issue: 2, pp. 171-180

Sterling, L. and Shapiro, E. (1988) The Art of Prolog, MIT Press, ISBN 0262-19250-0.

Thabet W.Y., Beliveau Y.J. (1994) “Modelling work space to schedule repetitive floors in multistorey buildings", J Construction Engineering Management, Volume: 120, Issue: 1, pp. 96-116