Geometric-Based Reasoning System
for Project Planning Utilizing AI and CAD Technologies
by
Ayman Ahmed Morad

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Civil Engineering

APPROVED:

[Signatures]

Y. J. Beliveau, Chairman
M. C. Vorster
A. Myklebust
T. R. Rakes
J. M. De La Garza

May, 1990
Blacksburg, Virginia
Geometric-Based Reasoning System
for Project Planning Utilizing AI and CAD Technologies

by
Ayman Ahmed Morad
Y. J. Beliveau, Chairman
Civil Engineering

(ABSTRACT)

Traditional planning and scheduling techniques have played an important role in system analysis over the last three decades. They provide construction planners with mathematical models to simulate the construction process as an aid in planning and control of complex projects. Although these techniques have been widely used by the construction industry, they possess many limitations.

Researchers and practitioners in the construction industry have followed two directions to overcome most of the limitations of current planning techniques. The first direction has been concentrated on the utilization of state-of-the-art Computer Aided Design (CAD) and 3D computer modeling technology. The objective of their work is to interactively generate and visually simulate the construction process on graphics display. The second direction has been influenced by the potential capabilities of Artificial Intelligence (AI) technology to accomplish "Automated Planning". This group has utilized knowledge-based and expert systems to automatically generate construction plans.

The research proposed here presents a geometric-based reasoning system called KNOW-PLAN. The system integrates CAD and 3D computer modeling technology with AI technology to automatically generate and simulate construction plans. The
system, therefore, can be classified as a third alternative in approaching the planning problem.

The research seeks to utilize geometric data to provide a dynamic sequencing for project planning. The research utilizes object location and object interaction with other objects as the primary source of reasoning for the project plan. The interaction of objects is based on a classification of objects with relation to connection types among them, the zones in which the objects are located, and relationships between the classes with which the objects are associated.

To accomplish the objectives of the research, an overall model called the KNOW-PLAN model has been formulated. This model is formulated to demonstrate theoretically the feasibility of implementing such a model in real-life. The implementation effort has been concentrated on the development of the crucial components of the KNOW-PLAN model using advanced computer applications. The implementation at this level is referred to as the KNOW-PLAN prototype system.
Acknowledgements

I would like to express my sincere appreciation and gratitude to my advisor, Dr. Yvan Beliveau, for his guidance and support during the course of this research. I would like to thank the advisory committee: Dr. Vorster, Dr. De La Garza, Dr. Myklebust, and Dr. Rakes, for their contributions to my education and their willingness to review my dissertation.

I would like to thank Jeff Skolnick for his assistance in developing the sample project used in this research. I would like to thank Hadi and Saed Hakim of Computhink Co. for providing me with computer facilities to do my research during my visits to Kuwait. My deepest and most sincere appreciation goes to my friends: Amal, Fadi, Hala, Halima, Hanin, Jamal, Jihad, Manal, Nameer and Ghada, Sameer, Wafa, and Walid for making my time in Blacksburg enjoyable, the late nights fun, and the good times better.

Finally, my everlasting thanks and love go to my family: my parents, my brother Maher, my sisters: Eman, Lubna, Hala, and Abeer, my brothers and sister-in-law, and my
nephews and nieces who supported me with continuous love and care and to whom I dedicate this dissertation.
Table of Contents

1. Introduction ........................................................................................................... 1
   1.1. Research Objectives ...................................................................................... 4
   1.2. Dissertation Methodology ............................................................................. 5

2. The Planning Problem .......................................................................................... 10
   2.1. Network-Based Techniques ......................................................................... 12
       2.1.1. PERT and CPM .................................................................................... 13
       2.1.2. Decision Models ................................................................................... 15
       2.1.3. Network-Based Techniques: Strengths and Limitations ....................... 16
   2.2. Artificial Intelligence Based Techniques .................................................. 18
       2.2.1. Means-Ends Analysis .......................................................................... 19
       2.2.2. Generate and Test Technique ................................................................. 20
       2.2.3. Production Systems .............................................................................. 24
       2.2.4. AI-Based Techniques: Strengths and Limitations ................................. 25
   2.3. Comparison and Discussion ........................................................................ 29

3. Technology Review .............................................................................................. 33
3.1. Artificial Intelligence and Knowledge-Based Systems ........................................ 34
  3.1.1. Introduction to Artificial Intelligence .................................................. 34
  3.1.2. Problem Solving in Artificial Intelligence ............................................. 35
    3.1.2.1. Breadth-First Search ............................................................... 37
    3.1.2.2. Depth-First Search ................................................................. 37
  3.1.3. Strategies in KBESs ............................................................................. 38
    3.1.3.1. Goal-Driven Control Strategy ....................................................... 39
    3.1.3.2. Data-Driven Control Strategy ....................................................... 39
    3.1.3.3. Selection of the Control Strategy .................................................. 41
    3.1.3.4. KNOW-PLAN Prototype System Control Strategy ......................... 41
  3.1.4. ART (Automated Reasoning Tool) .......................................................... 43
    3.1.4.1. The ART Language ...................................................................... 44
    3.1.4.2. The ART Inference Engine ............................................................ 47
    3.1.4.3. ART's Programming Methodologies .............................................. 49
  3.2. Computer-Aided Design and 3D Computer Modeling ..................................... 51
    3.2.1. CAD Systems Interface Standards ..................................................... 53
    3.2.2. Simulation and Animation ................................................................. 56
      3.2.2.1. Constructability Verification ....................................................... 57
      3.2.2.2. Visual Simulation ..................................................................... 58
  3.3. WALKTHRU 3D Visual Simulation System .................................................. 58
    3.3.1. System Description .......................................................................... 58
    3.3.2. System's Features ........................................................................... 60
    3.3.3. WALKTHRU and KNOW-PLAN ...................................................... 61

4. Literature Review ......................................................................................... 62
  4.1. CAD in Construction Planning ................................................................. 63
    4.1.1. Construction CAE ........................................................................... 64

Table of Contents vii
4.1.2. Simulation System for Construction Planning and Scheduling .................... 65
4.1.3. Stone & Webster's System .................................................. 67

4.2. Artificial Intelligence in Construction Planning ........................................ 68
4.2.1. AI Planners ................................................................. 69
4.2.2. Construction PLANEX ..................................................... 71
4.2.3. GHOST ................................................................. 72
4.2.4. Research at the University of Illinois ........................................... 73
4.2.5. SIPE and SIPEC .......................................................... 74
4.2.6. OARPLAN ............................................................... 75

5. KNOW-PLAN Developed Prototype and Overall Model ............................... 78
5.1. The KNOW-PLAN Developed Prototype System .................................... 81
5.2. The KNOW-PLAN Overall Model .................................................... 83
5.2.1. The KNOW-PLAN Model's Approach ........................................ 84
5.2.2. The KNOW-PLAN Model versus other Work .................................. 87
5.2.2.1. The KNOW-PLAN Model versus CAD-based Planning Systems ...... 88
5.2.2.2. The KNOW-PLAN Model versus AI-based Planning Systems ...... 88
5.2.3. The KNOW-PLAN Model's Components ....................................... 91
5.2.3.1. Stage 1: Generation of a 3D Computer Model .......................... 92
5.2.3.2 Stage 2: Database/Knowledge-Base Creation and Manipulation ..... 92
5.2.3.3 Stage 3: The Dynamic Sequencing Process .......................... 98
5.2.3.4. Stage 4: Interactive Sequence Modification .............................. 101
5.2.3.5. Stage 5: Conventional Scheduling and Reporting ...................... 101
5.2.3.6. Stage 6: Visual Simulation ............................................. 105

6. The 3D Computer Modeling Module .................................................. 107
6.1. Introduction ................................................................. 107
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2. 3D Computer Model Generation</td>
<td>109</td>
</tr>
<tr>
<td>6.2.1. Sample Project using CADAM</td>
<td>109</td>
</tr>
<tr>
<td>6.3. WALKTHRU Interface Modeling Programs</td>
<td>110</td>
</tr>
<tr>
<td>6.4. WALKTHRU Object Definitions</td>
<td>114</td>
</tr>
<tr>
<td>6.4.1. WALKTHRU on the SUN 4/260 Workstation</td>
<td>114</td>
</tr>
<tr>
<td>6.4.2. Object Structures</td>
<td>114</td>
</tr>
<tr>
<td>6.4.3. Build an Object Name File</td>
<td>115</td>
</tr>
<tr>
<td>6.5. Objects' Geometric Data</td>
<td>117</td>
</tr>
<tr>
<td>7. The Knowledge and Data Module</td>
<td>122</td>
</tr>
<tr>
<td>7.1. Introduction</td>
<td>122</td>
</tr>
<tr>
<td>7.2. KNOW-PLAN's Knowledge and Data</td>
<td>124</td>
</tr>
<tr>
<td>7.2.1. Activity Data</td>
<td>125</td>
</tr>
<tr>
<td>7.2.2. Class Data and Class Relationships</td>
<td>127</td>
</tr>
<tr>
<td>7.2.3. Activity Class Relations</td>
<td>133</td>
</tr>
<tr>
<td>7.2.4. Activity Connection Relationships</td>
<td>135</td>
</tr>
<tr>
<td>7.2.5. Zoning Data</td>
<td>138</td>
</tr>
<tr>
<td>7.3. Data Modeling</td>
<td>139</td>
</tr>
<tr>
<td>7.3.1. Entity-Relationship Model</td>
<td>141</td>
</tr>
<tr>
<td>7.3.2. KNOW-PLAN Entity-Relationship Model</td>
<td>142</td>
</tr>
<tr>
<td>7.4. KNOW-PLAN Central Database Implementation</td>
<td>144</td>
</tr>
<tr>
<td>7.4.1. Relational Database</td>
<td>146</td>
</tr>
<tr>
<td>7.4.2. The Central Database File Structure</td>
<td>147</td>
</tr>
<tr>
<td>7.4.3. The Central Database User Interaction</td>
<td>149</td>
</tr>
<tr>
<td>7.5. The Constructability Review Knowledge</td>
<td>151</td>
</tr>
<tr>
<td>7.6. Other Knowledge</td>
<td>154</td>
</tr>
<tr>
<td>7.6.1. User's Defined Knowledge</td>
<td>154</td>
</tr>
<tr>
<td>7.6.2. External Knowledge Sources</td>
<td>155</td>
</tr>
</tbody>
</table>
7.7. Data-Flow in the Central Database ........................................... 156

8. The Dynamic Sequencing Module .................................................. 158
  8.1. The Dynamic Sequencer Concepts ............................................. 159
    8.1.1. Least-Commitment Principle .............................................. 160
    8.1.2. State-Space Representation .............................................. 161
    8.1.3. Object-Oriented Programming Approach .................................. 161
  8.2. The Dynamic Sequencer Procedural Knowledge (Phase I) .................. 162
    8.2.1. External Data Interface Rules ........................................... 163
    8.2.2. Zone Allocation Rule ..................................................... 166
    8.2.3. Geometric Reasoning Rules .............................................. 175
    8.2.4. User and Other Logic Definition Rules .................................. 186
  8.3. The Dynamic Sequencer Procedural Knowledge (Phase II) .................. 187
    8.3.1. Preliminary Logic Refinement Rules .................................... 190
    8.3.2. Final Logic .............................................................. 191
      8.3.2.1. "State-Space" Representation ...................................... 192
      8.3.2.2. Conflict Resolution in the Project Plan Logic .................... 195
      8.3.2.3. The Final Refinement Reasoning Process ............................ 199
  8.4. Miscellaneous Declarative Knowledge ....................................... 205
  8.5. Schedule Calculation .......................................................... 209
  8.6. Creating the Schedule Interface Files ...................................... 213
  8.7. Summary of the Dynamic Sequencer Module .................................. 216

9. The Visual Simulation Module .................................................... 220
  9.1. The Central Database Postprocessor ........................................ 221
    9.1.1. Viewing Parameters ..................................................... 222
    9.1.2. Read Generated Logic and Schedule Data ............................... 225
    9.1.3. Record File Structure ................................................... 226

Table of Contents

x
Appendix B. The Path-Finder System ........................................ 282
  1. Problem Environment Representation ............................... 282
  2. The Objective Function .............................................. 283
  3. The Search Strategy .................................................. 283
  4. Applications for the Path-Finder ................................... 288

Appendix C. The Dynamic Sequencer Rules .............................. 289
  Appendix C1: External Data Interface Functions and Rules .......... 289
  Appendix C2: Zone Allocation Rule ..................................... 296
  Appendix C3: Logic Definition Rules (Geometric Reasoning) ....... 297
  Appendix C4: Preliminary Logic Refinement Rules ................... 310
  Appendix C5: Creating Link, O-Node, and H-Node Schemata ........ 311
  Appendix C6: Final Logic Refinement Reasoning Process .......... 312
  Appendix C7: Miscellaneous Definitions ............................ 316
  Appendix C8: Object-Oriented Schedule Calculation Process ....... 318
  Appendix C9: The Schedule Reporting Rule .......................... 320

Vita ................................................................................. 321
List of Illustrations

Figure 2.1. PERT/CPM Solution for the Jigsaw Puzzle Problem ..................... 14
Figure 2.2. Decision Model Solution for the Jigsaw Puzzle Problem ............... 17
Figure 2.3. Means-Ends Solution for the Jigsaw Puzzle Problem .................. 21
Figure 2.4. Generate and Test Solution for the Jigsaw Puzzle Problem .......... 23
Figure 2.5. Production System Solution for the Jigsaw Puzzle Problem .......... 26
Figure 5.1. The Planner’s Knowledge ................................................. 85
Figure 5.2. KNOW-PLAN Model’s Components ..................................... 93
Figure 5.3. Overall Picture Of the KNOW-PLAN Model ............................. 94
Figure 5.4. The 3D Computer Model Generation Stage ............................... 95
Figure 5.5. Database & Knowledge-Base Creation and Manipulation ............... 99
Figure 5.6. The Dynamic Sequencing Process Stage .................................. 102
Figure 5.7. The Interactive Sequence Modification Stage ............................ 103
Figure 5.8. The Conventional Scheduling and Reporting Stage ...................... 104
Figure 5.9. The Visual Simulation Stage ................................................ 106
Figure 6.1. Sample project Isometric .................................................... 111
Figure 6.2. Data Flow in the 3D Computer Modeling Module ........................ 121
Figure 7.1. The Class Geometric Knowledge .......................................... 130
Figure 7.2. Similar Entity Relationships (Involved) ................................... 131
Figure 7.3. The Class Relation Knowledge .............................................. 134
List of Tables

Table 6.1. Sample Project Components (objects) List .......................... 112
Table 6.2. Objects' Geometric Data (Sample Culvert Project) ............... 120
1. Introduction

Traditional planning and scheduling techniques have played an important role in system analysis over the last three decades. They provide construction planners with mathematical models to simulate the construction process as an aid in planning and control of complex projects. Although these techniques have been widely used in the construction industry, they possess many limitations. They are able to manipulate only the data provided by the planners during the planning and not the knowledge used to generate project plans. These techniques are usually carried out in an unstructured form with considerable reliance on planners' judgement, imagination and intuition. They require abstract visualization of the perceived configurations, characteristics and spatial relationships among various components of the project. This perception results in differences of opinion among various planners.

There is a need for the development of new techniques because of the significant limitations in the current planning tools. These new techniques rely on advanced visual tools in addition to construction knowledge rules defined by planning experts to enhance and automate the planning process of complex projects. The availability of advanced com-
puter technology, both in hardware and software, provides the capabilities for researchers and practitioners in the construction industry to develop such techniques.

Researchers in the construction industry have followed two directions in developing new innovative planning techniques. The first direction has been concentrated on the utilization of state-of-the-art Computer Aided Design (CAD) and 3D computer modeling technology. The objective of their work is to interactively generate and visually simulate the construction sequence on graphics workstations. The second direction has been influenced by the potential capabilities of Artificial Intelligence (AI) technology to accomplish automated planning. This group has developed new planning techniques which utilize knowledge-based and expert systems to automatically generate construction plans.

The research proposed here presents a geometric-based reasoning planning system called KNOW-PLAN. The system integrates Computer Aided Design and 3D Computer Modeling technology with Artificial Intelligence technology to generate and simulate construction plans. The system, therefore, can be classified as a third alternative in approaching the planning problem.

CAD and 3D computer modeling systems can provide powerful graphics computer support for construction planning. This support is concerned with the pictorial representation of the project description and components. These systems can offer the following capabilities to support the planning process in addition to the pictorial representation of the project:

1. They can provide object definition data as represented by a 3D computer model of the designed facility.
2. They can provide a simulation capability to graphically model the construction process.

3. They can provide tools to check for constructability of different planning scenarios using 3D computer animation.

4. They can provide visual communication based on efficient planning technique.

Knowledge-based systems, which are a branch of Artificial Intelligence, are computer programs that use logical relationships to embody knowledge and expertise about a specific domain to perform specialized tasks that typically require human judgement and expertise. Therefore, they offer a suitable environment to capture the knowledge and expertise of the construction planners to perform the typical tasks that the planners perform during the planning process. Such knowledge can be captured, grouped, and stored in the knowledge-base of a knowledge-based system. The stored knowledge represents the different information and constraints needed during the automated planning process. An important class of knowledge-based systems is called Expert Systems. The major function of an expert system is to simulate and augment the problem solving behavior of human experts in a narrow domain. Knowledge about the domain is the key factor in the performance of such expert systems [Feigenbaum 83]. Virtually all expert systems are knowledge-based systems, while the converse is not necessarily true. Based on the above definitions, the KNOW-PLAN system is considered as a knowledge and geometric based (KGB) system and not an expert system.

The primary objective of the research is to utilize the geometric data of the components of a designed facility to improve the reasoning process in construction planning. The construction planning process is done through a dynamic sequencing process that is based on Artificial Intelligence concepts and techniques. The geometric data is acquired
from a 3D computer model of the designed facility. Utilizing this data with spatial relationships among the different components provides excellent knowledge on how a project should be sequenced.

1.1. Research Objectives

The research presented in this dissertation seeks to utilize geometric data to provide a dynamic sequencer for project planning. The research utilizes object location and object interaction with other objects as the primary source of reasoning for the project plan. The interaction of objects is based on a classification of objects with relation to connection types among them, the zones in which the objects are located, and relationships between the classes with which the objects are associated.

The research provides an overall model on how geometric information can be used within the project planning reasoning process. This overall model is structured to utilize the geometric or spatial data as the primary source of knowledge in the project planning reasoning process.

Geometric data if utilized with class, zone, connection type data can provide a project plan with minimal user interaction. However, each different type of project will require additional knowledge to be used in the reasoning process. Also, each individual project will require other specific knowledge. The research provides a method of generating a project plan for any type of project and regardless of the specific type of project con-
straints. For each specific project, much addition of knowledge will be required. The re-
search does not develop this additional knowledge.

1.2. Dissertation Methodology

To accomplish the objectives of the research, an overall model called the KNOW-PLAN model has been formulated. This model is formulated to demonstrate theoretically the feasibility of implementing such a model in real-life applications. The implementation effort has been concentrated on the development of the crucial components of the KNOW-PLAN model using advanced computer applications. The implementation at this level is referred to as the KNOW-PLAN Prototype System. The KNOW-PLAN prototype system is a development effort to prove the concept of utilizing geometric data for project planning.

The main objective of the overall KNOW-PLAN model is to generate a dynamic se-
quence of the construction process using a knowledge and geometric based (KGB) sys-
tem. The final product is a visual simulation of the generated sequence of activities using simulation and animation techniques. The overall model is divided into six stages:

1. Generation of a 3D computer model of the facility;
2. Database and knowledge-base creation and manipulation;
3. Generation of a construction plan by the KGB system;
4. Interactive sequence modification;
5. Conventional scheduling and reporting;

The KNOW-PLAN prototype system represents the actual development of the KNOW-PLAN model as it is implemented by the current research. The KNOW-PLAN prototype system consists of four modules. The four modules interact with each other via an interface environment developed by the researcher. The four modules that comprise the KNOW-PLAN prototype system are as follow:

1. 3D computer modeling module;
2. Knowledge and data module;
3. Dynamic sequencing module;

The 3D computer modeling module is concerned with creating a 3D computer model of the facility to be planned. The 3D computer model consists of different objects that describe the configuration of the project. In this module, the geometric data needed to generate construction plans are extracted. It should be noted that the KNOW-PLAN prototype system utilizes a 3D computer modeling system and doesn’t provide 3D modeling capabilities.

The knowledge and data module is based on a central database management system. The central database controls the definition and manipulation of the data needed by the overall system’s components. The central database provides the communication between the different applications and modules that are integrated by the KNOW-PLAN prototype system.
The dynamic sequencing module is the center piece of the KNOW-PLAN prototype system. The processor in this module is called the "Dynamic Sequencer". The dynamic sequencer is a KGB planning system that generates construction plans based on the provided information in the system.

The visual simulation module is based on the capability of a 3D visual simulation system called WALKTHRU developed by Bechtel Power Co. In this module the simulation of the construction process on graphics display is performed.

The dissertation is divided into eleven chapters. The first chapter provides an introduction to the research. A brief outline is provided for the different directions of research in the area of construction planning using advanced technologies. The scope of the research and a brief description of the KNOW-PLAN overall model and the KNOW-PLAN prototype system are introduced.

Chapter 2 introduces the planning problem. Comparison between traditional planning approach and AI-based planning approach is provided. A jigsaw-puzzle planning problem is used in this chapter to describe the differences between the two approaches. The chapter generalizes the findings of the comparison to the more complex construction planning problem.

Chapter 3 provides a technology review for the different advanced technologies utilized in implementing the KNOW-PLAN prototype system. The technology review includes introduction to the following technologies: Artificial intelligence and knowledge-based technology, Computer Aided design and 3D computer modeling technology, and visual simulation and animation techniques.
Chapter 4 introduces the different directions of research in developing new planning systems that utilize state-of-the-art computer technology. Two directions are introduced. The first direction utilizes CAD technology in implementing new planning techniques. The second direction utilizes Artificial Intelligence technology in developing such systems. A brief description of systems developed using the two approaches are introduced.

Chapter 5 introduces the KNOW-PLAN prototype system and the KNOW-PLAN overall model in specific. The objectives and approach of the KNOW-PLAN overall model are presented. A description of the KNOW-PLAN overall model with a comparison between the KNOW-PLAN overall model and other planning systems introduced in Chapter 4 is included.

Chapter 6 describes the first module in the KNOW-PLAN prototype system. This module is called the "3D Computer Modeling Module". The chapter introduces the WALKTHRU (3D visual simulation) system as the main component in this module. The in/out interface with the WALKTHRU system is described in detail. The geometric data extraction process is described at the end of the chapter.

Chapter 7 describes the "Knowledge and Data Module". This module is based on the central database of the KNOW-PLAN prototype system. The chapter introduces the different kinds of knowledge and data needed by the KNOW-PLAN prototype system. The implementation of the central database as a Relational Model is described.

Chapter 8 describes the keystone module of the KNOW-PLAN prototype system. This module is concerned with the reasoning process that generates construction plans based on many AI concepts. The processor of this module is called the Dynamic Sequencer. The different phases of the reasoning process are described in detail.
Chapter 9 describes the last module in the KNOW-PLAN prototype system. This chapter describes the "Visual Simulator" of the system. The process of creating the necessary files needed to accomplish the interface between the dynamic sequencer and the WALKTHRU system through the central database is discussed.

Chapter 10 presents the generation of a construction plan for a sample culvert project using the dynamic sequencer. Different sensitivity analysis scenarios are included to present the impact of changing some knowledge parameters on the generated construction plan.

Chapter 11 provides a closing chapter for the dissertation. A summary of the KNOW-PLAN prototype system is provided in this chapter. The benefits and contributions of the research are outlined. Directives and recommendations for future work are listed. Finally, a conclusion on the overall dissertation and the research is provided.
2. The Planning Problem

Planning and scheduling are vital elements to the successful execution of any project. [Clough and Sears 79] defined planning as "the devising of a workable scheme of operations to accomplish an established objective when put into action". The planning aspect of any project consists of identifying the work activities needed to achieve project completion and to sequence these events, based on different project constraints and requirements, to obtain peak performance. Scheduling can be defined as introducing time parameters for these events to determine completion dates of individual activities and the overall completion of the project. Together, they form a system designed for maximum utilization of manpower, money, material, and time.

Construction of complex projects can involve thousands of activities. Proper planning and scheduling is required to complete the project at the agreed completion date. The planning and scheduling tasks will be very simple if all the activities are scheduled to follow each other in consecutive order. Unfortunately, the problem is not that simple. Each activity has its own time constraints and often is constrained by the start or finish of other activities. On the other hand, there are many activities which can be carried out
concurrently if they are entirely independent of one another. Accordingly, a typical project plan involves many activities that are independent of one another, and other activities which are dependent on one another. The task of the planning and scheduling process is to model the construction process taking into consideration the project requirements and constraints.

This chapter introduces two approaches to solve the planning problem. The first approach utilizes the conventional planning techniques as represented by Network-based techniques and decision models. These techniques are descriptive rather than optimization or problem solving techniques. The second approach utilizes Artificial Intelligence (AI) based techniques. These techniques are optimization techniques which employ problem solving strategies to generate a solution for the planning problem.

The problem of planning a sequence to build a 3 X 3 (9 pieces) jigsaw puzzle is selected as a general planning problem to demonstrate the strengths and limitations of each approach. The jigsaw puzzle planning problem is considered as a simplified version of the more complex construction planning problem. The outcomes of the comparison between the two planning approaches is used in this chapter to justify the need for an advanced planning system. These advanced systems are based on AI techniques and concepts to overcome the limitations of the conventional network-based techniques. Also, to utilize the state-of-the-art programming technologies to enrich the planning domain in the construction industry.

This chapter is divided into three sections. The first section introduces the Network-based techniques and decision models as a solution to the jigsaw puzzle planning problem. The second section introduces AI-based techniques as a better alternative for solving the problem. The last section compares the Network-based and AI-based tech-
niques with a narration on the resemblance and differences between the jigsaw puzzle and the construction planning problems.

2.1. *Network-Based Techniques*

Construction planners have been searching for a convenient and inexpensive way to represent the project plan in order to simulate and test different planning scenarios. They have met with little success and there remains a need for a modeling system which can be utilized to accomplish this task. Physical models in plastic or wood are able to depict the final product but are of limited value in planning. Mathematical models to simulate the construction process seem to offer a better solution. They are both quick and cheap and enable construction planners to test different solutions to a particular planning problem.

Network programming has been widely applied to project planning for the past three decades. Network-based techniques provide a mathematical modeling system which has had some success but which, on close examination, demands much from the planner. The planner of the network must identify the objectives of the work, must know the level of detail which is appropriate to a particular task and must know how to interpret the results which the model produces [Jackson 86].
2.1.1. PERT and CPM

The purpose of PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method) networking is to aid in the planning and control of one-time projects. The network modeling approach requires that the construction planner explicitly defines the project activities and their interrelationships. This process requires the consideration of: the activities which need to be performed; the sequence in which they will be performed; the resources required; and the time required for each activity.

Two scheduling scenarios are available to represent the sequence of installing the pieces of the jigsaw puzzle using PERT or CPM techniques. The first scenario is the linear scheduling scenario. At each step of the installation process, only one element of the elements which are related (interlock with) to the last installed element will be considered as potential element for installation. This means that the installation process will be performed continuously, no jumping over elements will be allowed. The linear scheduling scenario is not to be confused with the "linear scheduling technique" used in planning repetitive construction tasks. For the purpose of this dissertation, the linear scenario is used to represent a linearly sequenced tasks. Figure 2.1(a) depicts a possible sequence (which is one of many) for the 3X3 jigsaw puzzle under the linear scheduling scenario. The solution is presented as an activity on node network, in addition to a schematic representation of the installation path.

The second scheduling scenario is the non-linear scheduling: The planner will plan for installing one or more elements at each step of the installation process. The sequence is to install all elements which can be installed at a certain step given the installation constraint conditions at that specific step. Figure 2.1(b) depicts the only possible sequence
(a) Linear Scheduling

(b) Non-Linear Scheduling

Figure 2.1: PERT/CPM Solution for the Jigsaw Puzzle Problem
for the jigsaw puzzle under this scheduling scenario if the center piece of the puzzle is the first element to be installed. The solution is presented as an activity on node network, in addition to a schematic representation of the installation path.

2.1.2. Decision Models

Decision analysis provides the theory and methodology for selecting the best decision. This decision being consistent with the objectives of the decision maker and the information available. The selection of a "best" decision will be strongly affected by whatever information is available and known at the time the decision is to be made. Decision models consist of a tree-like structure of "stages" where each stage comprises decision node followed by chance nodes describing the possible outcomes of each available decision. A decision node represents the point at which the decision maker selects one of several alternative actions. The chance node following every possible decision indicates the possible outcomes of the decision [Drake and Keeney 78].

Decision models can be used to represent a set of solutions to the construction planning process. The decision tree starts with the first activity to be scheduled as planned by the user. Given the construction constraint conditions, the decision tree will have the possible activities to be scheduled as chance nodes of the next stage of the tree. Each path in the decision tree will represent a possible sequencing solution. It is important to note that the decision models will only present the possible solutions. The output of the decision model will be used as input to the Network-based techniques..

Decision models can represent the human thinking process. They reflect the results or consequences of the reasoning process which is performed in the mind of the con-
struction planner during the planning process. Figure 2.2 depicts a decision tree for the possible sequencing solution of the 3 X 3 jigsaw puzzle which is based on the linear scheduling scenario given that the center piece of the puzzle will be the first activity.

2.1.3. Network-Based Techniques: Strengths and Limitations

Network-based techniques have played an increasingly important role in system analysis over the last three decades. They are widely accepted in the construction industry and have proven effective for certain type of projects. The major reason for the popular adoption of these techniques by the construction industry is the relative ease with which many different types of projects can be modeled in network form.

Networks provide the construction planner with models to simulate the construction process to aid in planning, scheduling and control of large and complex projects. Although the techniques are widely used in the construction industry, they still include many limitations. Significant among these are:

1. Networks are not optimization techniques. They represent a given system in a descriptive way. They do not provide a solution.

2. The user must be able to conceptualize the system as a network before starting the implementation.

3. The user is somewhat constrained by the options provided in these techniques.

4. Network-based techniques don't provide any problem solving strategy to generate construction plans automatically.

5. The user of the network must identify the objectives of the work. Also, the user must know how to interpret the results produced by the system.
Decision Tree

NOTE: 
"Not all searched paths are shown in the decision tree"

Possible Installation Actions

Figure 2.2: Decision Model Solution for the Jigsaw Puzzle Problem
6. The user must know the level of detail which is appropriate for a particular task.

7. Network-based techniques as planning tools do not incorporate any practical approach to allow for uncertainty in identifying the logic or the interrelationships among the activities in the network [Levitt et al. 88]

In summary, the network-based techniques are problem illustration not problem solving tools. The user must provide the solution and the technique will attempt to illustrate the logic in model form.

2.2. Artificial Intelligence Based Techniques

Artificial Intelligence (AI) provides the potential to enable computers to emulate functions carried out by humans. Attempts to enable computers to perform problem solving began towards the end of the 1950’s [Shirai and Tsujii 84]. Al-based techniques as planning tools are new techniques which have been given great attention by researchers and practitioners in the construction industry. They are thought to have the potential to accomplish automated planning. Substantial research is underway into the use of AI-based techniques in construction planning.

AI-based techniques provide the planner with the tools to represent the knowledge needed in planning a construction project. They also provide the tools to reason with this knowledge and generate a construction plan which cannot be produced directly by network-based or conventional programming techniques. Three AI-based techniques
which can be used to solve the jigsaw puzzle planning problem are presented in this chapter. The mechanism of generating solutions in each technique is briefly described.

2.2.1. Means-Ends Analysis

Means-Ends analysis is a technique widely used in the problem solving strategies of many Artificial Intelligence systems. Means-Ends is a searching technique based on State-Space Transformation representation [Roist 88]. The state-space transformation searching strategy is based on defining the problem environment as a collection of states in which each state corresponds to a unique configuration of the domain elements. This collection is called the "state-space". A start state(s) and goal state(s) are defined to represent the initial state and desired solution conditions, respectively. A set of actions (operators) and rules is defined to identify the conditions that must be met to apply the corresponding action to move from one state to another.

Means-Ends technique searches through available actions and selects an action to be executed that will most reduce the difference between the current state and the goal state. This procedure is repeated to generate a sequence of such actions to reach the desired goals. More details about the state-space transformation are included in Chapter 3, "Technology Review".

The Means-Ends technique can be used as a linear or non-linear scheduling problem solving strategy. For linear scheduling, given a description of the initial state of the domain, and a set of goals, the planner will produce an ordered sequence of actions that will transfer the initial state to one that satisfies the goal description. For non-linear planning, a given description of the initial state of the domain, and a set of goals,
produces a network of actions that will transfer the initial state to one that satisfies the
goal description [Levitt, et al. 88].

The Means-Ends technique can be used to solve the jigsaw puzzle planning problem
under the two scheduling scenarios: linear and non-linear scheduling. Figure 2.3 depicts
the initial state, goal state, actions and the rules to solve the problem using linear plan-
ing. A possible solution is included in the figure. Detailed solution's steps are presented
in Appendix A1.

For non-linear planning, the same strategy can be used. However at any step, all actions
will be considered as potential actions for the next step. The system will execute the
optimal actions after checking all the actions and rules. Then it will update the system
knowledge by changing the current state of the "state-space" problem.

2.2.2. Generate and Test Technique

Generate and Test is another searching technique widely used in AI applications. It is
similar to the Means-Ends technique in the way it represents the "state-space" for the
problem domain. However, the searching methodology is different. Generate and Test
technique is a type of depth-first search that is used to perform classical reasoning by
elimination. Depth-first search is a fundamental searching strategy. The strategy is to
select a path and follow it through increasingly deep levels until the search discovers a
"close enough" solution or the end of the path [Rolston 88]. The searching strategy is
described in more detail in Chapter 3, "Technology Review".
Figure 2.3: Means-Ends Solution for the Jigsaw Puzzle Problem
The Generate and Test technique can be represented graphically in a form similar to the decision trees with the stages considered as searching stages and the chance nodes as space states. The search, therefore, starts with an initial state and generates all possible actions using predefined rules. At each level of the searching process, the system checks if the desired goals are achieved. If this is not so, the system selects one of the new states (according to some rules) and generates a new set of states. The searching process continues using a depth-first strategy until the desired goal is reached. If the path is terminated without reaching a solution, the system backtracks to a higher level in the search tree, select a specific state, and proceed to generate new set of states.

The Generate and Test technique will be used to solve the problem of the jigsaw puzzle under linear scheduling assuming the following: at each step of the planning process, only the elements which have an interrelationship (interlock) with the last scheduled element are considered as potential elements for scheduling (the next sequenced piece must be adjacent to the last sequenced piece).

The Generate and Test technique is not an efficient technique to solve the problem of non-linear scheduling, because there is no need to generate all possible actions. All possible actions should be executed in non-linear form. Figure 2.4 depicts a partial search tree using this technique to solve the jigsaw puzzle problem linearly starting with piece no. 7 as the first piece to be installed.
Figure 2.4: Generate and Test Solution for the Jigsaw Puzzle Problem
2.2.3. Production Systems

Production systems are the most commonly used scheme in knowledge-based and expert systems. Production systems use rules for knowledge representation. A production system consists of [Rolston 88]:

1. An area of memory that is used to track the current state of the problem space under consideration.
2. A set of [IF (conditions) THEN (actions)] production rules.
3. An interpreter that examines the current state and executes applicable production rules.

The condition portion of a production rule consists of a series of condition elements that describe the condition that must be true for the rule to be applicable. The action portion of the rule describes the actions to be taken when the rule executes. The possible actions generally include: activities such as entering new state description in the memory, modifying existing state description, and performing a user-defined action that is unique to the specific production. The interpreter in a production system simply recognizes and executes a production whose condition has been satisfied. To recognize applicable rules, the interpreter compares the condition to the current state of the global memory.

The production system provides a model for human thought because it is discrete, simple, flexible and matches the way people think. Many expert systems are considered as a production systems, where the knowledge is only represented by production rules [Harmon and King 85].
A sample production system was developed to solve the jigsaw puzzle problem under the linear sequencing scenario. The system was developed using VP-EXPERT, an expert system tool which utilizes a Backward Chaining (goal-directed) control mechanism for navigating the rules which represent the knowledge stored in the system.

It is assumed that the next element to be scheduled is not necessary to be related to the last scheduled element. However, the production system can be designed to perform under the assumption of a continuous path of installation, (i.e., always schedule the elements which are related to the last scheduled elements). The list of the program is included in Appendix A2. Figure 2.5(a) depicts a sample solution for the jigsaw puzzle problem under the linear scheduling scenario. The sequence solution depends on the order of the rules. Changing the order will change the sequence solution. Accordingly, new rules can be defined to force the system to generate a solution satisfying specific objectives.

Using an expert system with Forward Chaining (data-directed) control mechanism, a solution for the jigsaw puzzle can be generated based on the non-linear scheduling assumption. At each pass through the rules, all the rules are checked for validity. Each rule with satisfied conditions will have the potential to be executed (fired). After checking all the rules, all the actions of the satisfied rules will be executed. Figure 2.5(b) depicts a sequence which will be generated if the non-linear scheduling scenario is applied.

2.2.4. AI-Based Techniques: Strengths and Limitations

AI-based techniques are attracting many researchers who are involved in developing new planning tools in the construction industry. There are many completed and ongoing re-
(a) Linear Scheduling

(b) Non-Linear Scheduling

Figure 2.5: Production System Solution for the Jigsaw Puzzle problem
search which incorporate the utilization of AI-based techniques in construction to perform various automated planning functions. These techniques provide the planner with the tools to process the knowledge available about a certain domain to solve problems which cannot be solved directly by conventional programming languages.

The main strength of these systems is that they separate the knowledge from the problem solving strategy. This feature allows the user to update the available knowledge (such as the facts, frames, and rules) without affecting the problem solving strategy. The user can also change the problem solving strategy without affecting the knowledge stored in the system. These systems are characterized by their ability to handle heuristic knowledge. This feature allows them to be suitable problem solving systems in uncertain domains.

Many researchers characterized the human problem solving strategy as rules in the cognitive system of the human mind. These systems are, therefore, successful in developing programs that model human behavior [Harmon and King 85]. They also match the decision making process in construction planning which tends to look very much like [IF (conditions) THEN (actions)] rules. The techniques thus can be powerful tools to represent the planning problem and its solution [Levitt 87]. Other strengths are:

1. AI-based techniques provide solutions to the domain problem.
2. AI-based techniques search for and find all feasible construction sequences once the user has specified the first element.
3. Different problem solving strategies may be used to solve the problem: backward chaining versus forward chaining, depth-first search versus breadth-first search, backtracking strategies, etc.
4. AI-based techniques can be modified by defining new rules to check for an optimum solution among other solutions based on specific constraints.

5. The user does not need to conceptualize the whole real system before entering the knowledge. This can be updated whenever new constraints evolve.

6. Most knowledge-based systems can provide the user with explanation facilities to describe "How?" and "Why?" actions are taken by the system during a specific session.

7. As the size of the problem domain expands, the problem solving strategy remains the same. New knowledge (declarative or procedural) is simply added or updated.

8. AI-based techniques can handle the uncertainty in identifying interrelationships among the various activities in the generated construction plan.

AI-based techniques do however contain some limitations:

1. They are newly developed systems. The number of successful applications implemented using these techniques is limited.

2. Some of the available systems are limited to one problem solving strategy.

3. The systems are more expensive than network-based techniques.

4. Some of the AI-based techniques require a knowledge of programming in order to develop an application. This need is going to diminish as commercial development tools and environments are developed.

5. For complex projects, special knowledge engineers are needed to collect and organize the knowledge. Knowledge acquisition is a major stage of any knowledge-based system development process. Knowledge acquisition becomes very expensive as the complexity of construction projects increases.
2.3. *Comparison and Discussion*

This chapter has presented two approaches to present the problem of sequencing the pieces of the 3 X 3 jigsaw puzzle. The first approach uses network-based techniques which are models to describe, mathematically rather than solve real-world problems. The second approach uses AI-based techniques. AI-based techniques are capable of solving problems given certain constraints. The difference between the two approaches is explored. The AI-based techniques are found to be suitable tools to solve the jigsaw puzzle problem.

Network-based planning techniques are elegant and powerful tools for planning today's construction projects. However, a fundamental limitation of these traditional planning techniques is that they are able to manipulate only the data generated by the planning process, not the knowledge used in generating the project plan. AI-based techniques provide the means to generate plans based on stored knowledge. The network-based techniques do not perform a problem solving strategy. AI-based techniques are systems capable of solving problems given certain constraints.

The main advantage of AI-based techniques, such as knowledge-based and expert systems, is that a clear distinction is made between the knowledge and the mechanism used to process the knowledge. The separation of knowledge allows the knowledge to be added incrementally and independent of the problem solving strategy performed by the system. On the other hand, the problem solving strategy used by the system can be replaced by a new strategy without affecting the declarative knowledge stored in the system.
In the Network-based techniques, the user is constrained by the options provided by the technique. They are rigid systems with no flexibility to represent specific knowledge in a generic form. AI-based techniques are flexible programming tools which can represent such knowledge in a simple [conditions-actions] form.

Network-based techniques have no capability of generating project plans. Activity list along with predecessor relationships must be fed to the system by experienced construction planner. However, AI-based techniques are able to store and use heuristic knowledge about resources and other constraints important for detailed construction operations and thereby assist planners in generating detailed plans. Network-based techniques as planning tools are not incorporating any practical approach to count for uncertainty in the identification of required activities or their sequencing logic. In the contrary, AI-based techniques can handle the uncertainty involved in the content of project plans (i.e., alternative plans can be represented, manipulated, or generated at any point in the planning process).

Construction projects need proper planning and control to minimize the risk of failure. During the planning process, the construction planner organizes the available information, facts and constraints to come up with a possible construction schedule. The possible schedule depends on many constraints: physical locations and spatial relationships among various components of the project, availability of resources, constructability constraints, procurement constraints, design constraints and other constraints.

Comparing the construction planning problem with the jigsaw puzzle planning problem, many similarities and differences can be found. Both problems consist of different entities (elements or pieces in the case of the jigsaw puzzle and activities in the case of construction project) which should be scheduled. Both have constraints to be considered
during the planning process. However, the constraints are only simple spatial (physical) constraints in the puzzle and in the case of construction projects they are sophisticated constraints (physical and logical). For example in the jigsaw puzzle planning problem there is no structural constraint problems such as columns before beams constraints as in the construction planning problem. The number of these constraints in construction planning increases dramatically as the size of the project increases. The jigsaw puzzle is a simple version of the more sophisticated construction process.

The approaches used in this chapter to solve the jigsaw puzzle problem can be applied in a similar way to the construction planning problem. However, the problem size is much bigger and more constraints have to be considered.

AI-based techniques as planning tools are expected to have a great impact on the construction industry. They offer systems which incorporate domain specific knowledge to reason with and generate construction plans. The integration of both techniques will be a great advancement in construction planning as they complement each other by offering a problem solving strategy and a modeling system to describe the construction plans.

There is substantial research and development in the application of AI-based systems in the construction industry. AI-based techniques can be an effective tools in construction planning. The procedural knowledge needed to solve the construction planning problem can be easily stored and manipulated in the [IF (conditions) THEN (actions)] form. This means that the popularity of these techniques will increase in the future as tools for solving the construction planning problem. By utilizing the AI-based techniques in the planning process, construction planners can define rules to count for different alternatives. These rules can manipulate the knowledge used in generating the project plan.

2. The Planning Problem
Consequently, the limitation of the traditional planning techniques of manipulating only the data generated by the planning process will be overcome.

KNOW-PLAN a knowledge and geometric based (KGB) planning system to be introduced in the following chapters is an advanced planning system. This system is based on Artificial Intelligence and other technologies (Computer Aided Design, 3D Computer Modeling, Visual Simulation and Animation). The KNOW-PLAN system generates construction plans and visually simulates the construction process.
3. Technology Review

The availability of advanced computer technologies in the area of hardware and software allows researchers and practitioners to develop new Computer-Aided Engineering (CAE) applications. These applications enrich the design and construction processes. This chapter presents a review of the technologies which are utilized in the implementation of the KNOW-PLAN prototype system. A brief description of each technology is presented in addition to contributions to the overall implementation of the KNOW-PLAN prototype system. A series of definitions to prepare for further chapters is included. The different technologies presented are:

1. Artificial Intelligence; Knowledge-Based and Expert Systems;
2. Computer-Aided Design and 3D Computer Modeling;
3. WALKTHRU 3D Visual Simulation System.

The following subsections introduce each of these technologies in turn.
3.1. Artificial Intelligence and Knowledge-Based Systems

3.1.1. Introduction to Artificial Intelligence

Computers have traditionally been used to solve engineering problems that are formalized and analytical in nature. Using conventional programming techniques, a list of sequentially executable statements must be formulated before the computer can solve the problem. This requirement of explicit formalization of the problem into detailed, sequential statements has restricted the use of computer to problems that have well understood solutions. The desire to provide an aid to the solution of engineering problems that are less formalized or understood has led to the recent interest in Artificial Intelligence.

Artificial Intelligence (AI) is the computer-based solution of complex problems through the application of processes that are analogous to the human reasoning process [Rolston 88]. AI technology explicitly attempts to move the reasoning process into the program. Knowledge-based and Expert Systems (KBESs) are examples of the AI technologies. They use logical relationships to embody knowledge about a specific domain and perform specialized tasks that typically require human judgement and expertise. They are well recognized for their potential power to replicate human knowledge.

KBESs typically consist of three main components in addition to other subsystems. The first is the knowledge-base which contains a large collection of facts, object definitions and heuristic "rules of thumb" embodying all of the relevant domain specific information. This knowledge, acquired directly from a human expert, permits the program to
behave as a specialized, intelligent problem solver. The second component is the inference engine which consists of inference and control mechanism. Inference and control direct the system in its use of the knowledge stored in the knowledge-base of the system. The inference engine controls the deductive process. It implements the most appropriate strategy or reasoning process for the problem at hand. The third component is the user interface. The user interface is the subsystem by which the user interacts with the system to define, modify and access the knowledge-base in textual or graphical modes.

There are different approaches to represent knowledge in knowledge-based systems. Rules, facts and schemata (frames) are the most widely used knowledge representation methods. Rules are used to represent relationships in the form of: [IF (conditions) THEN (actions)]. Schemata provide another method for representing facts and relationships. They are well suited to represent descriptive and relational knowledge. A schema is a description of an object that contains slots for all the information associated with the object. Slots may store values, pointers to other schemata, sets of rules or procedures by which the values may be obtained. Schemata allow for rich representation of knowledge. Schemata are linked together to define a hierarchical structure of schemata to allow for inheritance of slot values down the hierarchy.

3.1.2. Problem Solving in Artificial Intelligence

AI techniques have the capability to provide a representation of the problem environment and to search for a solution within the bounds of that environment. State-Space Representation is used in AI applications to formally represent the problem environ-
ment. It is the basis for most of the problem solving strategies. The process of developing the State-Space Representation is described as follows [Rolston 88]:

1. Define the problem environment as a collection of states in which each state corresponds to a unique configuration of the domain elements. This collection is called "State Space".
2. Define the "Start State" within the space. The start state is defined as a set of simple facts or literals that represent the initial conditions of the problem. The start state is used to initiate the searching process.
3. Define the "Goal State" that correspond to an acceptable problem solution. The goal state is defined as a set of literals that represent the desired goal or solution of the problem. The search process terminates when the goal state is reached.
4. Define a set of operators and rules that identify the conditions that must be met to apply the corresponding operator. The operators are a series of potential actions defined with corresponding preconditions (literals that must be true before executing the action); and effects of the action on the world-state (described by addition and/or deletion of literals). Movement from one state to another is accomplished through the application of one or more operator.

The search process to find a solution for the problem is controlled by an overall control strategy. This strategy selects a path in the state space from multiple alternatives. During the searching process, the task of the control strategy is to decide which operator to execute next. Many powerful strategies have been developed to support AI applications [Rolston 88]. Breadth-First and Depth-First Search (two fundamental searching strategies) are described below.
3.1.2.1. Breadth-First Search

The Breadth-First search process starts by expanding the start state by generating all possible successor states using the applicable operators. If none of these states is a goal state, then the next search level will be generated. This level is produced by applying all applicable operators to each state on the just-searched level to produce all possible successor states. This process continues until a goal state is reached. Under this searching strategy, all states at a given level are searched for a solution before any state at a lower level is searched.

This strategy will always find a solution as long as one exists. Also, it will always find the shortest path. however, the number of states that must be generated for every level grows exponentially with increasing depth.

3.1.2.2. Depth-First Search

In the Depth-First strategy, the searching process selects a path and follows it through increasingly deep levels until a solution is discovered or the end of the path is reached. Generate and Test is an AI technique which is a type of depth-first search that is used to perform classical reasoning by elimination. The Generate and test approach relies on a generator and an evaluator. The generator develops complete candidate solutions. The evaluator tests each proposed solution by comparing it with the required goal state using some rules to define constraint conditions.

The search process starts with an initial state and generates all possible actions using the defined operators. The evaluator will check if the desired goals are achieved. If this is
not so, the generator will select one of the new states and generate a new set of states. The searching process continues until the desired goal is reached. If the path is terminated without reaching a solution, the system will backtrack to a higher level in the search space, select a specific state, and proceed to generate a new set of states to continue the search process.

The Generate and Test technique was used to develop the Path-Finder routine of the WALKTHRU system. The routine is described in detail in Appendix B.

3.1.3. Strategies in KBESs

Problem solving involves the search for a solution. The search begins at an initial state of known facts and conditions, and the search ends at a goal state. The solution path consists of all the states that lead from the initial state to the goal state. There are many alternative problem solving strategies that can be implemented using KBES tools and techniques. However, there are basically two approaches to problem solving currently used in KBESs: the derivation approach and the formation approach. The derivation approach involves deriving a solution that is most appropriate for the problem that is at hand. This is done from a list of predefined solutions stored in the knowledge-base of the system. Using this approach, the known facts and conditions are used to derive the most appropriate goal state of the problem solution. The formation approach involves forming a solution from the eligible solution components stored in the knowledge-base. Using this approach, the known facts and conditions are combined to form a goal state. A KBES may use one or both approaches depending on the complexity of the problem being solved.
Two control strategies which are appropriate for the implementation of the derivation approach are described below: Goal-Driven and Data-Driven control strategies.

3.1.3.1. Goal-Driven Control Strategy

GOAL is the end to which problem solving aspires. Goal-Driven control strategy is one of several methods for regulating the order in which inference are drawn in a KBESs. Goal-Driven is synonymous to Model-Directed, Top-Down Fashion, or Backward Chaining of reasoning process in KBESs. Goal-Driven control strategy is a process in which a chain of inferences from “effect” to “cause” is attempted. In other words, hypothesize first and confirm later.

A system uses a goal-driven strategy if it tries to support a goal state or hypothesis by checking known facts. If the known facts do not support the hypothesis, then the preconditions that are needed for the hypothesis are set up as subgoals. This process continues until the original hypothesis is either supported or not supported by known facts. The system may then pursue the validity of another hypothesis in the knowledge-base. The order in which the hypotheses are pursued is predefined.

3.1.3.2. Data-Driven Control Strategy

Data-Driven control strategy is another control strategy that regulates the order in which inferences are drawn in KBESs. Data-Driven is synonymous to Pattern-Directed, Bottom-Up Fashion, or Forward Chaining control strategies. It is called data-driven because it starts from the given data rather than from the assumed hypotheses. In a data-driven system, the program has no a priori knowledge of the possible solution. It
uses the acquired information to evaluate the tree of possibilities, as it progresses through the solution procedure. In this type of control strategy, the reasoning processes from an initial state (at which the program has no knowledge of the solution), through intermediate states (in which the program's knowledge of the solution improved), to a final state (when the program has reached its goal).

The control of data-driven strategy is somewhat more complex than of the goal-driven strategy. Data-driven systems possess little control structure. Instead, the incoming data fires recognition rules that perform actions upon the incoming stream of data and other internal data. The utilization of data-driven control strategy achieve impressive levels of accomplishment if there are enough constraints and redundancies in the data. Care must be taken, however, to avoid pursuit of long search paths obviously at odds with the overall solution being generated.

Data-driven control strategy is best used in “what-if” scenarios. The system begins with a fact and proceeds to search for the rule whose promise is verified by that fact. The conclusion is then added to the working memory in pursuit of the solution.

The main drawback of the data-driven strategy is that it is extremely wasteful to require as input data all the possible facts for all conditions. In many circumstances all possible facts are not known or relevant. By using schemata (frames) as a knowledge representation, this drawback can be overcome. This is due to the advantage of using default values and the if-needed and if-added procedural slot values.
3.1.3.3. Selection of the Control Strategy

When there are no goal states, there is no place for which to back chain, forward chaining (data-driven) control strategy is the only solution. However, if the possible outcomes (goals) are known and if they are reasonably small in number, the goal-driven control strategy will be the more efficient.

In some cases, the goal or solution needs to be constructed or assembled. It may be because the number of possible outcomes is large. In such cases, data-driven control strategy is often used. On other cases, the goal states may not be known when the system begins to reason. Consequently, data-driven control strategy is most useful in situations where there are many hypotheses (solutions) and few input data.

For some problems, it may be more efficient to utilize the two control strategies simultaneously to find a solution for the given problem. This is called Bi-Directional control strategy. As an example: forward search (data-driven) can be used from the initial state while simultaneously using backward search (goal-driven) from the goal state. In summary, the shape of the search process determines whether data-driven or goal-driven control strategies or both is more efficient for a given problem.

3.1.3.4. KNOW-PLAN Prototype System Control Strategy

Building plans from the ground up is an example of reasoning where no solution exists until the reasoning process is completed. Accordingly, few knowledge engineers would probably choose to use a goal-driven control strategy to tackle a complex planning problem. During the planning process, the construction planner organizes the available
information and facts to come up with a possible construction plan (schedule). The possible plan depends on many constraints: physical location and spatial relationships among various components of the project, constructability constraints, procurement constraints, design constraints and other constraints. Therefore, the planning problem is a data intensive domain with a large amount of facts and conditions to be processed in order to generate a construction plan. The overall goal of any planning process is to generate a plan for the project execution. However, before starting the solving process, the detailed characteristics of the desired goal state is unknown.

To generate a construction schedule using AI tools, the initial state of the problem can be defined. Also, there is no a priori knowledge of the possible solution. Accordingly, the system uses the acquired information to evaluate the tree of possibilities as it progresses through the solution procedure. Since the goal state can not be defined explicitly for the planning problem, the forward chaining (data-driven) control strategy is the only choice to start the reasoning process.

The KNOW-PLAN prototype system is implemented using ART “Automated Reasoning Tool” Knowledge-based environment with Forward chaining (data-driven) as the main control strategy. The reasoning process progresses from an initial state (at which the program has no a priori knowledge about the solution), through intermediate state (in which the program's knowledge of the solution improved), to a final state (where the program has reached a possible solution).

In KNOW-PLAN, data-driven approach can be very efficient in evaluating different “what-if” planning scenarios in the interactive sequencing and modification stage. By defining new facts, the system will proceed to search for the rules whose premises are
verified by the new fact. This will assess the effect of changing some parameters of the problem on the generated sequence during the dynamic sequencing process.

3.1.4. ART (Automated Reasoning Tool)

ART is the Automated Reasoning Tool from Inference Corporation. It is a language for knowledge engineering and an inference engine for KBESs. It runs on engineering workstations. ART-IM is a high performance implementation of ART language designed to run on mainframes and PC's. Both products include sophisticated development environments appropriate to computer platform for rapid development and debugging of KBES programs. In this document, the term ART is used to refer to the ART-IM tool-kit which is used in the research.

ART as an integrated knowledge-based building tool has three major components. These components are:

1. Language for knowledge representation and programming;
2. Inference engine;
3. Complete programming environment.

Each one of these components is described below.
3.1.4.1. The ART Language

The power of a KBES is derived largely from the knowledge captured within it. ART provides a flexible language for organizing declarative knowledge into a database, and procedural knowledge into rules. In an ART KBES, the knowledge embodied in rules manipulates the knowledge found in the database (declarative knowledge-base) to reach conclusion.

The database of a KBES contains the available information about the current state of the problem domain. This is the database which contains what is currently known about the domain. Also called the declarative knowledge-base, it contains information in the form of individual facts, or knowledge about the state of the relevant objects in the domain.

The other kinds of knowledge are heuristics or rules of thumb. These are developed from repeated experience in the domain. They usually relate specific conditions with their likely outcomes, consequences, or indicated actions. They embody special-case knowledge of an operational nature. The definition of facts, schemata, and rules are described as follows:

1. FACTS: A fact is an item of information in an ART database. It is a fundamental piece of declarative knowledge. A fact has several components including its number and a sequence of one or more elements representing information contained in the fact. An ART database may contain thousands of facts. The maximum number of facts is limited only by the available memory. A typical fact looks as follows:
which represents the fact ACT-1 precedes ACT-2. "f-3" is the fact number. The first term in the fact "precede" is a relation name and the remaining terms ACT-1 and ACT-2 are values. The relation name identifies the relation that links the values to each other.

2. SCHEMATA: When an application program deals with a complex domain, it is quite common for the application designer to impose a structure on that domain. For example, some of the data elements might be organized into related groups to make them easier to think about, describe, and manipulate as opposed to hundreds or thousands of unstructured facts. The ART schema system is a language for doing this sort of classification and for reasoning about data that has been structured. Each schema in an ART application file represents an object or class of objects, its associated attributes, and its membership in other classes. Thus, the schema system offers an object-oriented representation for declarative knowledge.

In addition to providing a way to structure all or part of a complex database, the system offers a convenient language for indicating that some data items share properties. Schema definitions can be organized into hierarchies in which knowledge about an object can be automatically deduced based on the classes to which it belongs. Object-oriented representation thus helps to streamline a large database that includes repetitive information. A typical schema definition might look as follows:

```
(defschema ACT-1
  (is-a activity)
  (duration 50))
```
Where:
"ACT-1" is the schema name
"is-a" and "duration" are slot names
"activity" and "50" are slot values.

The slots represent individual items of information describing characteristics about the schema object or its relationship with other schemata. A slot in a schema definition has a strong correlation to a fact. The name of the slot is analogous to a fact relation name. It relates the schema object to the slot's values.

3. RULES: A rule is an expression of operational (procedural) knowledge. A typical rule is a short piece of code, that describes the conditions under which the rule is applicable, and the actions the rule should perform under these conditions. A typical rule might look as follows:

(defrule logic-definition
  (higher ?ACT-1 ?ACT-2)
  (same-class ?ACT-1 ?ACT-2) ; Condition(s)
  = >
  (assert (precede ?ACT-1 ?ACT-2)) ; Action(s)

Conditions of a rule are patterns. A pattern is a description of information in the rule should be sensitive to the database. Patterns can be used in a rule to form a connection between existing facts in the database and the rule itself. They are the primary mechanism in preparing a rule for firing (to be executed). Actions are primarily concerned with altering the database by creating new facts or schemata and retracting old ones. Actions may also perform procedural tasks, such as input/output and screen management. Patterns can be used
in the action side of a rule, primarily to construct new facts to modify the database.

3.1.4.2. The ART Inference Engine

ART is a powerful inference engine that reasons with declarative and procedural knowledge captured in the ART language to derive conclusions about an application. Using the knowledge-base, the inference engine attempts to identify known patterns in a specific situations and to apply associated procedural actions that may uncover other known patterns, repeating the cycle until a specific goal is met. This cycle is called the ART’s Reasoning Cycle. The following discussion is based on the presentation of Bruce Clyton in the ART tutorial manuals.

The ART’s Reasoning Cycle: The most important basic concept in ART is that of data-driven computation. In ART, the facts drive the rules. In a typical application, a rule just sits there quietly until some careening fact from the database collides with it and titillates one of the rules’s patterns. At that point, it is said that the pattern has been instantiated, meaning that an instance matching the general pattern has been found. If all of the patterns of a rule have been instantiated, ART creates an “activation” of the rule. The activation of a rule means that ART has identified a unique collection of facts that satisfy the rules’s conditions. ART may find many activations of the same rule at the same time.

Activations are sent to the “agenda”, which is the list of activations currently competing for an opportunity to act. ART evaluates this list and executes only the most important activation from the list. When the “most important” activation is finally allowed to act,
it is said that the rule has been "fired." After firing a rule, ART revises the agenda, taking into account any changes in the database, and executes the most important activation in the revised agenda. This reasoning cycle repeats until interrupted or until ART discovers that the agenda is empty.

When a new fact is added to the database, it immediately runs riot and collides with patterns in all appropriate rules, in a sense moving them all one step closer to the agenda. After the fact has run this gauntlet once, it drops quietly into the database and remains there. One envisions it lying there trying to get its breath back. This is an important point. It means that a rule enters the agenda only by having its last remaining pattern instantiated by a new fact. A newly asserted fact is the only factor that can activate a rule.

**ART's Reasoning Strategy:** Most ART applications construct a "chain of inferences" from an initial set of known facts to some final conclusion. Each time a rule fires, some combination of known facts is used to generate one or more new facts, which in turn become the basis for further inferences. Eventually this step-by-step process creates a set of facts that together form a useful conclusion. This process of building a chain of inferences from facts to conclusions is called "forward chaining" a data-driven control strategy. This control strategy is the main reasoning strategy in ART.

It is also possible to identify a desired conclusion and work backward along a chain of inferences in search of known facts that would support the conclusion. This type of reasoning is called "backward chaining" a goal-driven control strategy. Backward chaining is an abstract idea. It is important to realize that backward chaining does not refer to any particular implementation, format, syntax, or mechanism. It is a conceptual
process in which the inference engine attempts to build a chain of inferences from "effect" to "cause" instead of the other way around. It is the same chain either way. With sufficient effort, most problems can be solved by using either forward or backward chaining. Sometimes there are marked differences in efficiency and convenience that favor one approach over the other.

In the most general sense, backward chaining is used to create needed facts that may be of use to partially-matched rules. The program identifies needed facts (desired conclusions) by generalizing from the patterns of partially-matched rules. Backward chaining makes an attempt to supply appropriate facts to such patterns. In ART-IM this process must be accomplished manually by the programmer.

In ART, backward chaining techniques can be used to lend support and assistance to forward chaining programs. In forward chaining, appropriate facts "drive" the rules to action, and the rules in turn may assert or retract facts. Backward chaining can be implemented in a similar fashion. Instead of "forward" rules there are "backward" ones, and instead of facts there are goals. These goals drive the backward chaining rules just as facts drive forward chaining rules. Forward rules communicate with backward rules by creating goals. Backward rules communicate with forward rules by asserting facts.

3.1.4.3. ART's Programming Methodologies

ART supports several methods or styles of programming. All of which can be productively used in KBES applications. In addition to the tools for knowledge representation, there are also several methods of controlling the actions undertaken by the system (the flow of control). Each of these methods has a different approach to viewing and solving
problems. It is this difference in vantage points which makes one method preferable over the rest for a particular application. The methodologies are:

- **Procedural Programming**: ART Procedural Language supports function calling, simple iteration (for, while) and conditional (if, and, not). Actions are performed sequentially, and in an order largely determined by the programmer. The predominant model is that of a procedure. In this model, a procedure receives data (arguments) and processes them based on the code in the procedure. If necessary, other procedures are called, which entails transferring control, and passing data, to those other procedures.

- **Rule-Based Programming**: The fundamental unit of ART Rule-based Programming is the "rule", which reacts to changes in the surrounding database. Based on these changes, the rules can fire, or execute, in an order based largely on the dynamic ordering of those changes. Rules cannot call other rules, and hence must communicate indirectly by making changes to the database which will, in turn stimulate other rules.

- **Object-Oriented Programming**: The fundamental unit of ART Object-oriented programming is the "object", represented by a schema. Control is managed by sending messages to objects (schemata). The object reacts to the message by searching within itself for a method (which is procedural in nature) appropriate to that message. If an object does not have a method for the received message, it searches to see if it has inherited any appropriate methods from its parents. Once a method has been found, the object carries out the actions associated with the method, usually by reading or writing data local to the object, or possibly by sending messages to other objects.
Object-oriented programming is distinguished from procedural programming by the location of the procedural information it contains. Procedural programming has an inherent separation between the actions (methods) themselves and the data which is used to determine what actions should be taken. Object-oriented programming directly binds the actions (the procedural code) to the data upon which they will operate. By localizing the procedural code with the data upon which it will act, object-oriented programming seeks to avoid the maintenance problems of separation encountered by procedural programming. As the format of data is changed, the methods on the object may also be changed.

3.2. Computer-Aided Design and 3D Computer Modeling

Computer-Aided Design (CAD) is a computing system which makes extensive use of computer graphics. These systems assist in the creation, modification, analysis, or optimization of a design. The computer system consists of hardware and software to perform the specialized design functions.

Most recent CAD systems possess the capability to define objects in three-dimensions. 3D wire-frames and 3D solid models are the two object representation approaches in 3D computer modeling. 3D computer modeling is the process of designing and creating the collection of data (in computer files or databases) that describes the configuration of the designed facility. The 3D computer models can be used for the following tasks [Encarnacao et al. 86]:

3. Technology Review 51
1. determination of geometric forms;
2. proof of geometrical compatibility;
3. determination of physical properties;
4. representation of the model with specific visualization techniques.

The use of 3D computer modeling has become an important part of the design and construction processes. This can be attributed to the fact that effective 3D computer modeling has almost eliminated the use of mock-ups, plastic models and lengthy reports. Also, 3D computer modeling is a natural by-product of the design process which can present the features of new design.

The major component of any CAD system is its database. Nearly all the functions of a CAD system depend on its database. The database of any system represents key elements in computer applications. It is a bank of information that will be used by the application programs to create the desired output. The computer-aided design process results in the creation of a database that describes the components of the designed facility [Stover 84]. CAD systems differ in the way they structure and organize their databases. However, any computer modeling database includes at least the following:

1. basic graphics elements;
2. geometry of the model components and their layout in space;
3. topology or structure of the model (how the various components are connected to form the model).
3.2.1. CAD Systems Interface Standards

CAD systems consist of a number of hardware and software components which have to exchange data. The hardware components include different processors, storage units, and alphanumeric and graphical input/output devices. These devices are interconnected via a data and address channel. The software contains modules to process and store design data to support the communication between different program modules and between the system and the user.

In order to communicate and exchange CAD files between diverse CAD equipments, systems and applications, standardized interfaces are required [Rembold and Dillmann 84]. In the field of CAD, several neutral interface standards have been established on a national and international basis. The following is an introduction to PHIGS which is an ISO and ANSI graphics standard system.

**PHIGS:** Programmer’s Hierarchical Interactive Graphics System is a sophisticated graphics support system that controls the definition and display of hierarchical graphics data. It integrates the host processor, a centralized database and high-performance workstations. It has been designed to manage 2D and 3D hierarchical data in an interactive environment. PHIGS eliminates device dependency. It is designed to off-load host processing by taking maximum advantage of the local capabilities of high-performance hardware systems.

PHIGS is a finite state machine with four state variables with accompanying state tables. Each state describes a different component of the overall system. The four operating states are: PHIGS system state, workstation state, structure state, and archive state
[Brown and Heck 85]. Objects are defined in the database by a sequence of elements, including output primitives, attributes, transformations and invocations of other objects and object part definitions.

PHIGS data are hierarchically organized. An object is defined by one or more structures. Structures relate to each other in a hierarchical network. Hierarchical data structures are commonly called trees. Each structure in a tree is referenced to as a node. The "root" node is the node from which all other nodes originate. A structure is called a parent if it references other structures. A structure is called a child if it is referenced by other structures. Ancestors of a structure are the structures from the root node to the current node.

Structures include different elements. Elements are the smallest unit of graphics information. Ten types of elements are available: output primitives, attributes, modeling transformation, view selection, pick identifiers, labels, name sets, structure invocations, application data, and escape function. Output primitives include 2D or 3D drawing elements. Several types of elements are available: line or polyline, symbol markers or polymarkers, text, fill area or polygon interior, fill area set (or polygon interior and edge), cell array or color grid, and generalized drawing primitives (implementation defined or primitive shapes).

The attribute element describes appearance of output primitives: color, thickness, line type. Structures inherit attributes from ancestors, unless other value is defined within the current structure. Root structure inherits its attributes from PHIGS description table (system defined values). Modeling transformations are attributes that describes the orientation of an object with respect to a coordinate system. There are local and global modeling transformations. Child structure can inherit the transformation from parent
structures if the location is dependent. All the data about different objects are stored internally in PHIGS database files. These files are called "archive files".

A PHIGS workstation is an abstract graphics device. It provides a logical interface through which the application program controls a physical device. There are different types of devices: input, output, and input/output devices. For every workstation type available in a PHIGS implementation, there is a workstation description table describing the capabilities and characteristics of that device. PHIGS also defines "state list" for each open workstation, independent of its type. A state table includes application specific information which can be modified by the user.

The primary benefits of using PHIGS as a graphics support system are:

1. Portability of programs. It is a computer and device independent graphics system. Application programs that utilize PHIGS can be transported between host processors, graphics devices and PHIGS implementations.

2. Sophisticated capabilities save development and maintenance time. Developing applications under the PHIGS environment will relieve the user from developing low level programming to develop input/output routines, modeling transformations, display control and other application requirements. These requirements usually consume long development time. With PHIGS, library subroutines calls perform the required tasks.

3. Better support for advanced graphics workstations. With the ability of PHIGS to control the display of graphics workstations, many PHIGS subroutines address the sophisticated capabilities of these workstations. Shading, clipping, and Z-buffering capabilities are some examples.
4. Increased program performance because of fewer error conditions. Since the user is mainly calling already written subroutines which are device independent, there is a smaller chance of errors in developing applications. This will lead to higher performance and efficiency.

As a recently developed standard for graphics, many new workstations have adapted this standard or its second generation (PHIGS+) to support their graphics capabilities.

3.2.2. Simulation and Animation

Simulation is the process of designing a model (mathematical, logical, or graphical) of a real system and experimenting with this model on a computer. Animation is a specific type of simulation process. It is the process of designing or creating the illusion of motion for the designed model on graphics display.

The use of construction simulation allows planners to develop and rehearse project plans very quickly and graphically demonstrate those plans to others. Real-time animation allows users to build an electronic pattern of a facility in the computer before it is actually built on site. Simulation and animation play a key role in the KNOW-PLAN system. They are used to perform the following:

1. Constructability Verification;
2. Visual Simulation of the construction process using the WALKTHRU system.
3.2.2.1. **Constructability Verification**

Significant advances in design could be achieved if design and construction professionals could test for constructability prior to construction start. Design professionals could test their initial sequencing of activities for constructability, as well as test for future sequencing due to out of sequence components. Lack of constructability is one of the primary causes of rework and subsequent claims in construction. The constructability of different components can be simulated to predict areas of conflict or interference before the construction process is started.

**KNOW-PLAN** utilizes the Path-Finder routine of the WALKTHRU system which will find the optimum path for an object from one position and orientation in a 3D computer model to another position and orientation. The Path-Finder routine optimizes the path subject to motion constraints on the mechanisms which manipulate the object as well as ensuring that there are no interferences along the path between the object and other objects in the path [Morad and Cleveland 89].

The Path-Finder routine can check the constructability of different sequencing scenarios before generating the final sequence by **KNOW-PLAN**. This is accomplished by simulating the movement of critical objects from an initial position to their final position within the 3D computer model. Hence, the routine enables the user to check if it is possible to install these objects without interfering or hitting other objects to verify the constructability of the objects. Accordingly, different sequencing scenarios can be simulated to generate the constructability constraints for specific objects. The constructability constraints can be input into the knowledge-base of the **KNOW-PLAN**
system as user-defined logic. The Path Finder Routine is introduced in Chapter 7 Section 7.5.

3.2.2.2. Visual Simulation

The simulation provides capability for the user to put the simulated construction activities into motion, so the plan can be visualized over time. With real-time animated images, displayed in color with perspective, the construction process can be animated on the screen of a graphics workstation to review the generated sequence. The WALKTHRU system is used by the KNOW-PLAN system for the visual simulation of the generated construction sequence.

3.3. WALKTHRU 3D Visual Simulation System

3.3.1. System Description

WALKTHRU is a real time, three-dimensional (3D) simulation system. It is developed by Bechtel Power Co. It allows the user to interact with existing 3D computer models in a lifelike manner. With WALKTHRU and a color graphics workstation; the user can move through the 3D computer model, seeing the physical objects as they would appear in the real world. WALKTHRU gives the user complete control of body and head orientation as movement is simulated through the model. Real-time animated images, dis-
played in color with perspective, provide a lifelike simulation of the user's movements. The user also can control the motion of individual objects in the model.

WALKTHRU files are generated from 3D computer models created on a CAD system or some other 3D modeling system. WALKTHRU currently supports files from IGDS (INTERGRAPH CAD System), 3DM (Bechtel modeling system), and IGES systems. An additional text file interface is provided to allow the definition of objects with arbitrary polygonal surfaces.

WALKTHRU's Record function can record any path of motion, as well as the motion of the 3D objects the user observes. The replay function can reconstruct this combined viewer and object motion, resulting in an animated "tour" of the model that the user can replay at any time. The user can also replay just the motion of the objects, allowing viewing of the effects of moving through the 3D computer model while the objects are in motion. Equipment installation, replacement and operations activities, and entire facility designs can be visualized before anything is built. This can be done more cost effectively than with static drawings or plastic models.

WALKTHRU runs on all Silicon Graphics workstations. However, in order to function properly, the workstation should have the following features: 8MB memory, 24 bit planes of image memory, Z-clipping, and Z-buffering. Bechtel is developing the WALKTHRU version to run on the SUN 4/260 workstation with Raster Technology GX4000 graphics hardware and software. This version is not complete yet. However, it has all the functions needed by the KNOW-PLAN system.
3.3.2. System's Features

WALKTHRU works with two primary input/output devices: a dial/button box, and a mouse. The dial/button box makes it easy to control the walk through the model. Alternatively, the mouse allows the user to use menus to control all the functions available on the dial/button box. Some WALKTHRU functions are controlled from the keyboard. WALKTHRU performs the following basic functions:

- **View Control**: The user can control all aspects of the view of the model. The direction or speed of travel can be changed, or the head can be turned while the body is on motion.
- **Display Control**: The user can control how the model is displayed. At any point, the user can switch between wire frame and shaded images for any or all objects. At any time the user can display a map that will show the viewer position relative to the model.
- **Object Motion**: WALKTHRU has a number of options for moving and positioning individual objects within the model. The user can define an object hierarchy so that the motion of one or more objects is dependent on the motion of another object. Also, the user can create customized menus for moving objects in the model.
- **Interference Detection**: While the user moves objects around, WALKTHRU can check if there are any interferences with other objects in the model. Interferences are highlighted on the screen.
- **Measuring**: The user can measure between any two points in the model and obtain values of distance, angle, and delta x, y, or z.
- **Record/Playback**: WALKTHRU can record a sequence of key frames that contain the current view parameters, object positions and orientation and the change in time.
from the last frame. The frames are stored in a Record file that can be read by WALKTHRU's REPLAY function. REPLAY interpolates between key frame parameters, and replays the sequence in real time.

- Shaded Image Animation: WALKTHRU can replay the viewer and object motion one frame at a time, shading each frame and sending the shaded frame to a video recorder. The final product is a videotape of the record motion, displayed in shaded image.

3.3.3. WALKTHRU and KNOW-PLAN

The KNOW-PLAN prototype system extracts the geometric data of different components of the designed facility from the 3D computer models of the designed facilities within the WALKTHRU system. This extracted data is used by the dynamic sequencer of the KNOW-PLAN prototype system to reason with the spatial relationships to generate the logic among various objects (components) of the facility. Therefore, the 3D computer models of WALKTHRU plays an important role in the knowledge input to the knowledge-base of the dynamic sequencer. Also, WALKTHRU is used by the KNOW-PLAN prototype system to verify the constructability of certain planning alternatives. It is used to visually simulate the construction activities of the designed facility.
4. Literature Review

The construction industry is looking for the development of new planning techniques to overcome the significant limitations in current planning techniques. These new techniques rely on advanced visual tools in addition to construction knowledge rules defined by planning experts to enhance and automate the planning process of complex projects. The availability of advanced computer technology, both in hardware and software, allows both researchers and practitioners in the construction industry to develop such techniques.

Researchers in the construction industry have followed two directions in developing new innovative planning systems. The first direction has been concentrated on the utilization of the state-of-the-art Computer Aided Design (CAD) and 3D Computer Modeling technology. The second direction has been influenced by the potential power of Artificial Intelligence (AI) technology to accomplish automated planning.

The objectives and approach of each direction with examples of related research are presented in this chapter.
4.1. CAD in Construction Planning

The potential of CAD systems in construction presents a great opportunity for integrating engineering and construction processes in a more cost effective way. The combination of computer graphics, animation, and 3D computer modeling has proven to be an extremely effective tools for real-time animation in support of engineering and construction from the conceptual design to the construction process.

CAD systems are widely used as planning tools in the manufacturing industry. The construction industry, meanwhile, is still not utilizing these systems efficiently for planning and scheduling of construction projects [Morad and Beliveau 90]. Graphical computer support for construction planning is concerned with the pictorial representation of the project components, handling equipment, and construction sequence. A technically advanced CAD system can offer the following added capabilities to enhance and support the planning process:

- It can provide object definition data represented by a 3D computer model of the facility available to the planner;
- It can provide a simulation capability to model the construction process graphically;
- It can provide visual communication based on efficient planning techniques.

With these capabilities, the construction plan can be generated by interactive communication techniques. The construction process can be simulated visually on advanced graphics workstation using the generated sequence and the 3D computer model of the designed facility.
There are many active groups of researchers and practitioners in the construction industry who are involved in the development of new planning systems based on CAD and 3D computer modeling technology. The R&D divisions of several construction and engineering firms such as Bechtel, Hitachi, and Stone & Webster are among these active groups.

4.1.1. Construction CAE

Construction CAE [Hayashi et al. 87] is a joint-effort system development between Hitachi and Bechtel. It is being developed to explore the concept of using 3D computer plant model to aid in planning and managing power plants construction. The system enables the user to perform "What-If" planning scenarios to arrive at optimized sequences, schedules, and resource profiles. It is an integrated set of programs which post-processes the 3D computer plant model.

The Construction CAE can process 3D computer model components of up to five different material commodity types. The operation for Construction CAE is divided into four modules:

1. Tag number database management module;
2. Preliminary sequencing module;
3. Construction simulation module;
4. CPM, resource, and commodity database module.

The preliminary sequencing module consists of two stages. In the first stage, the tag item of each commodity will be sequenced independently according to four installation criter-
ria: higher objects before lower objects; objects further from the point of access before closer; large objects before smaller; and finally, linearly connected commodities should be installed linearly.

The generated sequence for each commodity will be stored in a file as a separate sequenced list. In the second stage, interactive sequencing by the user will be performed. The user will select tag items from the five sequenced lists to create an overall mixed commodity construction sequence.

In the third module, the user will be allowed to rebuild the 3D computer model. The construction simulator is a fully interactive software tool, allowing for user input and immediate feedback through graphics display and animation. The simulation system will be performed in real-time on a graphics workstation.

The last module produces CPM schedules, commodity installation plans, summary schedules, manhour expenditure plans and maintains a commodity tracking database.

4.1.2. Simulation System for Construction Planning and Scheduling

This system was developed by Bechtel Western Power Co. and Bechtel Construction Inc. [Simons et al. 88]. It is a newly developed system that provides facilities for planning and scheduling of construction and/or maintenance activities. The system combines graphics capabilities for the visualization of the plant model, tools for interactive simulation of construction activities, knowledge-based decision support features, and an interface to a CPM program for automatic generation of project schedules.
The system consists of four major programs. The first is the Preprocessor. This program provides the link between the 3D computer model of the plant and the central project management database. This program performs the following tasks: converts the 3D computer model data into a simplified format; loads the data into the database; automatically generates unique construction tag identifier; associates modeled plant components to tagged objects; and performs design change monitoring and control functions. It also loads volume definitions in the database.

The second program is for the creation of handling equipment library. This utility program is used to create and maintain a database or library of material handling equipments. The equipment defined in the library is then available for use within the construction simulator.

The third program is the construction simulator. This program is the centerpiece of the system. It provides a set of interactive tools for simulation and planning using 3D graphical displays of the modeled plant. Several methods are provided within the simulator to assist the user in quickly establishing a possible sequence for tag object installation. Automatic rule-based sequencing may be applied to some or all the tag objects for a given commodity. Manual sequencing is supported by allowing the user to point to objects on the screen with the mouse. The user will select the type of precedence relationship to assign for the logic among the objects from a menu.

The last program is the CPM schedule processor. The simulation provides the tools for planners to develop and playback construction simulation plans. The result of which are written to the project management database. These results and other data from the database provide the CPM processor with the data elements necessary to generate and process a schedule in a batch processing mode.
The system incorporates three limited-scope expert systems. They are implemented as an embedded advisors. These three advisors are: handling equipment selection advisor; temporary facilities usage advisor; and safety consideration advisor.

4.1.3. Stone & Webster’s System

This system is part of an overall computer aided engineering system developed by Stone & Webster Engineering Corp. [Zabiliski 88]. The overall system is a computerized database system to integrate graphics, engineering data, and all other information necessary for the engineering, design, construction, operation, and management of any type of facility.

In the created facility design database, each object is an entity that can be accessed and manipulated independently. Starting with the 3D design model, the company developed methods for articulating the final model into a sequence of construction steps. All objects or components are expressed as a 3D solid elements. Each component is represented as an overlay on the graphical presentation. The component’s parts are then disassembled into a linear sequence, representing the desired construction steps.

Each step in the construction sequence model represents an activity in the construction network. The quantity of material to be input in place during each activity item is defined in the database. Accordingly, the system determines the activity duration based on the quantity and productivity rate.

Consequently, the construction planner can develop the sequence and the duration of every path in the construction network interactively using the graphics model and the
database. By specifying parallel paths and dependencies between paths, the planner can define the entire network, providing the input to a CPM processor.

4.2. *Artificial Intelligence in Construction Planning*

Knowledge-Based and Expert systems (KBESs) applications in construction have been given much attention in recent years by major construction firms and research institutes. Levitt describes four reasons for using KBESs in construction [Levitt 87]:

1. construction is an experience-based industry;
2. construction decisions must be made quickly;
3. construction decisions involve managerial issues;
4. construction automation needs smart robots.

For these reasons, KBESs are offering valuable new capabilities to provide decision support for construction engineering and management.

The development of KBESs for planning and scheduling has been initiated by many researchers and practitioners. There are many completed and ongoing research efforts aimed at the utilization of KBESs in construction planning. KBESs in planning and scheduling can be classified as follows:

1. KBES for activity duration estimation;
2. KBES for logic generation;
3. KBES for schedule generation;
4. KBES for project monitoring;
5. KBES for schedule analysis;
6. KBES to support construction planning such as resource allocation and selection of method of construction.

The third category is of primary interest for this research. This research focuses on the generation of construction schedules from a 3D computer model geometric data and knowledge stored in the knowledge-base of a KGB system. Thus, the following subsections introduce some of the ongoing research and work already done which are involved, directly or indirectly, in the third category. The approach taken by most of these research programs is to capture the construction planning experts’ knowledge and expertise and store them in the knowledge-base of a knowledge-based system. This knowledge is then used for reasoning and can automatically generate construction plans.

4.2.1. AI Planners

STRIPS [Fikes and Nilson 71] is one of the earliest implementations of AI in the area of linear planning. STRIPS generates linear plans based on the “means-end” analysis paradigm. The paradigm operates by identifying the difference between the current state and the goal state and then heuristically selects operators (actions) that will reduce the identified difference. STRIPS combines means-ends analysis with the use of a “goal-stack.” The goal stack contains both goals and operators that are intended to achieve the goals. In addition to the goal stack, it uses a predicate logic representation of the current system state and a set of operators that is used to modify the state. The generated plan is an ordered set of actions that transfer the initial state to the goal state.
ABSTRIPS [Sacerdoti 74] is an enhancement to STRIPS by providing a search process through an abstraction space. The system introduces hierarchical representation of the actions. This approach enhances the plan generation process by defining importance levels among the different possible actions. The importance value of each action (with its preconditions and effects) is used to select among potential actions at a certain stage of the search process.

INTERPLAN [Tate 75] uses subgoal promotion (moving the action whose preconditions were negated of the offending action) to correct for interactions between subgoals. This permits INTERPLAN to derive optimal plans for a wide range of problems than STRIPS [Levitt et al. 88].

NOAH (Nets of Action Hierarchies) [Sacerdoti 77] is a hierarchical planner that uses least-commitment principle to produce nonlinear plans. Problem solving in NOAH is accomplished by developing a procedural net, a sophisticated structure for representing both action and state information. The problem solving process begins with a single node that represents the goal to be achieved. This parent node includes a pointer to a collection of functions that expands goals to subgoals. Each of the subgoals is linked to every other subgoal and back to the parent node.

Each subgoal also includes a pointer to a set of functions that are used to further expand the subgoals. They also include a declarative representation of the effect, if any, of executing the expression functions. NOAH employs the least-commitment principle by expanding subgoals without any regard for ordering until the possibility of an interaction is detected. The possibility of such interaction is detected by one of several critics (small programs that review plans for potential conflicts).
NOAH cannot backtrack from bad decisions. NONLIN [Tate 77] was developed as an extension of NOAH to include a decision graph to permit backtracking and alternative resource decisions. NONLIN detects interaction by analyzing the underlying goal structure of the planning problem.

O-PLAN [Currie and Tate 85] is an extension of NONLIN which enhances its abilities in the area of project scheduling and resource management.

MOLOGEN [Stefik 81] generates nonlinear plans for genetic experiments. MOLOGEN utilizes a flexible control structure and explicit formalism for constraints on the activity plan. MOLOGEN extended the least-commitment principle and constraint propagation through the use of constraint posting. It partially describes objects by simply identifying relations between domain elements rather than explicitly selecting objects. These relations serve to constrain the possible choices to be selected during the search process.

DEVISER [Vere 83] is an extension of NONLIN to plan and schedule the operations of an autonomous unmanned space craft probes with temporal time constraints. The system can constrain activity schedules with the timing of external events.

4.2.2. Construction PLANEX

Construction PLANEX is a knowledge-based construction planning system [Hendickson et al. 87]. It is part of an integrated environment of processes and information flows for the vertical integration of architectural design, structural design and analysis, and construction planning. The integrated environment, being developed at Carnegie Mellon University, makes use of a number of AI techniques. The processes are implemented as
a KBES. A blackboard architecture is used to coordinate communication between processes. The global information shared among the processes is hierarchically organized is an object-oriented programming languages [Fenves et al. 88].

Construction PLANEX generates schedules for modular hi-rise buildings, including excavation, foundation, and structural construction. It generates project activity networks, cost estimates and schedules, including definition of activities, specification of precedence, selection of appropriate technologies and estimation of durations and costs. The system takes as input: specification of the physical elements in the design, site information, and resource availability. It produces as output: a complete plan, a provisional schedule and a cost estimate.

The program knowledge-base consists of a large number of knowledge sources. Each knowledge source is input in the form of a decision table, which is later converted into a network of frame schemes. These frames represents: design elements, element activities, project activities and frames to store information at the project, sector, block, and floor levels. The tabular input format allows for easy input and updating of the knowledge-base. A user interface and graphical simulation of the project are also provided.

4.2.3. GHOST

GHOST (Generation of Hierarchical networks for cOnSTruction) is a prototype knowledge-based project network generator [Navinchandra et al. 88]. It is implemented using IMST a knowledge-based expert system development environment developed at M.I.T. GHOST is a part of a larger integrated knowledge-based environment for construction planning called CONPLAN that is currently being developed at M.I.T.
GHOST uses blackboard architecture of knowledge-based system for facilitating the integration of diverse sources of knowledge through a global database (the blackboard). It takes as input a list of construction activities and produces as output a schedule by setting up precedence among activities. GHOST knowledge-base consists of several knowledge sources called critics. GHOST has the following critics: critics that know about physics; critics that know about construction; refinement critics; and critics that check for redundancy.

GHOST starts by first generating an optimistic, but probably infeasible, network at a certain level of abstractions. Then it proceeds to minimally modify the network to make it feasible. The optimistic network is one in which all the activities are carried out in parallel. It then uses the stored critics to modify the network by introducing linearization wherever activities cannot be done in parallel. This approach is used uniformly over all stages of plan generation. GHOST produces a temporally good network.

4.2.4. Research at the University of Illinois

This research attempts to provide support to construction planning. A knowledge-based system approach is used that involves three major phases: knowledge acquisition, formalization of the elicited knowledge, and implementation of a prototype knowledge-based system. This work is currently focused on the analysis of mid-rise building construction [Echeverry et al. 89]

A prototype KBES is being implemented that incorporates a subset of the acquired knowledge. The knowledge-base of this prototype consists of several modules that interact following a blackboard architecture. The first module consists of a hierarchical
breakdown of the different building systems into building components. The attributes of these components are input manually to the system and stored in a knowledge-base frames as objects. The physical relationships that are considered among the different components are: supported-by, covered-by, weather-protected-by, and embedded-in.

The second module supports a hierarchical breakdown of the building into activities. Another module contains the knowledge to perform allocation of crews (which are defined as a hierarchy of objects) that checks the preliminary schedule against different criteria. Another module contains the knowledge to determine activity logic. The sequencing of the different activities is supported by two operations. The first operation generates precedence constraints for each activity. The second operation propagates these constraints to produce activity sequencing.

This prototype is being implemented using KEE programming environment. The knowledge is implemented in terms of rules and frames (objects). All the scheduling calculation are performed by a procedural module that follows an object-oriented approach.

4.2.5. SIPE and SIPEC

SIPE (System for Interactive Planning and Execution Monitoring) [Wilkins 84] is an advanced classical planner for general purpose planning. It uses hierarchical, constraint-based, interactive planning to achieve greater domain independence. SIPE is based on the means-ends technique. It utilizes a least-commitment approach to control the search process to transfer the initial state of the planning problem to a desired goal state. SIPE has several features which distinguish it from the other AI planners introduced before. Among those significant features are [Levitt et al. 88]:
1. it is an interactive system, the user is able to watch and guide the planning process;
2. it has a rich set of formalism for knowledge representation, including both frame-based and logic-based representation;
3. It provides the ability to express deductive rules for deducing the effects of actions;
4. It introduces a new technique for reasoning about resources for efficiently detecting and remedying harmful parallel interaction.

SIPEC is a construction planning customized SIPE [Kartam and Levitt 89]. The system implements SIPE to plan a multi-story office building with repeated cycles of operation. The system utilizes a least-commitment approach to delay decisions concerning ordering links and variable instantiations until the system has as much useful information as possible for making decisions.

SIPEC uses frame hierarchy to store information about the building components. The installation of the different components is performed by generic actions (operators) attached to the frames representing the components. The ordering links among activities are based on physical laws and principles. Typical dependency principles implemented in the system are: Gravity support (law of nature) and enclosed-by.

4.2.6. OARPLAN

OARPLAN (Object-Action-Resource Planning System) [Darwiche 89] is a prototype planning system that generates construction plans from a description of objects that comprise the designed facility. OARPLAN is based on the notion that activities in a
project plan can be viewed as intersections of their constituents: objects, actions, and resources. Planning knowledge is represented as constraints based on activity constituents and their interrelationships. OARPLAN attacks the planning problem by expanding the scale of a high level activity into more detail as needed.

OARPLAN takes as input a description of the facility to be constructed and generates a hierarchical project plan. The ultimate goal of this research is to produce a planning system that can interpret descriptions of a facility at several stages of refinement in CAD format. The system would then render immediate feedback on construction planning and scheduling application of the evolving design to a designer. The prototype system currently requires that the building component and their relationships be entered as frames by the user. An effort to extract component description from CAD models is currently being done at Stanford University.

The means of representing an activity is adapted from PIPPA system developed by Marshal in 1987 [Darwiche 89]. Marshal defined an activity as an action that applies to a product component and that needs resources. Each element of <action> <object> <resource> is called an activity constituent.

Activities are represented at different levels of abstractions based on the levels of abstraction of the included actions and objects. The relative degree for abstraction of an action or an object is defined by its position in an abstraction hierarchy. In the current version of OARPLAN, only action and object constituents are represented and used in reasoning. Resources will be represented in a future version of the system.

Actions are either simple or compound. Simple actions are those that can be performed directly without refinement. Compound actions are those that can be elaborated further
by a set of lower level actions. Similarly, object constituents of activities can be of different types and grain sizes. There are simple and compound objects.

To generate a plan, OARPLAN starts with a high-level activity at the final level of the plan. Different knowledge sources contribute to the development of the plan by either elaborating each activity to lower activities or posting some sort of a dependency onto the plan. Elaborating activities creates multiple levels of a plan, each with a different level of abstraction. Dependency constraints apply to activities at the same level of a plan. When knowledge sources are unable to post any further modification to a plan, the resulting plan is considered to be in its final form.

OARPLAN is implemented using a blackboard-based architecture environment called BB1. The system is organized as a set of blackboards each having its own functions. The different blackboards of the system are: facility, action, plan, elaborated, and dependency blackboards.
5. KNOW-PLAN Developed Prototype and Overall Model

Traditional planning and scheduling techniques have played an important role in system analysis over the last three decades. They provide construction planners with mathematical models to simulate the construction process as an aid in planning and control of complex projects. Although these techniques have been widely used in construction industry, they possess significant limitations. Advancement in computer technology such as Computer Aided Design (CAD) and Artificial Intelligence (AI) are offering new and potentially powerful capabilities to develop new innovative planning and scheduling techniques. These techniques have the potential to overcome most of the limitations that exist in current planning and scheduling techniques.

The current state-of-the-art in planning requires a specialized individual called "The Planner (Scheduler)" (or several of these) who plans the activities and sequencing. Typically, the planner will interview important people such as managers, supervisors, and subcontractors. From their interviews, a set of lines and boxes will come forth to
describe the “Plan”. In addition, a “Pile” of output will be provided. The pile will identify resources to be committed to each task and a written description of the project activities and logical relationships. The existing process of planning and scheduling requires significant knowledge in planning and schedule output interpretation.

Wouldn't it be nice if the scheduling process was understandable on an initiative basis (like in a visual context)? The typical scheduling techniques of the 1950's and 1960's provided a form of visual scheduling. A typical office trailer would have faded paper tacked to the walls showing lines, milestones, and other pertinent data. A set of multi-colored pins would present progress to date; and, by contrasting pin colors, problem areas were apparent. A carpenter foreman, carpenter, or laborer could intuitively understand the status of the job. This long ago scheduling system has serious deficiencies. However, it did have the important ingredient of visual representation and ease of interpretation.

Current computing capabilities, such as CAD and 3D computer modeling, can provide today a visual scheduling tool through graphics animation. The animation capabilities can be, in turn, controlled and managed through stored knowledge about the project. The project can then be represented to users in an interactive medium visually showing how the sequencing logic would be presented in the standard formats commonly used, understood, and shared by the planner.

On the other hand, Artificial Intelligence technology offers computing environment for developing systems which incorporate domain specific knowledge (planning and scheduling knowledge) to reason with and generate construction plans. The knowledge needed to solve the construction planning problem can be stored in a knowledge-base of a knowledge-based planning system to be used later during the reasoning process to au-
tomatically generate construction plans. The integration of CAD and AI technologies is expected to have a great benefit towards the direction of solving the problem of the construction planning process.

This research proposes a new planning system which combines CAD and 3D computer modeling technology with AI technology to generate and visually simulate construction plans. The system, thus, can be considered as a third alternative in approaching the construction planning problem, in addition to the two directions already introduced in the previous chapter. This third alternative proposes to integrate CAD and 3D computer modeling and AI technology.

The primary objective of the research is to utilize the geometric data of the components of designed facility to improve the reasoning process in construction planning. The construction planning process is done through a dynamic sequencing process. This process utilizes the geometric data and other knowledge to provide the project plan. The geometric data is acquired from a 3D computer model of the designed project. Utilizing this data with spatial relationships among the different components provides excellent knowledge on how a project should be sequenced.

The scope of the work in this research is defined at two levels: “conceptual” and “technical”. At the conceptual level, the research demonstrates the feasibility of integrating CAD and AI capabilities to enhance and automate the construction planning process. This integration provides a radical departure from the conventional planning approach. The research provides a basis for this integration. The integration effort is based on common representation of the project components as objects with attributes which are passed from a 3D computer modeling system to a knowledge and geometric based (KGB) system using a proper interface.
The research provides a conceptual platform for future extensions and enhancements of the KNOW-PLAN model. The research explores, conceptually, the source of knowledge which resides in 3D computer models and CAD images of designed facilities. Such knowledge relieves the knowledge-based system developer from the manual input of the facility description and topology, components breakdown, constructability of components, and quantity take-off. In addition, 3D computer models provide visual tools to enhance communication and understanding of the project description and configuration.

At the technical level, the research has implemented crucial components of the proposed model. This implementation forms the basis for future extensions to develop the KNOW-PLAN overall model. The interface between CAD and AI systems has been developed through the development of a central database management system. The KNOW-PLAN prototype system demonstrates the feasibility of the conceptual objectives of the model. It can be considered as a cornerstone for the development of a practical knowledge-based planning system using CAD technology.

5.1. The KNOW-PLAN Developed Prototype System

A KNOW-PLAN prototype system has been developed to prove the concept of utilizing geometric-based data for reasoning in the automated planning process. The KNOW-PLAN prototype system is briefly introduced here. The prototype system is more completely detailed in Chapters 6, 7, 8, and 9.
The KNOW-PLAN prototype system is a partial implementation of the proposed KNOW-PLAN model. It consists of the crucial components of the overall model as they are developed by this research. The KNOW-PLAN prototype system provides the basis such that the overall future model can be accomplished. The Implementation concentrates on the components which define the concepts of the KNOW-PLAN model.

The keystone of the prototype system is the dynamic sequencer. The dynamic sequencer is a KGB system performing the automatic generation of the project plan based on the available knowledge. A user interaction subsystem is developed to provide the interface linkage between the 3D computer model and the KGB system.

The outcome of the development of the KNOW-PLAN prototype system is divided into four modules. The following four chapters will describe in detail each module. The developed modules are as follows:

1. The 3D computer modeling module;
2. The knowledge and data module;
3. The dynamic sequencing module;
4. The visual simulation module.

The KNOW-PLAN overall model is described in the following sections as a representation of what the fully developed system would look like. The KNOW-PLAN model is not fully implemented; however, the KNOW-PLAN prototype system is a significant effort towards the development of the KNOW-PLAN overall model.
5.2. The KNOW-PLAN Overall Model

An overall KNOW-PLAN model has been formulated to show what a final working system should look like. This model has not been fully developed. The objective of the KNOW-PLAN model is to capture construction planners’ knowledge and expertise and store it in the knowledge-base of a KGB system. The planners’ knowledge and expertise come from different sources. However, the main source of knowledge needed to define the construction plan is the perception of the topology and spatial relationships among various components of the project. Such relations can be extracted directly from the geometric data stored in the 3D computer model of the project.

The main concept of the KNOW-PLAN model is to generate a construction plan by reasoning with captured knowledge, extracted geometric data, and other information input by the user. The generated plan can be modified interactively by the user to perform different “what-if” planning scenarios. The “what-if” planning scenarios will enable the planner to assess the impact of various alternatives in construction methods, availability of resources, off-site assembling, etc. on the generated plan. The model incorporates an advanced visual simulation system to visually animate the construction process using the generated plan and 3D computer model of the project. The animation will help in identifying any conflict in the generated plan which is not resolved by the limited knowledge available in the knowledge-base of the KGB system.
5.2.1. The KNOW-PLAN Model's Approach

Construction planners are professional experts who are knowledgeable about the process of sequencing the different activities of construction projects. This process is based on planners' previous experience, intuition, and perception of the configuration of the facility to be constructed. During the planning process, they organize the available information in an unstructured way to come up with a feasible project execution plan (sequence). This plan usually differs from one planner to another depending on the individual talent, experience, and accessibility to the needed information.

Figure 5.1 shows the different information processed in the mind of the construction planner during the process of developing a project plan. The information, which are related to different issues, are processed simultaneously in the mind of the planner during the process of developing the plan. Some of the issues considered in this process are: physical location and spatial relationships among various components of the facility, design constraints, constructability constraints, resource constraints, the plan's level of detail (abstract levels), method of construction, weather constraints, laws of nature, stability constraints, laws and regulations, plan parallelism, expected durations, and project specifications and requirements. In addition to the planner's individual capabilities, such as: experience, talent, intuition, perception, and imagination which play an important role in this complicated process.

Planners' knowledge and expertise, needed to develop the project plan, come from different sources. However, the main source of knowledge needed to define the project plan is the perception of the spatial relationships among various components of the project.
Figure 5.1: The Planner's Knowledge
in space. With the availability of CAD systems and 3D computer models, such relationships can be acquired from a 3D computer model of the facility.

CAD and 3D computer modeling can provide powerful graphics computer support for construction planning. This support is concerned with the pictorial representation of the project description and components. These systems can offer the following capabilities to support the planning process in addition to the pictorial representation of the project:

1. They can provide object definition data as represented by 3D computer model of designed facility;
2. They can provide a simulation capability to graphically model the construction process;
3. They can provide tools to check for constructability of different planning scenarios using three-dimensional animation;
4. They can provide visual communication based on efficient planning technique.

KGB systems offer a suitable environment to capture the knowledge and expertise of the construction planners to perform the typical tasks that the planners perform during the planning process in a structured unbiased way. Such knowledge can be captured, grouped, and stored in the knowledge-base of a KGB system. The stored knowledge will be able to represent the different information and rules needed during the automated planning process.

Accordingly, most of the issues which are considered by the planner during the planning process can be acquired (from the 3D computer model), captured (from construction planners), and stored in the knowledge-base of a KGB system. The approach of this research is to develop a model for generating construction plans. Using a KGB system,
the model will accommodate the captured knowledge based on the planners' expertise and their problem solving strategy. This expertise and strategy will be used in the reasoning process to generate automatically the sequence of activities (plan) for the project. The sequence will be generated using the extracted spatial relationships among various components of the facility from 3D computer model and the rules defined in the knowledge-base of the KGB system. With graphics capabilities of a CAD and Visual simulation system, such plan can be animated on graphics display to show pictorially the construction process.

5.2.2. The KNOW-PLAN Model versus other Work

The literature review in chapter 4 provided a brief description of some of the prior and ongoing research in the area of applying either CAD or AI technologies to construction planning and scheduling. Most of the described systems are utilizing only one of the two technologies in their implementation of new planning and scheduling systems. None of the systems has integrated or has significant integration of CAD and AI to come up with a new approach for enhancing the planning and scheduling process.

The integration of CAD technology with AI technology is expected to play an important role in developing new planning systems to enhance and enrich the planning process of complex construction projects. KNOW-PLAN provides a step forward to accomplish such integration for better and more efficient construction planning.
5.2.2.1. The KNOW-PLAN Model versus CAD-based Planning Systems

Visual simulation of the construction process on graphics display is a new approach to improve and enrich the planning process. Bechtel's work utilizes this new technological capability in implementing their computerized system. However, it doesn't use any knowledge-based system to automate the process of generating construction plans. They approach the problem by sorting different tagged objects in files. These files are later used to generate the sequence interactively by the user. The system is able to simulate a certain type of projects (such as power plants with limited number of object classes).

Stone & Webster's system uses a similar approach to perform sequencing. The system develops the sequence based on an interactive session with the planner (user). The system starts by defining linear sequence for different paths in the plan network. Then, defining dependencies between paths to come up with an entire network.

In the KNOW-PLAN model, the simulation of the construction process is implemented using as input: a sequence of activities generated by a KGB system rather than interactively by the user. The 3D computer model of the facility is processed to extract spatial relationships among the facility components. These relations are manipulated during the reasoning process by the inference engine of the system to come up with a feasible construction plan for a generic facility.

5.2.2.2. The KNOW-PLAN Model versus AI-based Planning Systems

Early AI planners, such as: STRIPS, ABSTRIPS, INTERPLAN, NOAH, NONLIN, O-PLAN, MOLOGEN, and DEVISER, provide conceptual tools to generate a plan.
However, they possess significant limitations as tools to be used in construction planning. They are domain-independent planners. This makes them insufficient to be applied directly to the knowledge-intensive construction planning domain. Most of these planners express the knowledge about the domain in the form of action, preconditions, and effects (constructive and destructive). This form of knowledge representation cannot express properly the nature of construction knowledge which is based on variable constraints (conditions to be imposed on the plan generation reasoning process).

[Hendrickson et al. 87] explains the limitations of these planners as systems for construction planning. Among these limitations are:

1. They generally incorporate only a relatively small number of well-defined, repetitive tasks. In contrast, construction requires numerous distinct tasks for completion.

2. Construction planning involves the selection of appropriate resources to apply. In contrast, these systems provide such resources.

3. Construction has numerous important planning concerns with respect to time constraints, cost and resource trade-offs, and spatial restrictions that are not explicitly considered by the AI planners,

4. The large size of construction planning problems suggest that efficient algorithmic scheduling tools may be required rather than only myopic, heuristic allocations. The computational burden of scheduling in construction is significant: construction schedules can include hundreds of activities.

5. Construction planning is highly knowledge-intensive, so the explicit use of expert knowledge is required in the planning process.
The other AI-based planning systems, implemented using knowledge-based or object-oriented approaches, provide tools to automate the process of generating construction plans. However, most of these systems don’t utilize the knowledge embedded in 3D computer models to minimize the user’s input and interaction. They require the user to provide project description, topology, and components with their spatial relationships. They lack the capability of acquiring such information with minimal user interaction.

Consequently, these systems limit their planning domain to a specific type of projects, such as off-shore platforms, mid-rise buildings, office-buildings. None of the systems approaches the problem by providing project’s components representation in a “generic” form. These systems still depend on presenting the generated plan in the conventional way of networks, bar charts, and textual reports. This type of presentation lacks the ability to provide visual tools for project plan presentation for better communication and better understanding of the nature of the plan.

The KNOW-PLAN model provides a step forward in automating the process of generating project plans. The differences between the KNOW-PLAN model and other systems are:

1. **KNOW-PLAN combines CAD, 3D computer modeling, and Artificial Intelligence capabilities to provide an integrated system for construction planning.**

2. **KNOW-PLAN extracts the rich knowledge available in 3D computer models.** This knowledge is exported to a KGB system in different forms to be manipulated directly or indirectly during the automated reasoning process. The knowledge available in 3D computer models can be classified as follows:
   - knowledge about the project description and configuration;
   - knowledge about the breakdown of the project components;
• knowledge for defining spatial relationships among various components of the project (geometric data);

• tools for constructability checking which is the source for constructability constraints to be used in the plan generation reasoning process.

3. KNOW-PLAN's ability to extract project components and their spatial relationships enables it to be a suitable tool for a "generic" project with minimum user interaction (required data).

4. KNOW-PLAN incorporates a visual simulation system as part of the overall model to present the generated plan to the user on graphics display, this is in addition to the conventional forms of presentations.

In summary, the KNOW-PLAN model's approach of generating project plans primarily based on knowledge extracted from 3D computer models and other knowledge sources, input by the user, forwards an additional step towards automating the construction planning process.

5.2.3. The KNOW-PLAN Model's Components

The overall KNOW-PLAN model is divided into six stages:

1. Generation of a 3D computer model;
2. Database and knowledge-base creation and manipulation;
3. Dynamic sequencing process;
4. Interactive sequence modification;
5. Conventional scheduling and reporting;
Figure 5.2 depicts a schematic presentation of the different stages of the model. Figure 5.3 depicts an overall picture of the model showing the interaction between the different stages of the model. Each stage is represented by a separate box. The following subsections further describe each component.

5.2.3.1. **Stage 1: Generation of a 3D Computer Model**

The outcome of the design phase which uses a CAD system is a 3D computer model of the designed facility. The designed 3D computer model will be transformed to a format readable by WALKTHRU, a 3D visual simulation system.

The transformed 3D computer model will be processed to extract the geometric data needed to define the spatial relationships among various objects of the 3D computer model. The extracted data will be stored in a central database for further processing. The manipulation and transformation process is based on the interface specifications of the WALKTHRU system. Figure 5.4 depicts the details of the 3D computer model generation stage.

5.2.3.2 **Stage 2: Database/Knowledge-Base Creation and Manipulation**

The central database, in which the extracted geometric data from Stage #1 is stored, consists of different database files (DBFs). The DBFs contain the necessary data needed by the KGB system for the plan generation reasoning process and the visual simulation process. The central database provides an interactive environment to simplify the user interaction with the KNOW-PLAN model. Most of the data and knowledge needed are entered to the system through this interface.
Figure 5.2: KNOW-PLAN Model’s Components
Figure 53: Overall Picture of the KNOW-PLAN Model
Figure 5.4: The 3D Computer Model Generation Stage
The central database contains files for different categories of objects. There are two main
types of objects. First, objects which define "entities" such as: component objects (ac-
tivities), class objects, zoning objects, connection type objects, and resource objects. The
second category is objects which define relations between entities, such as: activity con-
nection relations, class-to-class relations, activity-to-class relations, resource-to activity
relations. The Relational Data Modeling approach of information modeling is imple-
mented in developing the central database. Records in the central database files corre-
spond to schemata (frames) in the knowledge-base of the KGB system. Typical
information required to represent the different types of objects are:

1. Component Objects (Activities): The information related to each component of
   the facility is stored in an activity database file. This file includes data about:
   component geometric data, activity duration, activity number, activity de-
   scription, activity mandatory dates, and constraint procurement dates. The
   component geometric data is extracted from the 3D computer model of the
   facility. The activity duration can be entered to the system either by the user
   or by an interface to a simulation system for an automatic calculations.
   Mandatory dates of completion will be used to consider any time constraint
   which affect the installation of a component. Procurement dates will be used
   to consider any procurement constraints which will affect the installation or
   execution of a specific activity.

2. Class Objects (Group Listing): Each component(object) in the model will be
   attached to different classes from which the object will inherit some character-
   istics. The class (group) listing will be initialized by the system from a previ-
   ously defined list stored in the database. The group listing will be used to
   define the direction of installation for a specific group of objects. For example,
the objects related to a specific group will be executed from bottom to top such as concrete elements in a building structure. Other groups will be executed back to front or right to left, etc. This data can be defined by the designer during the creation of the 3D computer model based on a specific coding system.

3. **Zoning Objects**: the project will be divided into zones with defined geometric boundaries. The zones will have an important role in optimizing the automated plan generation reasoning process.

4. **Connection Type Objects and Activity Connection Relations**: Each object in the facility will be assigned attributes pointing to a connection type list which represents the means of connection of an object with other objects around it. The type of connection could be structurally supported, embedded in, protected by, ...etc. The connection type with the spatial relationships among various objects will be used by the KGB system to generate the project plan.

5. **Class-to-Class Relations**: This type of relation defines the direction of installation of an object of a specific class with other objects of different classes.

6. **Activity-to-Class Relations**: This type of relation defines the classes to which an object (component) is related. Such a relation will be used in the KGB system to inherit the direction of installation of different objects as they relate to different classes.

7. **Resource Objects**: The resource requirements for an activity represent constraints to be considered by the KGB system during the plan generation process.

It is important to note here, that most of the information listed above represents different types of knowledge needed to perform the reasoning process to generate a project.
plan by the KGB system. They are entered in the central database to simplify the data
and knowledge entry task instead of defining this knowledge directly within the KGB
system. Some of this knowledge will be stored in standard files to be used in different
projects. These files are updated continuously to reflect any additional knowledge ac-
ququisition performed by the user. The KNOW-PLAN model has the necessary external
interface to map the available information from the central database to the hierarchical
schemata structure of the knowledge-base of the KGB system.

Additional knowledge can be added to the knowledge-base to define rules needed to
reason with during the generation of the sequence. Such rules can define procurement
constraints, resource constraints, design constraints, organization constraints, and other
planning constraints. This additional knowledge can be either a system's defined
knowledge or a user's defined knowledge. Figure 5.5 depicts the flow of data in the da-
tabase and the knowledge-base creation and manipulation stage as described before.

Chapter 7 will describe in more detail the contents of the central database as it is im-
plemented in the KNOW-PLAN prototype system.

5.2.3.3 Stage 3: The Dynamic Sequencing Process

The geometric data acquired form the 3D model can be the basis for a possible sequence
of installation for the components (objects) of most structures. This sequence can be
generated by analyzing the objects' attributes stored in the central database and the ex-
tracted spatial relationships among various objects in the model. The attributes needed
for this purpose are the connection types between objects and the classes to which the
objects belong. By using these two attributes, a possible sequence can be generated by
Figure 5.5: Database & Knowledge-base Creation and Manipulation
the KGB system as a network of activities which shows the sequence of installation based mainly on the geometric data. This network is called "geometric-based network."

The model will allow the user to run the Path Finder routine of the WALKTHRU system [Morad and Cleveland 89]. This routine finds the optimum path of an object from one position and orientation in a 3D computer model to another position and orientation. The utilization of this routine will help the planner in verifying the constructability of critical objects. The outcome of the simulation process is a list of constraints which describe which objects should not be installed before a specific object or objects are installed. These constraints will be stored in the form of facts or rules in the knowledge-base of the KGB system. These rules will be invoked by the inference engine of the KGB system to generate a network called "constructability network."

Similar networks will be generated by the KGB system to define the installation sequence among various objects based on other constraints. Resource constraints network, mandatory dates network, and Procurement constraints network are some examples of these networks. All the rules defined by the user will be considered to generate a "user's constraints networks" which reflect the unique requirements for a specific project.

Each link in the generated networks will have a priority value which will be considered during the generation of the compiled final network. Links with higher priority will override the links of lower priority. Network links generated from rules defined by the user will have higher priority than network links generated from the geometric data. The priority feature will solve the conflict between the logic asserted in the different networks between two objects. The final step in the dynamic sequencing process is the generation of the final logic, taking into consideration all the rules affected by the previously gen-
erated networks. Figure 5.6 depicts the flow of information and the reasoning process phases in the dynamic sequencing process stage.

5.2.3.4. Stage 4: Interactive Sequence Modification

The next stage in the model is an optional stage where the user will be able to interactively modify the generated sequence. The new sequencing conditions will be input as new rules in the KGB system. In case these new rules conflict with the previously generated sequence, the KGB system will respond with an explanation of the conflict to the user. If the user is satisfied with the new changes, an "additional network" will be generated. The system will combine this new network with the already generated networks from stage #3 to come up with a modified plan. This plan reflects the new changes of the knowledge in the knowledge-base of the KGB system. Figure 5.7 depicts the flow of information and the reasoning process phases in the interactive sequence modification stage.

5.2.3.5. Stage 5: Conventional Scheduling and Reporting

In this stage, the model will generate schedule reports as they exist in the traditional scheduling techniques. The installation duration of the objects and the generated sequence will be input to a CPM processor to generate schedule reports, bar charts, logic network, and time-scaled network. Figure 5.8 depicts the conventional scheduling and reporting stage.
Figure 5.6: The Dynamic Sequencing Process Stage
Figure 5.7: The Interactive Sequence Modification Stage
Figure 5.8: Conventional Scheduling and Reporting Stage
5.2.3.6. Stage 6: Visual Simulation

The final stage in the model is to visually simulate the construction process based on the generated plan. The generated plan will be transformed to a data format readable by the "Play Back" routine of the WALKTHRU system. The transformed data file will be designed as a Record file which contains frames of positions in the 3D computer model plus display characteristics at different intervals of time. This will represent the sequencing process on a time scale, thus reflecting the execution steps of the construction plan. The "Play Back" routine concept is based on the same concept used in VCRs. The system interpolates between key frames parameter values of the Record file, then the system replays the saved sequence of the frames in real-time. The outcome of such a process is a visual animation of the construction process according to the generated plan using the 3D computer model of the designed facility. Figure 5.9 depicts the visual simulation stage.
Figure 5.9: The Visual Simulation Stage
6. The 3D Computer Modeling Module

6.1. Introduction

The interaction of CAD applications with knowledge-based planning systems has a great potential to improve the quality of the design and construction processes. The question of interface, as a key to data exchange and interaction between different computer applications, becomes more and more important in achieving such integration. The provision of CAD data for further use in the construction planning process is the concern of the interface effort to implement such integration. In order to communicate and exchange CAD data and files between diverse computer applications, standardized interfaces have been implemented by the industry [Rembold and Dillmann 84]. A look at the different levels of standardization in the CAD field was presented in Chapter 3.

In order to utilize CAD systems in the development of new knowledge and geometric based (KGB) planning systems, the KGB planning system should interact with CAD systems to use existing CAD product (object) definition data. Objects or components
of a 3D computer model of a facility represent the physical entities which are associated with the different activities or group of activities in construction plans (schedules). The interaction can be accomplished by developing a communication environment for file exchange. Which means that object definition data has to be transferred to a specific format readable by the KGB planning system.

The approach of the KNOW-PLAN prototype system is to acquire the spatial relationships among various components of the designed facility from its 3D computer model. The acquired data is used in the reasoning process to generate a project plan. The 3D computer model can be designed using a CAD application. Therefore, the first task of the KNOW-PLAN prototype system is to generate a 3D computer model for the project to be planned. Then, the system will acquire the needed data from this model to export it to a KGB system (the dynamic sequencer). The KNOW-PLAN prototype system utilizes the WALKTHRU system to provide the necessary data needed from 3D computer models.

This chapter describes the details of the first module of the KNOW-PLAN prototype system. The purpose of this module is to generate a 3D computer model, perform the interface with WALKTHRU, and execute the data extraction process as it is implemented in the KNOW-PLAN prototype system.
6.2. 3D Computer Model Generation

A 3D computer model of the designed facility should be generated using a CAD system which can interface with the WALKTHRU system. WALKTHRU can interface with many CAD systems using modeling interface programs (to be introduced later in this chapter). The CAD system should be able to generate three-dimensional computer models. The 3D computer model contains the geometric definitions of the project components. In the KNOW-PLAN prototype system, each component (object) in the 3D computer model is associated with one activity in the project plan. Future extensions will provide the capability to associate many objects with one activity or one object with many activities at different levels of plan abstraction with hierarchical object and activity structure definitions.

6.2.1. Sample Project using CADAM

CADAM (Computer Aided Design And Manufacturing) is an advanced CAD system. A research effort at Virginia Tech is developing an interface between CADAM and WALKTHRU. The CADAM to WALKTHRU interface has been designed to translate geometric data from a 3D CADAM drawing to the WALKTHRU system. The interface can be invoked from CADAM being run under VM/CMS on IBM mainframe computers. Refer to [Beliveau, et al. 90] for details regarding the interface process.

The KNOW-PLAN prototype system used CADAM on the IBM 5080 mainframe to generate a 3D computer model of a sample project. The generated model is transferred
to the WALKTHRU system using the current version of the mentioned research effort. Figure 6.1 shows an isometric view of the sample project model. The sample project represents a culvert structure which consists of different structural elements. This sample project is selected to present the KNOW-PLAN prototype system. The different components of the culvert represent objects of different classes. Table 6.1 lists the different components of the sample project with their associated object numbers. The numbers represent the object numbers which are used in the 3D computer model manipulation within the visual simulation system.

6.3. WALKTHRU Interface Modeling Programs

Several interface programs currently exists to transfer 3D computer models into WALKTHRU model files to be able for WALKTHRU to interact with the 3D computer model. The tool to be used depends on what CAD system was used to create the model, and on how the user wants to use WALKTHRU.

The basic steps needed to run WALKTHRU with the 3D computer model are:

1. Define the Objects: The user must decide how objects to be defined within WALKTHRU.
2. Run the translation interface: The user must convert the model into a WALKTHRU model file using one of four conversion programs available. During the conversion, the model will be divided into objects according to decision made in the first step.
Table 6.1: Sample Project Components (Objects) List

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>West Barrel</td>
</tr>
<tr>
<td>20</td>
<td>Center Barrel</td>
</tr>
<tr>
<td>18</td>
<td>East Barrel</td>
</tr>
<tr>
<td>21</td>
<td>West Transition Wall Side 1</td>
</tr>
<tr>
<td>22</td>
<td>West Transition Slab</td>
</tr>
<tr>
<td>1</td>
<td>West Transition Wall Side 2</td>
</tr>
<tr>
<td>2</td>
<td>East Transition Wall Side 2</td>
</tr>
<tr>
<td>3</td>
<td>East Transition Slab</td>
</tr>
<tr>
<td>7</td>
<td>East Transition Wall Side 1</td>
</tr>
<tr>
<td>6</td>
<td>West Wrapwall Slab</td>
</tr>
<tr>
<td>15</td>
<td>West Wrapwall Side 1</td>
</tr>
<tr>
<td>5</td>
<td>West Wrapwall Side 2</td>
</tr>
<tr>
<td>9</td>
<td>East Wrapwall Slab</td>
</tr>
<tr>
<td>4</td>
<td>East Wrapwall Side 1</td>
</tr>
<tr>
<td>8</td>
<td>East Wrapwall Side 2</td>
</tr>
<tr>
<td>13</td>
<td>West Headwall</td>
</tr>
<tr>
<td>14</td>
<td>West Headwall Support 1</td>
</tr>
<tr>
<td>23</td>
<td>West Headwall Support 2</td>
</tr>
<tr>
<td>12</td>
<td>West Headwall Support 3</td>
</tr>
<tr>
<td>17</td>
<td>East Headwall</td>
</tr>
<tr>
<td>11</td>
<td>East Headwall Support 1</td>
</tr>
<tr>
<td>10</td>
<td>East Headwall Support 2</td>
</tr>
<tr>
<td>16</td>
<td>East Headwall Support 3</td>
</tr>
</tbody>
</table>
3. Build an Object Name File (Optional): If the 3D computer model is a large or complex model with multiple identical objects which to be moved independently, an Object Name File should be built.

When a 3D computer model is converted to run with WALKTHRU, the model can be subdivided into a number of parts, referred to in WALKTHRU as "objects". An object is a part of the model that can be referred to independently from other objects in the model.

When a WALKTHRU conversion program is run, each CAD design file specified will become an object in the WALKTHRU model file. In this manner, a WALKTHRU model file can be constructed from several IGDS, 3DM, or IGES files. Alternatively, a single large IGDS, 3DM, or IGES file can be converted into multiple WALKTHRU objects by running the conversion program multiple times, repeatedly specifying the same input and output files, while using one or more options to limit the entities processed with each pass. In this manner, each run of the conversion program creates a single object in the WALKTHRU model file from the parts specified of the large IGDS, 3DM, or IGES file.

WALKTHRU has four different conversion programs to translate 3D computer models into WALKTHRU model files.

1. WALKIGDS: Converts Intergraph Design Files (IGDS files);
   WALK3DM: Converts Bechtel 3DM Intergraph Design Files (3DM files);
2. WALKIGES: Convert Initial Graphics Exchange Specification data files (IGES files);
3. WALKPRE: Converts WALKTHRU ASCII Files;
4. WALKGDS: Converts GDS Things files.

Refer to the WALKTHRU reference manual for more detail about these interface programs.

6.4. WALKTHRU Object Definitions

6.4.1. WALKTHRU on the SUN 4/260 Workstation

WALKTHRU has been implemented on the Silicon Graphics IRIS series workstations. The implementation utilizes the graphics software and hardware capabilities of these workstations. A current development effort is being initiated by Bechtel to develop a version of WALKTHRU to run on the SUN 4 workstations with Raster Technology graphics hardware and software.

The KNOW-PLAN prototype system utilizes the current uncompleted version of the WALKTHRU system on the sun workstation. Although the version is not complete, it includes all the functions needed by the KNOW-PLAN prototype system.

6.4.2. Object Structures

The object data structure contains all the information needed to define an object in the WALKTHRU system. All objects are defined as an array in the model file. Each element
in the array corresponds to an object in the model file. Therefore, if there is a total of 
50 objects (physical and virtual) in the model file, the object array will have 50 entities 
(0-49). Object data are extracted from the CAD 3D computer model by running a spe-
cific WALKTHRU interface program.

The object data structure in the WALKTHRU program, which is used in performing the 
different functions of the system, are: object number, object rotation matrix, cumulative 
rotations about each local axis, cumulative movement about each local axis, local range 
of object (volume boundary), number of physical objects assigned to the object, flag in-
dicating whether to draw linkage between objects, and name assigned to the object.

Each object is defined geometrically by a set of surface elements. The surface element 
structure is the structure that contains the data needed to describe and draw a surface. 
WALKTHRU defines an array for surface elements. One element in the array for each 
object in the WALKTHRU model file. These elements act as the beginning of a linked 
list of surface elements. Therefore, if there is a total of 50 objects in the model file, the 
surface array will have 50 entries (0-49). To draw the graphics associated with object 
number 8 from the WALKTHRU model file, the system will start with surf[7] and con-
tinue draw through the list of elements attached to surf[7].

6.4.3. Build an Object Name File

The objects defined in the conversion process are stored in the WALKTHRU model file, 
and are called "physical" objects because they physically exist in the file. If the model 
contains identical objects in different places, such as four beams in a specific floor, "vir-
ual" objects can be created based on a physical object, in order to save storage space in the WALKTHRU file.

Virtual objects are created by building Object Name files that define virtual objects. WALKTHRU creates virtual objects at run time by making copies of the physical objects to use in the model. The Object Name file also allows the user to give each physical or virtual object a name. Then, objects in the model can be referred to by name, instead of having to remember object numbers. The format of an Object Name file is:

```
  vobjnum  pobjnum  objname
  vobjnum  pobjnum  objname
     ...    ...    ...
```

Where vobjnum is the virtual object number, pobjnum is the physical object number, and objname is the name of the object. For the beam example, the file will look like this:

```
  1   1  beam #1
  2   1  beam #2
  3   1  beam #3
  4   1  beam #4
```

In the Object Name file, the virtual object numbers must be sequential, starting with 1, and there must be no gaps in the numbering sequence. The physical object numbers must exist in the WALKTHRU disk file, or an error will result. Object names are limited to 20 characters and must begin with a letter.

If the physical object number is preceded by a minus sign, such as -1, then WALKTHRU creates a virtual object that exists only as a point with its initial position at the origin of the physical object. There are no graphical data associated with points in the WALKTHRU model. These invisible objects are useful for defining objects rota-
tion points in conjunction with object hierarchies. Virtual objects concept plays an important role in the Path-Finder routine for constructability checking to be introduced in Chapter 7.

6.5. Objects’ Geometric Data

One of the main functions of the KNOW-PLAN prototype system is to acquire the geometric data of different components of designed facility from a 3D computer model. This acquired data will be transferred to the central database of the system. Then, it will be transferred to the knowledge-base of the dynamic sequencer for processing and manipulation to generate a project plan. The geometric data are extracted directly from the model file of the designed facility as it exists in the WALKTHRU environment.

The KNOW-PLAN prototype system extracts the needed data for the reasoning process by the dynamic sequencer and the visual simulation process by the visual simulator within WALKTHRU from two sources (files). The first source is a “Volume File” that defines the maximum and minimum values of the objects’ boundaries in the X, Y, and Z coordinates. The file is a simple ASCII file which lists the needed values for each object in the 3D computer model. This file will be created within the WALKTHRU system. This process is performed manually in the KNOW-PLAN prototype system. The WALKTHRU system will provide an internal user function to create this file automatically within the system.
Another alternative for getting the geometric data is during the model conversion process using a "volume definition file" which relates to the model file. The volume definition file defines a cube in model space that is used as a filter in the WALKTHRU conversion programs. The cube is defined by two points in model space. The first point is the minimum x, y, z coordinate. The second point is the maximum x, y, z coordinate. In the volume definition file, the cubes are assigned volume names. The names are any text string without spaces. The format of the volume definition file is:

```
volume_name_1
x_min_coordinate  y_min_coordinate  z_min_coordinate
x_max_coordinate  y_max_coordinate  z_max_coordinate
volume_name_2
x_min_coordinate  y_min Coordinate  z_min Coordinate
x_max_coordinate  y_max_coordinate  z_max_coordinate
......
volume_name_n
......
```

Each volume corresponds to an object in the 3D computer model. The boundary of each object are extracted using the coordinate values defined in the volume definition file.

The second source for the data to be extracted is an initial record file. Record files in WALKTHRU contain a sequence of one or more key frames. Each key frame contains view configuration and optional object orientation data about the model at a particular moment in time. The initial record file is a file which contains only one key frame for which all the objects of the model are displayed on. The file will contain the coordinates of the center of the object and the rotation of the object in the x, y, and z direction. This
data are currently extracted manually from the initial Record file. Chapter 9 will discuss
the Record file structure in more details.

The geometric data extracted from the initial record file and the boundary data extracted
from the volume definition file of the model are transferred to the central database for
later processing. Table 6.2 lists the geometric data of the different objects of the sample
project as they are defined by WALKTHRU model file with their associated activity
number.

Figure 6.2 summarizes this chapter by depicting the data flow in the 3D computer
modeling module. The figure illustrates the possible methods of applications' interface
with the WALKTHRU system. The different data exchange files needed to communi-
cate with the central database of the KNOW-PLAN prototype system are illustrated in
the figure.
### Table 6.2: Objects' Geometric Data (Sample Culvert Project)

<table>
<thead>
<tr>
<th>Activity No.</th>
<th>OBJ No.</th>
<th>MINIMUM X</th>
<th>Y</th>
<th>Z</th>
<th>MAXIMUM X</th>
<th>Y</th>
<th>Z</th>
<th>CENTER X</th>
<th>Y</th>
<th>Z</th>
<th>ROTATION X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_1</td>
<td>19</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>55.000</td>
<td>28.500</td>
<td>16.157</td>
<td>27.500</td>
<td>14.250</td>
<td>8.083</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_2</td>
<td>20</td>
<td>55.000</td>
<td>0.000</td>
<td>0.000</td>
<td>110.000</td>
<td>28.500</td>
<td>16.157</td>
<td>82.500</td>
<td>14.250</td>
<td>8.083</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_3</td>
<td>18</td>
<td>110.000</td>
<td>0.000</td>
<td>0.000</td>
<td>165.000</td>
<td>28.500</td>
<td>16.157</td>
<td>137.500</td>
<td>14.250</td>
<td>8.083</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_4</td>
<td>21</td>
<td>-40.000</td>
<td>22.250</td>
<td>0.000</td>
<td>-39.000</td>
<td>51.300</td>
<td>20.700</td>
<td>-39.500</td>
<td>36.775</td>
<td>10.350</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_5</td>
<td>22</td>
<td>-40.000</td>
<td>5.099</td>
<td>0.000</td>
<td>-39.000</td>
<td>23.461</td>
<td>4.000</td>
<td>-39.500</td>
<td>14.250</td>
<td>2.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_6</td>
<td>1</td>
<td>-40.000</td>
<td>-22.800</td>
<td>0.000</td>
<td>-39.000</td>
<td>6.250</td>
<td>20.700</td>
<td>-39.500</td>
<td>-8.275</td>
<td>10.350</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_7</td>
<td>2</td>
<td>204.000</td>
<td>22.250</td>
<td>0.000</td>
<td>205.000</td>
<td>51.300</td>
<td>20.700</td>
<td>204.500</td>
<td>36.775</td>
<td>10.350</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_8</td>
<td>3</td>
<td>204.000</td>
<td>5.099</td>
<td>0.000</td>
<td>205.000</td>
<td>23.461</td>
<td>4.000</td>
<td>204.500</td>
<td>14.250</td>
<td>2.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_9</td>
<td>7</td>
<td>204.000</td>
<td>-22.800</td>
<td>0.000</td>
<td>205.000</td>
<td>6.250</td>
<td>20.700</td>
<td>204.500</td>
<td>-8.275</td>
<td>10.350</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_10</td>
<td>6</td>
<td>-39.000</td>
<td>1.249</td>
<td>1.667</td>
<td>0.000</td>
<td>27.251</td>
<td>4.000</td>
<td>-19.500</td>
<td>14.250</td>
<td>2.833</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_11</td>
<td>15</td>
<td>-39.000</td>
<td>21.250</td>
<td>1.667</td>
<td>0.000</td>
<td>48.300</td>
<td>28.467</td>
<td>-19.500</td>
<td>34.775</td>
<td>15.067</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_12</td>
<td>5</td>
<td>-39.000</td>
<td>-19.800</td>
<td>1.667</td>
<td>0.000</td>
<td>7.250</td>
<td>28.467</td>
<td>-19.500</td>
<td>-6.275</td>
<td>15.067</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_13</td>
<td>9</td>
<td>165.000</td>
<td>1.249</td>
<td>1.667</td>
<td>204.000</td>
<td>27.251</td>
<td>4.000</td>
<td>184.500</td>
<td>14.250</td>
<td>2.833</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_14</td>
<td>4</td>
<td>165.000</td>
<td>21.250</td>
<td>1.667</td>
<td>204.000</td>
<td>48.300</td>
<td>28.467</td>
<td>184.500</td>
<td>34.775</td>
<td>15.067</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_15</td>
<td>8</td>
<td>165.000</td>
<td>-19.800</td>
<td>1.667</td>
<td>204.000</td>
<td>7.250</td>
<td>28.467</td>
<td>184.500</td>
<td>-6.275</td>
<td>15.067</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_16</td>
<td>13</td>
<td>0.000</td>
<td>0.000</td>
<td>16.167</td>
<td>1.000</td>
<td>28.500</td>
<td>28.467</td>
<td>0.500</td>
<td>14.250</td>
<td>22.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_17</td>
<td>14</td>
<td>1.000</td>
<td>27.250</td>
<td>16.167</td>
<td>9.000</td>
<td>28.500</td>
<td>28.467</td>
<td>5.000</td>
<td>27.875</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_18</td>
<td>23</td>
<td>1.000</td>
<td>13.625</td>
<td>16.167</td>
<td>9.000</td>
<td>14.875</td>
<td>26.467</td>
<td>5.000</td>
<td>14.250</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_19</td>
<td>12</td>
<td>1.000</td>
<td>0.000</td>
<td>16.167</td>
<td>9.000</td>
<td>1.250</td>
<td>26.467</td>
<td>5.000</td>
<td>0.625</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_20</td>
<td>17</td>
<td>164.000</td>
<td>0.000</td>
<td>16.167</td>
<td>165.000</td>
<td>28.500</td>
<td>28.467</td>
<td>164.500</td>
<td>14.250</td>
<td>22.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_21</td>
<td>11</td>
<td>156.000</td>
<td>27.250</td>
<td>16.167</td>
<td>164.000</td>
<td>28.500</td>
<td>26.467</td>
<td>160.000</td>
<td>27.875</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_22</td>
<td>10</td>
<td>156.000</td>
<td>13.625</td>
<td>16.167</td>
<td>164.000</td>
<td>14.875</td>
<td>26.467</td>
<td>160.000</td>
<td>14.250</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ACT_23</td>
<td>16</td>
<td>156.000</td>
<td>0.000</td>
<td>16.167</td>
<td>164.000</td>
<td>1.250</td>
<td>26.467</td>
<td>160.000</td>
<td>0.625</td>
<td>21.317</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Figure 6.2: Data Flow in the 3D Computer Modeling Module
7. The Knowledge and Data Module

7.1. Introduction

Knowledge-Based Planning Systems involve the use of knowledge-based systems to automatically generate construction plans. Construction planning is a knowledge-intensive domain. Construction planners' knowledge should be captured and stored in the knowledge-base of these systems so that the computer can perform the planning tasks usually performed by the planners. Knowledge about the construction planning domain from different sources is required to be captured to accomplish this task.

Typically, construction knowledge includes: spatial relationships among various components (objects) of the project, constructability constraints, resource requirements, cost constraints, weather-constraints, mandatory dates constraints, and others. However, the main source of knowledge used in construction planning is the knowledge about the spatial relationships among various objects. This knowledge can be used by
knowledge-based systems to assert possible sequence of installation among different objects based on a geometric reasoning process.

The KNOW-PLAN prototype system has the capability of defining spatial relationships among various objects based on geometric data extracted from a 3D computer model of the project. In order to accomplish the task of transforming the needed geometric data from the 3D computer model to the dynamic sequencer, a mean of communication between the CAD system and the KGB system is required. This communication can be implemented by providing an integration tool to integrate both systems.

Systems are integrated if they have elements in common. Data is the most important element to keep consistent among systems. When many applications are to be integrated, an efficient data exchange and interface should be developed. In the KNOW-PLAN prototype system, a central database management system is implemented to provide such integration. The central database is designed to provide the interface between the 3D computer modeling system and the dynamic sequencer (a KGB planning system). In addition, the central database provides the user with the tools to manipulate the stored data before the interface process is performed.

This chapter introduces the data required by the dynamic sequencer to perform the geometric reasoning process. The geometric reasoning process is concerned with asserting possible sequence of installation of different objects based on their geometric data and the relation between the classes to which the objects are related. The data modeling and the implementation of the central database is also discussed in this chapter. Finally, other sources of knowledge incorporated in the KNOW-PLAN prototype system are presented.

7. The Knowledge and Data Module
7.2. KNOW-PLAN's Knowledge and Data

Knowledge-based and expert systems are computer programs that use logical relationships to embody knowledge about a specific domain and perform specialized tasks that typically require human judgement and expertise [Rolston 88]. The primary source of expertise is knowledge. Primarily, the experts' problem solving skill arises from their knowledge about the task at hand. The power of a knowledge-based system doesn't arise from speed of computing nor the ability to retain endless details; but rather a knowledge-based system's power resident in the extensive knowledge stored.

Knowledge is not synonymous with information. Rather, knowledge is information that has been implemented, categorized, applied, and revised. The refined information can exist in many forms. The knowledge can be exemplified by concepts, constraints, and heuristic methods for using data and principles that govern domain specific operations. Knowledge consists of (1) symbolic description of definitions, (2) symbolic descriptions of relationships, and (3) procedures to manipulate both types of descriptions [Feigenbaum and Mccorduck 83, Waterman and Peterson 81, and McGraw and Harbison-Briggs 89].

Knowledge can be presented as declarative knowledge or procedural knowledge. Declarative knowledge comprises statements that relate some element of truth regarding the subject domain. Procedural knowledge are well defined, invariant rules that describe fundamental sequence of events and relations relative to the domain [Rolston 88].

Construction planning is a knowledge-intensive domain. A large amount of knowledge is necessary to solve the complicated planning problem. Knowledge-based planning
systems are able to generate construction plans if a sufficient amount of knowledge is captured and stored. The KNOW-PLAN prototype system is a knowledge-based planning system that contains knowledge about generating construction plans. However, the systems primary advantage is its ability to reason with the geometric data of the project components. The KNOW-PLAN prototype system also reasons with other kinds of knowledge, if exist, to generate construction plans.

The system incorporates two types of knowledge representation: declarative and procedural knowledge. Declarative knowledge describes the different elements of refined data needed by the KGB system to perform its task. The procedural knowledge consists of rules that perform the actual reasoning process.

In this chapter, the different data needed to define the KNOW-PLAN's declarative knowledge is presented. Other sources of knowledge that are incorporated into the system is also presented in this chapter. The procedural knowledge of the KNOW-PLAN prototype system is presented in Chapter 8.

7.2.1. Activity Data

Construction planning involves breaking down construction projects into different tasks. Each task is usually defined as one activity or a set of activities depending on the desired level of abstraction of the construction plan. Hence, the activities are considered as the basic entities of the construction plan. The primary objective of the KNOW-PLAN prototype system is to plan the sequence of executing the project's activities. The sequence is defined by asserting precedence relationships among different activities of the plan. In the KNOW-PLAN prototype system, the activity list of a project is defined
according to the break down of the project into components (objects) as defined by a 3D computer model of the project. Each object in the 3D computer model is associated with one activity in the construction plan.

In the KNOW-PLAN prototype system, each activity is associated with different attributes that describe the activity. The description of the activity consists of three groups of attributes. The first group consists of attributes that describe the activity name, and number. The second group describes the geometry of the physical object related to the activity. The geometric data include: the boundaries of the object, the coordinates of the center of the object, and the rotations of the object. The geometric data are used by the dynamic sequencer reasoning process to generate a possible execution sequence among different activities of the construction plan. The third group is the schedule attributes. The schedule attributes include: activity duration and the early and late start and finish dates of the activity. The activity duration is input by the user. The early and late start and finish dates of the activity are calculated by the dynamic sequencer of the system based on the logic (sequence) generated by the reasoning process.

A future version of the system is expected to define different levels of plan abstraction. Activities will be broken down into subactivities at different levels. Each object in the 3D computer model will be allowed to be associated with more than one activity in the construction plan. Also, an activity can be presented by more than one object in the 3D computer model. The duration of the activities can be generated by a knowledge-based system based on quantity take-off, production rates, and availability of resources.
7.2.2. Class Data and Class Relationships

A class is an assignment of an object which is related in behavior with similar objects. The attributes of a class entity are the basic elements of geometric knowledge in the KNOW-PLAN prototype system. This knowledge is used by the system to relate activities to one another, based on their spatial relationships, in order to assert precedence relationships among the different activities. A class can define: trade, design elements, group of activities that need similar resources, or group of activities that are constrained by a specific requirements. It is up to the user to define the classification of different classes for a specific project. However, it is recommended to have general classes that defines the different trades according to the Master Format coding system. According to this directive, classes can be defined as: site work, concrete, blockwork, masonry, electrical, mechanical, etc. Classes can be broken down into different subclasses such as: beams, columns, slabs, walls, etc... for the concrete class. Different levels of classes can define the hierarchical structure of the classes.

A future extension of the KNOW-PLAN system should be extended by providing generic files that define class hierarchy for different types of projects. The user will be allowed to load as many files as necessary to the central database to be used in the plan generation process by the dynamic sequencer. The central database of the KNOW-PLAN prototype system has the facility to read as many files as the user desires and to pass the selected class data to the dynamic sequencer. The system will incorporate different data optimizers which are knowledge-based systems to be used in optimizing the data defined in the class data and relationships database files.
The KNOW-PLAN prototype system provides many attributes to define the knowledge about a specific class. A typical class has the following attributes: class name, direction of installation, and priority of installation. The class name is a symbol that represents the class. The direction of installation represents the acquired knowledge about the sequencing of the activities related to this class in the X, Y, and Z directions. The value of these attributes can be Yes or No. An example is as follows: If the value is Yes in the X direction for class C1, it means that "if ACT(1) and ACT(2) are of class C1 and the object of ACT(1) is lower in the X direction than the object of ACT(2) then, install ACT(1) before ACT(2)". If the value is No in the X direction, then install ACT(2) before ACT(1). If the value is Yes in the Y direction, then install the lower object in the Y direction before the higher object. If the value is Yes in the Z direction, then install the lower object in the Z direction before the higher object.

The priority of installation has two variables: first priority and second priority. The first priority defines which direction of installation is dominant. The second priority defines the second level of domination. If the first priority is Y and the second priority is Z, the dynamic sequencer will sequence the objects based on the direction of installation in the Y direction. If both objects have the same maximum and minimum values in the Y direction, the system will sequence the objects based on the direction of installation in the Z direction. If the two objects still have the same maximum and minimum in the Z direction, the direction of installation in the X direction will be used to define the sequence.

Figure 7.1 depicts an example of the installation sequence of ACT(1) and ACT(2) which are related to class C1. Three cases are presented to further explain the concept of sequencing activities of the same class based on the direction of installation and installa-
tion priority knowledge stored in the system. The first case in the figure is explained below.

**CASE#1:** The first priority for class C1 is in the X direction. According to the direction of installation in the X direction, the lower object in the X direction will be sequenced before the higher object. Since the two objects are at the same level in the X direction, the second priority of class C1 should be used. The second priority is in the Z direction. According to the direction of installation in the Z direction, the higher object will be sequenced before the lower object. Accordingly, ACT(2) should precede ACT(1) based on the geometric data of the two objects and the geometric knowledge of class C1.

The above discussion is only valid if the two objects to be sequenced are of the same class. In the case of different classes, class relationships data are available in the system to assert the sequence between activities of different classes. Two types of relations between different classes are defined in the KNOW-PLAN prototype system: dominating class relations and special class relations.

Dominating class relations define which class among each two classes is dominating with a specific type of connection. If class C1 is dominating class C2 in the case of (t1) type of connection, the object related to class C1 will be sequenced before the object related to class C2. This is valid if the two objects are connected with a t1 type of connection. Figure 7.2 (a) depicts the dominating class relationships between different classes based on the type of connection between the objects that are related to these classes. For example, class C1 dominates class C2 if the connection type is t2. However, class C2 dominates class C1 if the connection type is t1.
Figure 7.1: The Class Geometric Knowledge
(a) Dominating Class Relationships

(b) Special Class Relationships

(c) Activity Connection Relationships

Figure 7.2: Similar Entity Relationships (Involute)
Example:

If ACT(1) is of class 'beam'
and ACT(2) is of class 'slab'
and ACT(1) and ACT(2) are connected with "structurally dependent" connection
and class 'beam' dominating class 'slab' with this type of connection.
then:
    install ACT(1) of class 'beam' before ACT(2) of class 'slab'.

The second type of relationships between different classes is the special class relationships. These relationships specify the situations in which a dominating class relationship is not applicable. A typical special class relationship may look as follows:

    class #1 is C1
    class #2 is C2
    higher in X is C1
    higher in Y is C2
    higher in Z is C1
    type of connection is t1
    install C2

According to the above special class relation, if class C1 is dominating class C2 with t1 type of connection, then an object ACT(2) of class C2 will be sequenced before the object ACT(1) of class C1 if the following conditions are satisfied: "ACT(1) is higher in the X direction than ACT(2), ACT(2) is higher in the Y direction than ACT(1), and ACT(1) is higher in the Z direction than ACT(2)". Accordingly, the domination of objects of class C1 is not applicable under the geometric conditions specified by the special class relation. Any two classes can have more than one special class relation. Each special
relation is based on the geometric conditions which make the dominating class relation not applicable.

Figure 7.2(b) depicts the special class relationships between different classes. Note that, the sequence of installation is reversed as opposed to Figure 7.2(a). Figure 7.3 depicts an example of the installation sequence of ACT(1) and ACT(2) which are related to the classes C1 and C2, respectively. Three cases are presented in the figure to explain the concept of sequencing activities of different classes based on the dominating and special class relations knowledge between class C1 and class C2. The first case in the figure is explained below.

**CASE#1:** Install ACT(2) before ACT(1) according to the connection type case B, where C2 is dominating C1. Since the geometric conditions between ACT(1) and ACT(2) don’t match a special class relationship, the dominating class relationship is applied.

### 7.2.3. Activity Class Relations

In the KNOW-PLAN prototype system, an activity represents an object within the 3D computer model. Each activity has to be related to different classes in order to utilize the sequencing knowledge defined about different classes in the geometric reasoning process. Relating activities to classes allows the activities to inherit the direction of installation data from the related classes. Two activities of different classes can inherit the dominating and special class relations knowledge of their related classes. The inherited knowledge is used in the geometric reasoning process to come up with a sequence of
Figure 7.3: The Class Relations knowledge
installation. The relationships between classes and activities can be defined as many-to-
many type of relationships. An activity can be related to many classes. Also, each class
could have several activities.

Figure 7.4 provides some examples of sample relationships between classes and activi-
ties. Class C1 has two activities: ACT(1) and ACT(3). Class C2 has only ACT(1). Class
C3 has ACT(3) and ACT(4). Class C4 has ACT(5) and ACT(6). Class 'C5' has only
ACT(5). On the other hand, ACT(1), ACT(2), ACT(4), and ACT(6) are related only to
one class: C1, C2, C3, and C4, respectively. ACT(3) is related to the classes C1 and C3.
ACT(5) is related to the classes C4 and C5.

7.2.4. Activity Connection Relationships

In construction projects, physical objects are related by the type of connection that exist
among them. An object can be related to another object by different means of con-
nections. Typical connection types are: structurally supported, embedded in, protected
by, ...etc. [Gray 86] provides a framework for the definition of the possible type of
connection between various construction objects.

In the KNOW-PLAN prototype system, the geometric reasoning process asserts the se-
quence between two objects based on the relationship that exist between the two classes
to which the two objects are related. If the two objects belong to the same class, the
direction of installation of the class is used to assert the sequence. In this case, no type
of connection is needed to be defined between the two objects.
CLASSES

ACT  (1)
ACT  (2)
ACT  (3)
ACT  (4)
ACT  (5)
ACT  (6)

ZONES

Z1  Z2  Z3  Z4  Z5

Figure 7.4: Different Entity Types' Relationships
On the other hand, the connection type data is needed in the case where the two objects are of different classes. The geometric reasoning process uses the defined connection type between the two objects to assert the sequence of installation between the two objects in conjunction with the attributes of different class relationships between the two classes. Section 7.2.2. describes the basis for asserting the sequence of installation between two objects of two different classes using the type of connection between the two objects.

Typically, objects of different classes always have the same type of connection. For example, a concrete column of class 'column' always has a connection type of "structurally dependent" with a concrete slab of class 'slab'. In the KNOW-PLAN prototype system, the symbol 'tg' (Type of connection: General) is used to represent the default type of connection between two objects of different classes whatever the type of connection is. Using the 'tg' type of connection as a default will minimize the amount of data required to be entered by the user. Therefore, the user is required to enter only the special cases of the type of connection between the two objects to count for the cases which are not satisfied by the default value.

Future work (outside the scope of this dissertation) would provide a version of the system that will allow the definition of type of connection between objects automatically by utilizing an embedded knowledge-based system within the 3D computer modeling system. This will be accomplished by simulating the differential movement of each object and identifying the type of connection. The identification of the type of connection is based on the result of the interference checking during the simulation process.

Figure 7.2(c) depicts a sample connection relationships among different activities. ACT(1) is connected with ACT(2) by a 't1' type of connection. ACT(2) is connected
with ACT(3) by a ‘t1’ type of connection. ACT(2) is also connected with ACT(4) by a ‘t2’ type of connection. ACT(3) is connected with ACT(4) and ACT(5) by ‘tg’ the default value of the connection type between their classes. ACT(5) is connected with ACT(6) by a ‘t2’ type of connection.

7.2.5. Zoning Data

Construction projects are divided into zones. Zones are volume boundaries that define specific volumes of the work space. A zone consists of group of physical objects which are located fully or partially within the zone. In a building project, zones can define the different floors of the building. Or it can define the different sectors of a specific floor. In highway projects, a zone can define a specific segment of the highway. In industrial plants, a zone can define a specific sector of the plant. In housing projects, a zone can define a group of housing units.

Construction planners usually divide the project to different zones. The concept of dividing a project into zones is usually used to concentrate closely on the sequencing of activities in the zone as a separate subnetwork. The planner defines interdependancies between the different subnetworks, so relations (sequence) between the activities of different subnetworks can be defined. The subnetworks and the interdependancy among them are usually combined to come up with the final plan.

In the KNOW-PLAN prototype system, the concept of dividing the project into different zones is incorporated. The system allows the user to define the zoning structure for the specific project. A zone is defined by asserting the limits (boundaries) of the zone volume space. This approach allows the search space for a solution to be minimized. The
result is a reduction in the time spent by the system during the reasoning process. This is due to the reduction in the number of cycles (rules fired) needed to relate a specific object with other objects in the model. An object geometric data is only compared with other objects’ geometric data which are located in the same zone. An object can be related to objects in other zones if the object shares a common surface with objects in other zones or overlaps into other zones.

An object can be located in many zones in the KNOW-PLAN prototype system. Also, a zone can contain more than one object. Therefore, the relationship between zoning entities and activity entities is a many-to-many type relationship. The interdependency between subnetworks of different zones is based on the presence of an object in different zones. Also, it is based on the ability of an object to be related to objects in other zones which share a common surface with the object. Figure 7.4 depicts an example of the relationship between activities and zones. ACT(1) is located in the zones Z1 and Z3. ACT(2) is located in the zones Z3 and Z4. ACT(3) is located only in zone Z2. ACT(4) is located in the zones Z1 and Z3. ACT(5) is located in the zones Z2 and Z4. ACT(6) is located in the zones Z4 and Z5.

7.3. Data Modeling

Database management system (DBMS) is a specialized software that allows the definition of database structure, and the manipulation of data organized according to a logical structure [Bonczek et al. 84]. A DBMS is capable of supporting and managing any number of independent databases. Databases are the basic component of a DBMS.
A database is a computerized collection of records of many different types, organized according to a single integrated logical structure that allows redundancy to be eliminated (or at least controlled to a significant degree) [Bonczek et al. 84].

The data in a DBMS is usually organized according to some data model. Data models are central to database system. They provide a conceptual basis for designing a formal basis for the definition of the items of data and their interrelationships [Hughes 88]. The purpose of data modeling is to develop a global design for the database with the ultimate objective of achieving an efficient implementation which satisfies the defined system requirements. Data models represents the underlying structure of a DBMS. Therefore, the quality of a DBMS is significantly influenced by the quality of the underlying data model.

There are basically three types of data models in DBMS technology: hierarchical data model, network data model, and relational data model. During the past decade, the relational data model has replaced the other two data models as the dominant influence in database management systems. In the implementation of the central database of the KNOW-PLAN prototype system, the relational data model is selected as the appropriate modeling structure for the system’s database.

The first step in the database requirement analysis is to identify the data objects “entities” to be modeled, the properties of those entities’ “attributes”, and the association among those entities’ “relationships”. Entity-Relationship models are commonly used to present the underlying structure of data models.
7.3.1. Entity-Relationship Model

The Entity-Relationship (E-R) model is an informal model which provides a natural means for representing entities and thus relationships among these entities. In an E-R model, information is represented by means of three primitive concepts: entities, attributes, and relationships.

**ENTITIES:** correspond to the objects being modeled. For a database application, an entity is a thing about which a descriptive information is to be stored, which is capable of independent existence and can be uniquely identified. Entities are the fundamental components of data models. Entities are typically grouped under different classes called "entity types". An entity type represents objects with the same data structure such as "activities" in a construction plan.

An "instance" of an entity type is an occurrence of that type for which values of the attributes have been specified such as the activity "central barrel erection". Each entity type defined will have its own data structure diagram. The data structure diagram records all the data that needs to be known about an instance of an entity type.

**ATTRIBUTES:** represent the properties (characteristics) of the entities (objects) being modeled. An entity can be meaningfully described by its attributes. The value of an attribute in a specific entity is the specific data that describes a specific characteristic of the entity.
RELATIONSHIPS: represent association among entities. They represent the mapping or linkage between two sets of entity types. They are the results of interaction between the participating entity types, or logical association between the entity types. The functionality of a relationship may be a one-to-one, one-to-many, or many-to-many relationship. It is called a simple binary relationship if the relationship relates an instance of an entity type with an instance of another entity type. The relationship can be a complex relationship among entities of the same type "involuted relationship", or relationship involving more than two entity types.

Entity-relationship models are presented schematically by diagrams which depict the natural structure of the data. In these diagrams, rectangles represent entity types and diamonds represent relationships. Relationships are linked to other constituent entity type by arcs, and the degree of the relationships (one or many) is indicated on the arc. An entity-relationship schematic diagram for the KNOW-PLAN central database data model is shown in Figure 7.5 The figure is explained in the following subsection.

7.3.2. KNOW-PLAN Entity-Relationship Model

The KNOW-PLAN prototype system's entity-relationship model consists of three different entity types. The three entity types are: activity entity type, class entity type, and zone entity type. The activity entity type represents the actual objects (components) of the 3D computer model of the project. Each activity entity resembles one object in the 3D computer model. The attributes of an activity includes: object geometric data, activity schedule data, and activity description data. The class entity type represents the classes (trades) to which activities belong. The attributes of a class include: class de-
scription data, data needed to define the direction of installation for the activities that belong to this class. The zone entity type represents the geometric data of the different zones of the project as defined by the user.

The different entities and entity types are related according to five types of relationships. There are simple relationships and complex relationships. There are two simple relationships in the KNOW-PLAN’s E-R model. The first simple relationship is between activity entities and class entities. The functionality of this relationship is many-to-many (M:N). That means, an activity can be related to many classes and at the same time a class can have many activities. The second simple relationship is between activity entities and zone entities. The functionality of this relationship is many-to-many (M:N). Each activity can be located in different zones, and each zone can contain more than one activity. The activity-zone relationships are asserted by the KGB system. This is accomplished by comparing the activity geometric data and the zone geometric data.

The complex relationships in the KNOW-PLAN’s E-R model are involuted relationships with a functionality of many-to-many (M:N). There are three involuted relationships in the model. The first involuted relationship is between activities. This relationship represents the connection data between activities of different classes. The second involuted relationship is between classes. This relationship defines the dominating class between each two classes. The third relationship is also between classes. This relationship defines the special cases under which the dominating class relationship is not valid.

The attributes of the different entity types and relationships are described in subsection 7.4.2. Figure 7.5. depicts the entity-relationship model for the KNOW-PLAN prototype system’s central database. In the figure, entity types are represented by rectangles, re-
lationships are represented by diamonds, and the functionality of the relationships is represented by arcs.

7.4. **KNOW-PLAN Central Database Implementation**

When many applications need to be integrated (linked), an efficient data interface environment should be developed. In the KNOW-PLAN prototype system, geometric data from 3D computer model needs to be passed to the dynamic sequencer. This data will be used for reasoning with other construction knowledge in the process of generating a construction plan. In order to visually simulate the construction process, the sequence of the activities should be passed to the 3D computer modeling system.

The dynamic sequencer needs knowledge about: the classes to which activities are related, the relationship between classes, and the connection between different activities. All this knowledge is required to be entered to the system, either directly by the user, or to be read from files of previous projects.

A central database management system is developed as part of the KNOW-PLAN prototype system to perform the above mentioned tasks of data exchange, knowledge definition, and data definition and manipulation. All the data and knowledge required by the dynamic sequencer are entered to the system via a menu driven central database management system. The central database has the functionality of allowing the user to manipulate all the data and knowledge stored in the database files. Also, it has the ability
Figure 7.5: E-R Model for the KNOW-PLAN Central Database
to read and write ASCII files. ASCII files are needed in the data exchange process between different applications integrated by the KNOW-PLAN prototype system.

Accordingly, the central database provides the communication means between the user and the different applications utilized by the KNOW-PLAN prototype system. This section discusses the actual implementation of the central database as a relational data model according to the E-R model described previously. The database files structure and the user interaction with the system is also introduced.

7.4.1. Relational Database

The data model of a relational database is based on the mathematical notion of relations. In this model, both entity types and their relationships are represented by two dimensional tables. The relational data model has a sound foundation, based on the mathematical theory of relations and the first-order-logic. It presents the user with a simple view of the data in the form of two-dimensional tables.

It is convenient to represent a relation as a table. The columns of the tabular relation represents attributes with distinct names. Entity type is represented by a relation scheme in which the attributes of the entity type become attributes of the relation. Each record (row) of the relation represents an instance of the entity type. The same can be applied to the entity relationships. Each relationship is represented by a relation scheme in which attributes of the entity relationship become attribute of the relation. Each record of the relation represents an instance of the entitys' relationship. In database management systems, a typical relation is stored physically in a file called "database file".
7.4.2. The Central Database File Structure

The central database of the KNOW-PLAN prototype system is implemented using a relational data model approach. Each entity type and entity relationship is defined by a relation table as a database file (DBF). The central database is implemented using the Clipper 87 compiler which has its own data definition and manipulation language.

The central database controls the definition and processing of the data needed to perform the integration of CAD data and the knowledge-based system. The data in the central database flows in two directions. First, the flow of data from the 3D computer modeling system into the knowledge-based system. This direction of flow is called the "preprocessor phase". Second, the flow of data from the knowledge-based system to the 3D computer modeling system through the central database. This direction of flow is called the "postprocessor phase". The preprocessor phase is discussed in this chapter. The postprocessor phase is discussed in Chapter 9.

The central database consists of seven DBFs which are needed to perform the flow of data from the 3D computer modeling system into the knowledge-base system in the preprocessor phase. The seven DBFs are divided into two groups. The first group includes the DBFs needed to define the different entity types needed by the system. The DBFs which are related to this group are: activity DBF, class DBF, and zoning DBF. The second group includes the DBFs needed to define the relationships between entities. The DBFs which are related to this group are: activity connection DBF, activity class relationships DBF, dominating class DBF, and special class relationships DBF. These files define the contents of the data and knowledge introduced at the beginning of this.
chapter. Other DBFs included in the central database are introduced in Chapter 9. The attributes of the relation tables (DBFs) are listed below.

**Activity DBF Attributes:** activity name, activity number, activity duration, minimum X, maximum X, minimum Y, maximum Y, minimum Z, maximum Z, center X, center Y, center Z, rotation X, rotation Y, rotation Z, early and late start dates, and early and late finish dates. All the values of these attributes are extracted automatically from the 3D computer model except: activity name and duration, which are entered by the user, and the early and late start and finish dates, which are calculated by the system according to the generated sequence of the dynamic sequencer.

**Class DBF attributes:** class name, class description, direction of installation in the X, Y, and Z direction, and the priorities of the direction of installation. This database file contains the knowledge needed in the reasoning process to sequence activities of the same class. This knowledge is entered in the system by the user once for all similar projects.

**Zoning DBF Attributes:** zone name, minimum X, maximum X, minimum Y, maximum Y, minimum Z, and maximum Z. The minimum and maximum values define the boundaries of the zone.

**Activity/Class Relations DBF Attributes:** activity name and the related class name. This knowledge is to be entered by the user. However, with a proper coding of objects during the design of the 3D computer model such knowledge can be extracted from the label (code) of the objects in the 3D computer model.
Activity Connection DBF Attributes: connection name, the names of the two connected activities, and the connection type. This data is to be entered by the user. However, a knowledge-based system can be implemented within the 3D computer modeling system to generate such data.

Dominating Class DBF Attributes: class domination name, the names of the two related classes, and the name of the dominating class. This knowledge is to be entered by the user once for all similar projects.

Special Class Relations DBF Attributes: class relation name, the names of the two related classes, the connection type, the relations between the two classes in the X, Y, and Z directions, and the name of the class of the object to be installed first. This knowledge is to be entered by the user once for all similar projects.

7.4.3. The Central Database User Interaction

As mentioned before, the user interaction with the KNOW-PLAN prototype system is accomplished through the central database. Most of the data and knowledge needed to perform the automatic generation of construction plan by the dynamic sequencer and the visual simulation of the construction process by the visual simulator are entered via the central database. The central database is designed as an interactive menu-driven system. The main menu of the system includes three options:

1. Preprocessor data manipulation
2. Postprocessor data manipulation
3. Exit to DOS

The preprocessor menu includes options to manipulate each one of the different entity
types or entity relationships data introduced in this chapter. The postprocessor options
are described in Chapter 9. Selecting the "activity data" option from the preprocessor
menu will lead to the "Activity Data Menu". The activity data menu has six options:

1. Activity data manipulation
2. Read Geometric data
3. Calculate center
4. Generate ASCII file for KNOW-PLAN
5. Generate ASCII file for Optimizer
6. Read optimized file

The first option gives the user the option to manipulate the contents of the activity DBF.
The manipulation of the activity DBF allows the following functions to be performed:
create, edit, delete, view, view and delete, clear the DBF, sort, screen report, and printer
report. The second option reads the ASCII file extracted from the 3D computer model
and maps the file’s data as records in the activity relation table. The third option calculates the center of each object (if the data is not extracted from the 3D computer model).
The fourth option generates an ASCII file which contains all the data about the activities needed by the KNOW-PLAN’s dynamic sequencer. The fifth option generates an
ASCII file for a data optimizer. The last option reads the activity optimized file and
maps the file’s data as records in the activity relation table. The data optimizer in not
implemented; however, the interface between the central database and the data optimizer
is developed.
The class menu has four options:

1. Class data manipulation
2. Generate ASCII file for KNOW-PLAN
3. Generate ASCII file for Optimizer
4. Read Optimized file

These options performs the same functions performed by the similar options in the activity menu. The remaining types of entities and relationships menus has the same four options listed for the class menu. Figure 7.6 depicts the hierarchical structure of the central database menus for the preprocessor phase. The menus for the postprocessor phase are described in Chapter 9.

7.5. The Constructability Review Knowledge

The Constructability Task Force of the Construction Industry Institute (CII) defines constructability as: "the optimum use of construction knowledge and experience in planning, engineering, procurement and field operations to achieve overall objectives". Significant advances in design could be achieved if design and construction professionals could test for constructability prior to the start of construction. Design professionals could test their initial sequencing of activities for constructability, as well as future sequencing due to out of sequence components. Lack of constructability is one of the primary causes of construction rework and subsequent claims in construction [Morad 88].
Figure 7.6: The Central Database Menu Structure (Preprocessor Phase)
Some constructability issues can be considered as the planning of collision free paths in which the following problem is addressed: "How does one get from here to there?". This problem is solved so often and so intuitively by people that they are seldom aware of the complexity underlying it. The collision free path in construction is the problem of determining a continuous path through which an object and its manipulation mechanisms can be moved from an initial position and orientation to a final position and orientation without having interferences with other objects in space.

This section presents the utilization of Path-Finder, an AI-based system developed by Bechtel Power Co., to provide knowledge about constructability constraints to be considered during the reasoning process by the dynamic sequencer. The Path-Finder is described in detail in Appendix B.

The Path-Finder system provides the user with tools to check if a specific planning scenario is feasible or not. This process is performed interactively by the user through the simulation of object movement from a specific position to a final position in the 3D computer model. The outcomes of the simulation process is a list of constraints that should be considered during the planning process. A typical list of constraints will look as follows:

Object #3 cannot be installed
if Object#1 is installed
and if Object#5 is installed
and if Object#6 is installed

However, it can be installed
if Object#1 is installed
and if Object#5 is not installed
and if Object#6 is installed

This means that Object #3 to be installed before Object #5. This constraint can be asserted in the knowledge-base as an execution fact in the form:

(Execute Object#3 Object#5 CONST 90)

Where, "Execute" means install Object#3 before Object#5. "CONST" is the name of the network in which this logic (link) to be added in the context of the KGB system (the Dynamic Sequencer). "90" is the priority of this link. More discussion about logic facts is included in Chapter 8.

It is envisioned that in a future version of the system the constructability checking will be performed automatically without the user interaction for the critical objects in the 3D computer model. Critical objects of the 3D computer model can be identified by the designer during the design phase.

7.6. Other Knowledge

7.6.1. User's Defined Knowledge

In the KNOW-PLAN prototype system, the user is allowed to define rules and constraints and to add to the knowledge-base of the dynamic sequencer to be considered during the reasoning process. The separation of the knowledge from the control mech-
anism is the main advantage of knowledge-based systems. The user can add knowledge to the system in two forms. The first form is as declarative knowledge such as facts and schemata. The second form is as procedural (operating) knowledge such as rules.

A typical execution fact which can be added to the knowledge-base at any time during the reasoning process may look as follows:

(execute Object#9 Object#7 USR1 100)

The architecture of the KNOW-PLAN dynamic sequencer allows as many facts or rules to be added to the knowledge-base by the user as different knowledge sources. The links (logic) generated based on this knowledge sources is posted to the context of the system as execution facts. These execution facts combined with the execution facts generated by the system's defined knowledge will be used by the refinement knowledge source (critics) to come up with the final logic. Chapter 8 discusses these issues in more details.

7.6.2. External Knowledge Sources

The KNOW-PLAN prototype system is designed to accommodate any knowledge source which follows the structure of the knowledge-base of the system. The only constraint on these knowledge sources is that they have to generate links between activities in the form of execution facts which has the following pattern:

(Execute object#(i) object#(j) KS(N) Prio-level)
The system has its own rules which define the refinement knowledge source (critics) that combine all the logic facts to come up with a final logic.

7.7. Data-Flow in the Central Database

The second task in the KNOW-PLAN prototype system is to outline the knowledge and data needed in the reasoning process for generating a construction plan. The central database, during the preprocessor phase, functions as a mean of communication for passing data from the 3D computer modeling system to the dynamic sequencer. The central database provides the tools to manipulate the extracted data, from the 3D computer model, and the data defined in the database files that represents the available knowledge. The central database interacts with the dynamic sequencer which is the keystone of the KNOW-PLAN prototype system.

Figure 7.7 depicts the data-flow in/out of the central database during the preprocessor phase. The different kind of knowledge passed or entered into the dynamic sequencer includes: systems' defined geometric knowledge, constructability knowledge, users' defined knowledge, and external knowledge sources. The various knowledge data are shown in the figure as they flow through the system.
Figure 7.7: Data Flow in the Knowledge and Data Module
8. The Dynamic Sequencing Module

The third module in the KNOW-PLAN prototype system is the dynamic sequencing module. The processor in this module is called the Dynamic Sequencer. The dynamic sequencer is a KGB system which generates construction plans based on the knowledge stored in its knowledge-base. The dynamic sequencer is the center piece of the KNOW-PLAN prototype system. The dynamic sequencer is implemented based on many AI concepts and techniques. The different concepts utilized includes: least-commitment principle, state-space representation, and object-oriented programming approach. Section 8.1. describes briefly the role of each concept in the reasoning process of the dynamic sequencer.

The primary source of reasoning knowledge in the dynamic sequencer is the objects' geometric data and the class knowledge stored in the central database. The system incorporates external data interface rules to automatically map the data and knowledge stored in the central database to the schemata structure of its knowledge-base.
The knowledge in the knowledge-base of the system is represented in two forms. The first form is the declarative knowledge which is represented by facts and schemata (frames). The knowledge-base has a hierarchical schemata structures which describe the different entities of the domain knowledge. Facts, in the knowledge-base, are created internally by the dynamic sequencer during the reasoning process. The procedural knowledge is the second form of knowledge. Rules are the primary form of this type of knowledge in the dynamic sequencer. The rules in the system are categorized into different groups. Each group perform a specific task during the reasoning process.

This chapter introduces the different AI concepts applied in the implementation of the dynamic sequencer. The different classifications of rules are presented. A detailed description of the dynamic sequencer's reasoning process is presented. The schedule calculation process which is based on an object-oriented programming approach is presented. Finally, the data flow and the reasoning process of the dynamic sequencer are summarized.

8.1. The Dynamic Sequencer Concepts

The dynamic sequencer is a KGB system which is developed using the ART programming language. The system adopts most of the features of knowledge-based systems. However, it doesn't provide the typical explanation facilities that exist in knowledge-based systems. The control strategy of the dynamic sequencer is a Data-Driven (forward chaining) control strategy. The reasoning process starts from a state in which the system
has no a priori knowledge of the possible solution. It uses the acquired knowledge to evaluate the tree of possibilities as it progresses through the solution procedure.

The system adopts, in addition to the primary concepts of knowledge-based systems, different AI concepts at different stages of the reasoning process. The primary AI concepts adopted in the system include: least-commitment principle, state-space transformation, and object-oriented programming approach.

8.1.1. Least-Commitment Principle

The system can be described as an implementation of the least-commitment principle. In the least-commitment principle, decisions are postponed to the last possible moment, when the maximum amount of information is available. This has the effect of conducting decision making with the availability of information. This approach avoids the possibly negative impact of arbitrary decisions [Rolston 88].

In the reasoning process of the dynamic sequencer, the rules use the stored knowledge to find potential relationships (links) between activities. However, the relationships that are found are not asserted as part of the final logic. All the relationships are posted into the knowledge-base in the form of execution facts. At the final stage, all the potential relationships are criticized to come up with the final logic. In this way, any conflict between different links are avoided. Accordingly, the least-commitment principle is utilized to postpone the assertion of the relationships as part of the final logic until the last stage of the reasoning process. This is done when all information about potential relationships have been studied by the system.
8.1.2. State-Space Representation

State-space representation is used in AI applications to formally represent the problem environment. It is the basis for most of the problem solving strategies. The representation concept is introduced in Chapter 3 "Technology Review". The concept of the state-space representation is adopted implicitly by the dynamic sequencer in the final stage of the reasoning process. It is used by the refinement knowledge source to come up with the project plan by criticizing all the networks generated by the reasoning process in preceding stages. The state-space representation of the final stage of the plan generation reasoning process is described in details in section 8.3.2.1.

8.1.3. Object-Oriented Programming Approach

ART supports several styles of programming. The object-oriented programming approach is one of these styles. The fundamental unit in ART's object-oriented programming is the object, represented by a schema. The object reacts to the messages sent to it by searching within its slots for a method which is appropriate to that message. The method is procedural in nature. It performs a sequence of ART commands to reply to the message.

The object-oriented programming approach is used by the dynamic sequencer at the last stage of the reasoning process. Messages are sent to the activity schemata to calculate their schedule dates. The activity schemata perform the methods that are activated by the sent message. The result is the early schedule and late schedule dates for that specific
activity. The schedule dates are based on the logic generated by the refinement reasoning process.

8.2. The Dynamic Sequencer Procedural Knowledge

(Phase I)

The main objective of the reasoning process of the dynamic sequencer is to come up with a project plan. This plan should satisfy all the constraints imposed by the knowledge stored in the knowledge-base. The dynamic sequencer utilizes the two primary types of knowledge: declarative and procedural (operational) knowledge. The declarative knowledge represents the refined data that describes the different classification of objects (entities) within the problem domain. This kind of knowledge represents: activities, classes, zones, and relationships between them. During the reasoning process, the concluded relationships among activities are represented by declarative knowledge in the form of execution facts which are posted into the knowledge-base.

The procedural knowledge consists of rules that perform the actual reasoning process. The reasoning process of the dynamic sequencer is divided into two phases. The first phase consists of the necessary steps to conclude (assert) potential relationships among different activities. These relationships are logically viewed as different networks depending on the knowledge source that asserts such relationships. The second phase of the reasoning process includes rules that combine the generated networks into a final network which represents a possible solution for the planning problem. In general, the
rules of the system are classified into different groups depending on the outcome of their actions.

In this section, the first phase of the reasoning process is described in detail. Each classification of rules is discussed separately. The second phase of the reasoning process is discussed in detail in section 8.3.

In the first phase, different classifications of rules are executed. There are rules that provide an external data interface procedures. Other rules allocate activities into zones. The main rules in this phase are the rules that are executed to perform the geometric reasoning process. Finally, other rules are executed to conclude relationships between activities based on user-defined constraints, constructability constraints, or other knowledge sources’ constraints.

8.2.1. External Data Interface Rules

In Chapter 7, the central database management system of the KNOW-PLAN prototype system is introduced as the link between the 3D computer modeling system and the dynamic sequencer. Also, the central database is designed to be the mean of communication with the system developer and the user to enter the necessary knowledge required by the geometric reasoning process.

In order to make the dynamic sequencer open to the external computing environment, external data interface procedures are required. These procedures will enable the system to map the data stored in the central database into the knowledge-base. ART provides
a variety of functions which provide an open architecture so that data integration with a variety of data sources may be added.

External data interface is implemented in the dynamic sequencer by a set of rules. Each rule provides an access to an external ASCII file. The ASCII files contain the different instances of an entity type or entities' relationships. These ASCII files are generated automatically within the central database from the different database files that exist in the database. There are seven external data interface rules in the dynamic sequencer knowledge-base. The following is a list of these rules:

1. Activity external data interface rule;
2. Zone external data interface rule;
3. Class external data interface rule;
4. Activity/class relations external data interface rule;
5. Activity connections external data interface rule;
6. Dominating class relations external data interface rule;
7. Special class relations external data interface rule.

Each of the above listed rules performs the necessary interface functions that create or modify the schemata structure of the concerned entity type or entities' relationships instances as they exist in the knowledge-base. Therefore, the activities, zones, classes, activity/class relations, activity connections, dominating class relations, and special class relations schemata are originated during the interface process. Figure 8.1 depicts a list of the external data interface rules and the structure of the schemata that are created during the interface process.
Figure 8.1: External Data Interface Schemata Mapping Structure
Figure 8.2 through Figure 8.8 depicts flow charts of the interface process as it is executed by the external data interface rules. The flow charts are self-explanatory. The source code listing of the different rules and the related external data interface functions are included in Appendix C1.

8.2.2. Zone Allocation Rule

During the external data interface process an activity schema is created for each activity in the central database. Two empty relation slots are created for each activity. These relation slots are: ACT-ZONE and ACT-ZONE-BOR. The ACT-ZONE slot is created to contain the name of the zone schemata that the 3D object, which is associated with the activity, is located in. An object is located in a zone if the volume boundaries of the object partially or entirely overlaps with the volume boundaries of the zone. The zone boundary values are stored in the zone schemata which are created during the external data interface process.

Geometrically, two volumes are entirely overlapped if one volume is embedded in the other. Assume that the values of the boundaries of volume #1 are: minX(V1), maxX(V1), minY(V1), maxY(V1), minZ(V1), and maxZ(V1). And the values for volume #2 are: minX(V2), maxX(V2), minY(V2), maxY(V2), minZ(V2), and maxZ(V2). Then, volume #2 is embedded in volume #1 if the following conditions apply:

\[
\begin{align*}
\text{minX(V1)} & = < \text{minX(V2)} = < \text{maxX(V2)} = < \text{maxX(V1)} & \text{ and} \\
\text{minY(V1)} & = < \text{minY(V2)} = < \text{maxY(V2)} = < \text{maxY(V1)} & \text{ and} \\
\text{minZ(V1)} & = < \text{minZ(V2)} = < \text{maxZ(V2)} = < \text{maxZ(V1)}
\end{align*}
\]

8. The Dynamic Sequencing Module

166
Figure 8.2: Activity External Data Interface Procedure
Figure 8.3: Zone External Data Interface Procedure
Figure 8.4: Class External Data Interface Procedure
Define a string for act/class relation record
Define the format of the act/class relation record file
Define a global variable for the buffer name to contain the act/class relation string

Open a stream for act/class relation data and bind it to ?stream variable

Do while not end of file (EOF) Yes
Read line from the current stream and store it in ?line-buf variable

Yes
Reach end of file?

Extract the value of the first field as per the defined EDI function and bind it to activity schema name

Extract the value of the second field as per the defined EDI function and bind it to the act-class slot

Does the act-class slot exist in the schema?

Yes
Create the slot act-class in the activity schema

No

Close the stream

Assert the the class as a value of the slot act-class

Figure 8.5: Activity/Class Relation External Data Interface Procedure
Figure 8.6: Activity Connection External Data Interface Procedure
Figure 8.7: Dominating Class External Data Interface Procedure
ART
External Data Interface (EDI) Functions

Open a stream for class relation data and bind it to stream variable

Do while not end of file (EOF)

Read line from the current stream and store it in line-buf variable

Reach end of file?

Extract the value of the first field as per the defined EDI function and bind it to the schema name

Does the schema exist?

Create the schema slots:
cls-1  cls-2
rel-X  rel-Y  rel-Z
con-type  cls-install

Map the buffer data into the schema slots as per the designed format of the EDI functions.

Close the stream

Assert the current schema as a class-relation schema

Figure 8.8: Special Class Relation External Data Interface Procedure
On the other hand, two volumes are partially overlapped if they share a common volume. In the KNOW-PLAN prototype system, an object is located in a specific zone if it partially or entirely overlaps with the zone's boundaries. Accordingly, the dynamic sequencer uses the zone allocation rule to assert the zones in which an object is located. The relation slot ACT-ZONE of the activity schema, related to the object, will contain the name of the zone schemata that the object is located in.

The ACT-ZONE-BOR is defined to contain the names of the zone schemata that the object shares a common surface with. The ACT-ZONE-BOR plays an important role in asserting relationships between activities of different zones if they share common surfaces. An object shares a common surface with a zone if one of these conditions applies:

1. The minimum value of the object in a specific direction equals the maximum value of the zone in that direction. And the ranges of the object's boundary values in the other two directions overlap with the ranges of the zone's boundary values in these two directions.

2. The maximum value of the object in a specific direction equals the minimum value of the zone in that direction. And the ranges of the object's boundary values in the other two directions overlap with the ranges of the zone's boundary values in these two directions.

Accordingly, the dynamic sequencer uses the zone allocation rule to assert the zones with which the object shares a common surface. The relation slot ACT-ZONE-BOR of the activity schema, related to the object, will contain the name of the zone schemata that the object shares a common surface with.

8. The Dynamic Sequencing Module
The source code listing of the zone allocation rule is included in Appendix C2. Figure 8.9(a) depicts the different cases of the object/zone spatial relationships in which an object can be related to a zone. The different cases are: partially overlapped, entirely overlapped, share a common surface, and no overlap exists. Figure 8.9(b) depicts an example of an object's spatial relationships with different zones of the project's work space. Figure 8.10 depicts the geometric conditions that apply to the different geometric relationships between an object and a zone. The same cases will be used in the geometric reasoning process in section 8.2.2. to define the geometric relationships between two objects.

8.2.3. Geometric Reasoning Rules

The primary advantage of the KNOW-PLAN prototype system over other knowledge-based planning systems is its ability to utilize planning knowledge that resides in a 3D computer model. It uses this knowledge during the reasoning process to generate a possible construction plan. The component of the overall reasoning process which utilizes this knowledge, which knows about geometry and spatial relationships among different objects of the 3D computer model, is called geometric reasoning. The outcomes of the geometric reasoning process is a network which defines potential relationships among different activities based on geometric relationships.

The geometric reasoning process is based on the conclusion of spatial relationships of the project objects in conjunction with the knowledge that define the classes, direction of installation of objects of the same class, direction of installation of objects of different classes, and connection type between objects of different classes. Chapter 7 provided
No Overlap  Entirely Overlapped  Partially Overlapped

Share a Common Surface

(a) Spatial Relationship Cases

(b) Example of Object-Zone Spatial Relationship

*Figure 8.9: Object-Zone Spatial Relationships*
Geometric data comparison between two entities "A" and "B". "A" and "B" can be two objects, an object and a zone, or two zones. Five cases can exist between the two entities. The test condition for each case is explained below:

**CASE #1:**
Overlapping or Sharing Common Surface, same as (NOT (NOT Overlapping))

\[
\begin{align*}
\text{(NOT (OR) } & \langle \max X_A \quad \min X_B) \\
& \langle \max Y_A \quad \min Y_B) \\
& \langle \max Z_A \quad \min Z_B) \\
& \rangle \min X_A \quad \max X_B) \\
& \rangle \min Y_A \quad \max Y_B) \\
& \rangle \min Z_A \quad \max Z_B))
\end{align*}
\]

**CASE #2:**
No Overlapping No Common Surface

\[
\begin{align*}
\text{(OR) } & \langle \max X_A \quad \min X_B) \\
& \langle \max Y_A \quad \min Y_B) \\
& \langle \max Z_A \quad \min Z_B) \\
& \rangle \min X_A \quad \max X_B) \\
& \rangle \min Y_A \quad \max Y_B) \\
& \rangle \min Z_A \quad \max Z_B))
\end{align*}
\]

**CASE #3:**
Only Overlapping

\[
\begin{align*}
\text{(NOT (OR) } & \rangle \max X_A \quad \min X_B) \\
& \rangle \max Y_A \quad \min Y_B) \\
& \rangle \max Z_A \quad \min Z_B) \\
& \rangle \min X_A \quad \max X_B) \\
& \rangle \min Y_A \quad \max Y_B) \\
& \rangle \min Z_A \quad \max Z_B))
\end{align*}
\]

**CASE #4:**
Sharing Common Surface or Not Overlapping

\[
\begin{align*}
\text{(OR) } & \rangle \max X_A \quad \min X_B) \\
& \rangle \max Y_A \quad \min Y_B) \\
& \rangle \max Z_A \quad \min Z_B) \\
& \rangle \min X_A \quad \max X_B) \\
& \rangle \min Y_A \quad \max Y_B) \\
& \rangle \min Z_A \quad \max Z_B))
\end{align*}
\]

**CASE #5:**
Share Common Surface Only

Is the case when the test passes: CASE #1 and CASE #4

*Figure 8.10: Geometric Data Comparison Between Objects & Zones*
examples of how the knowledge about classes is used to generate relationships between activities.

The geometric data comparison between two objects is performed by the geometric reasoning rules. These rules define which object among any two objects is higher in the X direction, which is higher in the Y direction, and which is higher in the Z direction. Figure 8.11 depicts the different cases that can exist between two objects in a specific direction. The different cases can be summarized into three conditions:

1. Object A is lower than object B, or object B is higher than object A;
2. Object A is higher than object B, or object B is lower than object A;
3. Object A and Object B have the same range in a specific direction.

This conclusion is used in a geometric reasoning rule to assert the logic between the two activities (objects) in the form of execution facts.

There are three geometric reasoning rules in the KNOW-PLAN prototype system. The three rules assert logic between activities based on the following conditions:

1. Rule#1: asserts logic between activities of the same class in the same zone.
2. Rule#2: asserts logic between activities of the same class in different zones.
3. Rule#3: asserts logic between activities of different classes and located either in the same zone or different zones.

The first two rules assert logic between two activities only if the two objects, associated with the two activities, are geometrically overlapping or sharing a common surface. The
CASES:
1. (A Lower than B) or (B Higher than A)
2. (A Higher than B) or (B Lower than A)
3. (A same as B) or (B same as A)

Figure 8.11:
Geometric Data Comparison Between Two Objects in: X, Y, or Z
cases of overlapping and sharing common surface described in Figure 8.9 and Figure 8.10 between an object and a zone can be applied here to geometrically relate two objects.

In the case of objects that are of the same class and located in the same zone, the rule starts by checking if the two objects are overlapping or sharing a common surface. If the two objects are overlapping or sharing a common surface, the rule finds which object is higher in each direction of the three coordinate axes: X, Y, and Z. Using the knowledge stored about the direction of installation and the priority of installation of the objects' class, the rule asserts an execution fact that define a possible logic between the two objects based on their spatial relationship. Figure 8.12 depicts a flow chart of the procedural code that is performed by the action side of this geometric reasoning rule.

In the case of two objects of the same class and located in different zones, the same procedure used in the first rule is used. However, the rule checks only if the two objects share a common surface. If it finds a common surface, the rule proceeds with the remaining actions. Figure 8.13 depicts a flow chart of the procedural code that is performed by the action side of this geometric reasoning rule.

The third rule asserts logic between activities of different classes. The rule is applied to activities in the same zone or different zones. Two approaches have been tested to assert logic between activities of different classes. The first approach utilizes the same equations for geometric data comparison. The second approach uses the type of connection between the objects as an important factor in asserting this kind of logic.

In the first approach, the direction of installation can be defined based on the geometric relationship between the two classes to which the activities belong. There are two main
Figure 8.12: Geometric Reasoning Rule (Activities of Same Class and Zone)
Figure 8.13: Geometric Reasoning Rule (Activities of Same Class and Different Zones)
drawbacks to this approach. First, a large search space is required. This is due to the fact that each object will be compared with all objects in the same zone and all objects that are located in a neighbor zone and share a common surface with the objects. Second, the direction of installation between objects of two different classes usually is not unique. It is based on the type of connection that exists between two objects.

The logic between two activities of different classes can be a function of the type of connection between the objects related to the two classes. For example, if the first object is of class 'mechanical' and the second object is of class 'masonry wall', the mechanical object will be installed before the wall. This is valid if the mechanical object is a pipe and it is connected with the wall with a "embedded in" type of connection. However, the wall should be installed first if the mechanical object is an equipment to be "supported by" the wall.

The second approach utilizes the type of connection as a factor in the logic definition decision. Classes can be related to each another based on the type of connection between the objects that are related to these classes. Therefore, two classes can have different direction and priority of installation based on the type of connection between the objects.

The second approach has been utilized in the development of the KNOW-PLAN prototype system to avoid the deficiency of the first approach in addressing the different cases of the type of connection between the objects. According to this approach, the rule first checks which object is higher in each direction: X, Y, and Z. Then it checks if their is a special class relation defined in the knowledge-base that match the geometric relationship between the two objects. If a special class relation exists; the direction of installation between the two classes, to which the two objects are related, is used to assert
the logic. If a special class relation doesn't exist, then the dominating class relation is used to assert the logic. Figure 8.14 depicts a flow chart for the procedural code that is performed by the action side of this geometric reasoning rule.

The geometric reasoning rules assert facts in the knowledge-base to represent the concluded logic between activities. These facts are called 'Execution Facts'. The facts have a relation name and a sequence of four ART objects. The relation name is 'Execute'. The first ART object represents the name of the preceding activity. The second ART object represents the name of the succeeding activity. The third ART object represents the name of the network to which the logic link belongs. All the facts that are generated by the geometric reasoning rules are asserted as part of the geometric network. The ART object that represents the links of this type of network is called 'GEOM'. The last ART object in the sequence represents the priority of the link. The priority value of each link plays an important role in the process of combining the links of different networks into a final network at the last stage of the dynamic sequencer's reasoning process.

The geometric reasoning process asserts execution facts with a priority value of '50'. This priority value is expected to be the lowest value for the execution facts in the knowledge-base. A typical execution fact asserted by the geometric reasoning rules may look as follows:

(Execute ACT-1 ACT-2 GEOM 50)

The source code listing of the geometric reasoning rules are included in Appendix C3. The first rule is for activities of the same class and located in the same zone. The second rule is for activities of the same class but located in different zones. The third rule is for activities of different classes.
Figure 8.14: Geometric Reasoning Rule (Activities of Different Classes)

Legend:
- ?C1: the class of the first object
- ?C2: the class of the second object
- ?rel-X: the class of the higher object in X direction
- ?rel-Y: the class of the higher object in Y direction
- ?rel-Z: the class of the higher object in Z direction
- ?install: the class of the object to be installed first

Object of ?C1 then object of ?C2 means
Install the activity related to the object of class ?C1 before the activity related to the object of class ?C2

START

Execution Facts

Object of ?C2 then Object of ?C1
Object of ?C1 then Object of ?C2
Object of ?C1 then Object of ?C1
Object of ?C2 then Object of ?C1
8.2.4. User and Other Logic Definition Rules

In the KNOW-PLAN prototype system the user is allowed to define rules to define specific constraints or requirements as separate knowledge sources. The outcome of the constructability checking can be incorporated in the dynamic sequencer in the form of rules or directly asserted as execution facts. The user is also allowed to assert directly execution facts to be considered during the final reasoning process to generate the combined network. The only constraint on the rules that are defined by knowledge sources other than the geometric reasoning rules is to assert execution facts in the form of:

(Execute ACT(i) ACT(j) Network-name Priority)

The different execution facts are grouped logically into networks which define the source of the facts. The source of the facts can be from geometric reasoning, constructability checking, user defined knowledge, ...etc. The name of the network has nothing to do with the final generation of the plan. It is only used to provide the source of each fact as it is asserted in the knowledge-base. Therefore, it is used as a source of explanation facility for the user.

Figure 8.15 depicts a sample logic networks generated based on different knowledge sources. In the example, four networks are illustrated: geometric network, constructability network, user(1) defined network, and user(2) defined network. Figure 8.16 depicts the combination of these four networks. The combined network has conflicts, loops, and implied logic. The task of the second phase of the reasoning process is to optimize this network to avoid conflicts, loops, and implied logic to come up with the
final project plan. The reasoning process in the second phase of the dynamic sequencer is discussed in the following section.

8.3. The Dynamic Sequencer Procedural Knowledge

(Phase II)

In phase I of the dynamic sequencer reasoning process, different knowledge sources assert execution facts in the knowledge-base to define potential logic between different activities. The second phase of the dynamic sequencer takes as input all asserted execution facts. These execution facts are processed by the dynamic sequencer to come up with the final project plan by criticizing the asserted logic.

The criticism process is based on the refinement of the logic between activities to avoid redundancy, conflict, implied logic, and loop logic. The dynamic sequencer has a set of rules that perform the refinement reasoning process. This set of rules can be viewed logically as an additional knowledge source which knows how to refine a project plan. In this section, the rules are called critics. There are different critics in the system. Each type of critics performs a specific task in the refinement reasoning process. The refinement reasoning process is described below in detail.
Figure 8.15: Knowledge Sources’ Related Networks
8.3.1. Preliminary Logic Refinement Rules

The preliminary logic refinement is the first step in the refinement reasoning process. The purpose of this step is to retract the execution facts of low priority that are implied by other high priority execution fact. Also, to retract all the facts of low priority that are conflicting with other high priority execution facts. There are two rules in the dynamic sequencer that accomplish these tasks. Appendix C4 lists the source code listing of these two rules.

The first rule refines execution facts of the same logic but with different priorities. Assume the following two execution facts exist in the knowledge-base:

\[
\begin{align*}
(\text{Execute} & \quad \text{ACT-1} & \quad \text{ACT-2} & \quad \text{network#1} & \quad 50) \\
(\text{Execute} & \quad \text{ACT-1} & \quad \text{ACT-2} & \quad \text{network#2} & \quad 60)
\end{align*}
\]

The first execution fact is implied by the second execution fact. Since the priority of the second execution fact is higher than the priority of the first execution fact, the rule retracts the first execution fact from the knowledge-base.

The second rule refines the execution facts of opposite logic and of different priorities. Assume the following two execution facts exist in the knowledge-base:

\[
\begin{align*}
(\text{Execute} & \quad \text{ACT-1} & \quad \text{ACT-2} & \quad \text{network#1} & \quad 50) \\
(\text{Execute} & \quad \text{ACT-2} & \quad \text{ACT-1} & \quad \text{network#2} & \quad 60)
\end{align*}
\]

Since the second execution fact has higher priority than the first execution fact, the rule retracts the first fact. If the priority values are the same in the two execution facts, the
rule will assert a new fact that declares the association between the two facts as reversible logic. The new fact takes the following form:

\[(\text{rev-eq-prio} \text{ ACT-1 ACT-2 prio-value})\]

The conflict can be resolved by defining new rules in the system to prompt the user to select among two logic. Or by providing another form of conflict resolution based on specific criteria.

8.3.2. Final Logic

At the current stage of the refinement reasoning process, after the preliminary refinement logic, the knowledge-base contains many execution facts that describes the potential logic between the different activities of the project. The primary purpose of the refinement reasoning process is to manipulate these execution facts to come up with a final project plan which has no conflict. In order to accomplish this task, the dynamic sequencer creates new schemata structures that holds the current state of the reasoning process. The data included in these schemata is used to finalize the project plan.

The following subsections describe the structure of the new schemata. Followed by the different conflict cases between defined logic among different activities. Finally, the process of generating the final plan is described.
8.3.2.1. "State-Space" Representation

The concept of state-space representation is explicitly utilized in the refinement reasoning process. The refinement reasoning process proceeds from an initial state (at which the program has no a priori knowledge about the solution), through an intermediate state (in which the program's knowledge of the solution is improved), to a final state (when the program has reached a possible solution). Each state in the state-space represents the status of different logical ART-objects in the knowledge-base. Different types of logical ART-objects which describe the current state of the reasoning process exist. There are schemata that represent the different logic (links) between different activities. There are schemata that represent the activities. In addition to two facts that describe the members of an opened-list and closed-list.

The links schemata define part of a state description. These schemata are created during the refinement reasoning process to map the components of the execution facts into schemata definition. A link schema for every execution fact is created. A typical execution fact of the form:

(Execute ACT-1 ACT-2 Network Priority-value)

is mapped to a link schema of the following structure:

(schema LNK-ACT-1-ACT-2
  (is-a LNK)
  (from-act ACT-1)
  (to-act ACT-2)
  (type network)
  (prio priority-value)
The slot "used" in the link defines the status of the link at different states of the state-space. If the 'used' slot has a value of 'Yes' at a specific state, that means this link is a member of the final plan at the current state. If the value is 'No', that means this link is not a member of the final plan at the current state. The changes from one state to another is based on the conclusion of the refinement reasoning process at different states. The "opposite" slot value relates a link to its reversible link, which is a link between the same two activities, with the same priority value, but in different direction.

The O-node (original node) schemata defines another part of the state description. An O-node schema is created for every activity schema in the knowledge-base. The O-node schemata are the reservoir for the status of the current schedule attributes of an activity at different states of the refinement reasoning process. The structure of the O-node schema of an activity schema ACT-1 may look as follows:

```
(schema O-N-ACT-1
  (is-a O-node)
  (act ACT-1)
  (act-before)
  (act-after)
  (init-act-before)
  (init-act-after)
  (status))
```

The act-before, act-after, init-act-before, init-act-after, and status slot values are asserted during the refinement reasoning process. The values are changeable depending on the status of the current solution at a specific state. The init-act-after and init-act-before are
two slots that initialize the 'priority cycles' of the refinement reasoning process. The priority cycles of the refinement reasoning process are described later in this chapter.

The system could have been implemented without creating the O-node schemata. This can be done simply by appending the structure of the activity schemata. However, the O-node schemata are created to simplify the debugging process during the development of the KNOW-PLAN prototype system.

The H-node (hierarchical node) schemata define another component of a state description. These schemata represent a hierarchical tree of nodes that point to the predecessors and successors of each activity. An H-node schema is created for each activity schema in the knowledge-base. The H-node schemata hierarchy contains information about the ancestors, parents, sons, and descendants of each H-node. The H-node schemata are mainly used in the refinement reasoning process to check for conflicts in the project plan.

The structure of an H-node schema of the ACT-1 activity schema may look as follows:

(schema H-N-ACT-1
  (this-is H-node)
  (act ACT-1)
  (ancestor-N))

The ancestor-N slot values of the H-node schema describes the list of the ancestors of an activity at a specific state. The H-node schemata hierarchy can be viewed as a precedence network with nodes as activities and ancestor-N slot values as the links (logic) between the activities.

8. The Dynamic Sequencing Module
In addition to the structure of the link schemata, O-node schemata, and H-node schemata, a state is described by the members of an opened-list and closed-list. The members of the opened-list and closed-list are the name of the activities that are to be processed or already processed, respectively, by the current priority cycle of the refinement reasoning process. The two lists are represented internally by two facts in the knowledge-base. The following is an example of these two facts:

(opened-list ACT-5 ACT-6 ACT-7 ACT-8)
(closed-list ACT-1 ACT-2 ACT-3)

The role of these two facts in the final reasoning process is described in Subsection 8.3.2.3.

Accordingly, at an intermediate state of the refinement reasoning process, a state is described by the slot values of the different schemata that are part of the state description. Also, the state is described by the members of the opened-list and the closed-list. The reasoning process proceeds from one state to another by changing the values of some slots that describes the state. Also, by moving members from the opened-list to the closed-list and posting new members to the opened-list. The full description of the refinement reasoning process is included in subsection 8.3.2. Appendix C5 includes the source code listing of the rules that create the link schemata, O-node schemata, and H-node schemata.

8.3.2.2. Conflict Resolution in the Project Plan Logic

The primary objective of the refinement reasoning process by the dynamic sequencer is to come up with a project plan that has no conflict between the links among the different
activities. The refinement reasoning process attempts to criticize the links that resides in the knowledge-base so conflict between different links is avoided.

The refinement reasoning process starts with the higher priority links and asserts them as permanent members of the final plan network. The process proceeds by evaluating lower priority links. The evaluation concludes whether the selected link is to be asserted as a member of the final plan network or not. If a conflict is expected to arise from asserting it in the network, the link is asserted as having a slot value of 'No' in the 'used' slot. If no conflict is expected, a slot value of 'Yes' is asserted, i.e. it is a member of the final plan network. The process proceeds by selecting links according to their priority values. This process is performed at different cycles called 'priority cycles'.

There are two types of conflicts that could arise by asserting a link in the final plan. The first type is referred to as 'implied logic'. Implied logic occurs if a link is already implied by two or more links. If ACT-1 is asserted as a predecessor of ACT-2 by a link LNK-1, and ACT-2 is asserted as a predecessor of ACT-3 by the link LNK-2, then a link LNK-3 from ACT-1 to ACT-3 is already implied by the other two links. Therefore, there is no need to assert LNK-3 as a member of the final plan logic.

The refinement reasoning process uses the H-node schemata structure to check for implied logic. The concept of ancestor, parent, son, descendant relationship between the H-nodes is used to predict implied logic cases. Figure 8.17 depicts the different cases that can occur between different H-nodes (which represent activities). In the figure, black links defines links which have a slot value 'Yes' in the 'used' slot. That means they have been already asserted as members of the final plan network by previous priority cycles. The white link represents the current link, between ACT-A and ACT-B, that is to be
tested for the presence of implied logic. According to the figure different cases can exist. The different cases are:

**Case #1**: The tested link is to be asserted as a member of the final plan network. None of the already asserted links has to be retracted (no implied logic case exists).

**Case #2**: The tested link shouldn’t be asserted as a member of the final plan network (implied logic case exists).

**Case #3**: An already asserted link has to be retracted from being a member of the final plan network. Also, the tested link has to be asserted as a member of the final plan network (implied logic case exists).

Case #2 is applicable if: ancestor or parent of node B is son or descendant of node A. Case #3 is applicable if: parent of node B is a parent or ancestor of node A (retract link between node B and its parent), Or if: son or descendant of node B is son of node A (retract link between node A and its parent). If case #2 or case #3 don’t apply to the current situation, then case #1 is applicable. Accordingly the tested link is to be asserted as a member of the final plan network.

The second type of conflict is referred to as “loop logic”. This type of conflict occurs if a loop is expected by asserting the tested link. Figure 8.18 depicts the different cases in which a loop exists. In general, a loop exists if node B is a parent or an ancestor of node A. Under such circumstances, the tested link shouldn’t be asserted as a member of the final plan network. Logically, the loop can be avoided by retracting any link from the
Figure 8.17: Implied Logic Cases
loop and asserting the tested link. However, since the tested link has lower priority than the already asserted links, the dynamic sequencer solves the conflict by ignoring the presence of the tested link.

**8.3.2.3. The Final Refinement Reasoning Process**

The final refinement reasoning process is the last process to be performed by the dynamic sequencer to come up with the final project plan. The input to this process is a set of schemata structures that are used to describe the status of the state-space during the reasoning process. There is one primary action (operator) that perform the transformation process in the state-space, thus moving the search from one state to another. The action is to select a link schema from the potential links in the knowledge-base. Then to check if the selected link provides any conflict with other links that are already asserted as members of the final plan network. Based on the outcome of the checking process, changes in the knowledge-base occur. These changes define a new state description in the state-space.

There are many rules in the dynamic sequencer that are concerned with this stage of the reasoning process. Some of these rules perform the actual final refinement reasoning process. Others, are included to lend support and pass control from one rule to another. That means, sending messages to activate specific rules. The following discussion outlines the steps of the refinement reasoning process. The steps are presented by the flow chart depicted in Figure 8.19.

The first step in the final refinement reasoning process is to find the different priority values that exist for the different link schemata in the knowledge-base. These values are
Figure 8.18: Loop logic Conflict Cases

In all cases:

same condition can be applied

"B" is a PARENT or ANCESTOR of "A"

⇒ "There is a LOOP"
Figure 8.19: The Refinement Reasoning Process
stored in a slot 'prio-data' of a schema called 'control-data'. The 'control-data' schema is created to maintain all the control data needed during the reasoning process. For each value in the 'prio-data' slot, a reasoning cycle is initiated and performed in sequence. The cycles proceed from a higher priority value to lower priority value. The different cycles of this stage of the reasoning process are called 'priority cycles'. Each priority cycle involves the execution of many rules. The cycles are initiated based on control data and facts that are passed from one rule to another.

The highest priority value in the 'prio-data' slot of the schema 'control-data' is considered as the 'current cycle value' for the first priority cycle. The system proceeds to test which are the link schemata that have a priority value equal to the current cycle value. If a specific link schema passes the test, the 'init-act-after' slot of the O-node schema related to the 'from-act' slot value of the link will be appended. The value of the 'to-act' slot of the link will be asserted in the 'init-act-after' slot. Also, the init-act-before slot of the O-node schema related to the 'to-act' slot value of the link will be appended to have the value of the 'from-act' slot value of the link.

The next step is to assert the fact that defines the opened-list and the closed-list. The members of the opened-list are all the activities that are related to the different O-node schemata that have no value in the 'init-act-before' slots. These activities are considered as start activities for the current priority cycle. This means that an activity is posted to the opened-list if it has no predecessors, based only on the links that have priority value equal to the current cycle value. Accordingly, for each cycle, the opened-list doesn't contain all the activities of the project. On the other hand, the closed-list is always empty at the beginning of any priority cycle.
The final refinement reasoning process proceeds by selecting the first member of the opened-list. All the links that have a priority value equal the current cycle value and start from the selected member are tested to check for implied logic or loop logic. According to the test outcomes, changes in the related link schemata are performed to reflect the outcome. This is accomplished either by asserting new links as members of the final plan network, retracting a link from the final plan network, or maintaining the current network in the knowledge-base. The process described in this paragraph is depicted in a flow chart form in Figure 8.20.

After checking all the links of the selected member from the opened-list, the member is posted to the closed-list. New members that are related to the last selected member with the checked links are posted in the opened-list. This is done only if they don’t exist in the opened-list or closed list. The outcome of the above process results in entering a new state description in the search space. The process is repeated by selecting another member of the opened-list and performing the same procedure described in the previous paragraph.

The current priority cycle is terminated when the opened-list becomes an empty list. This leads to initiating a new priority cycle with the next priority value in the 'prio-data' slot of the 'control-data' schema. The final refinement reasoning process progresses by moving through the different priority cycles. The process is terminated when all values in the 'prio-data' slot are considered (which represents the goal state of the search process).

The outcomes of the final refinement reasoning process is the final plan network for the project. The attributes of this network are defined in the 'act-before' and 'act-after' slots of the O-node schemata. These attributes represents the predecessors and successors of
Figure 8.20: Optimizing the Final Project Network
each activity. Also the link schemata contain indicators that describe whether the link is a member of the final project plan or not. If the value of the slot 'used' is 'Yes', the link is a member of the final network. Otherwise it is not a member.

Appendix C6 includes the source code listing of the rules that are involved in the final refinement reasoning process. Figure 8.21 depicts the network of the sample project introduced previously in this chapter. This network is the outcome of the preliminary logic refinement reasoning process. It is also the input for the final logic refinement reasoning process. Figure 8.22 and Figure 8.23 depicts the status of the network at different critical states of the refinement reasoning process. Figure 8.22(a) is the status of the network after the first priority cycle which has a priority value of '80'. Figure 8.22(b) is the status of the network after the second priority cycle which has a priority value of '70'. Figure 8.23(a) is the status of the network after the third priority cycle which has a priority value of '60'. Figure 8.23(b) is the final network after the last priority cycle. This network represents the final project plan generated by the dynamic sequencer.

8.4. Miscellaneous Declarative Knowledge

The dynamic sequencer knowledge-base contains miscellaneous definitions that support the schemata structure used in the reasoning process. These definitions lend support to the reasoning process by defining the nature of slot behavior of different schemata slots. Some of the miscellaneous schemata defines the parent schemata for different types of schemata introduced in this chapter.
(a) The Network After the '80' Priority Cycle

(b) The Network After the '70' Priority Cycle

*Figure 8.22: The Project Network During the Final Logic Refinement (part 1)*
(a) The Network After the '60' Priority Cycle

(b) The Final Network After the '50' Priority Cycle

*Figure 8.23: The Project Network During the Final Logic Refinement (part 2)*
A 'control-data' schema is provided in the dynamic sequencer to control the reasoning process during the final logic refinement process. The control-data contains slots that define attributes needed to pass control between rules. Also, to initiate the different priority cycles of the final refinement reasoning process. The control-data schema is also used in the process of calculating the schedule dates of the different activities. The schedule date calculation is based on an object-oriented programming approach. Section 8.5 describes this process in detail.

Appendix C7 provides a list of all miscellaneous schemata definitions needed by the dynamic sequencer. In addition, a global variable which control the salience value of different rules in the system is included in the list.

### 8.5. Schedule Calculation

The primary outcome of the reasoning process is a final project plan. This plan can be viewed as a network of activities with relationships between them. These relationships define the predecessors and successors of each activity. The next step is to calculate the schedule dates: early start, early finish, late start, and late finish. These dates are transferred to the central database to be able to generate a 'Record' file that is used in performing the visual simulation of the construction process using the WALKTHRU system.

The object-oriented programming approach is adopted to perform the schedule calculation task. The object-oriented approach is based on the concept of a message sending.
All the action in object-oriented programming comes from sending messages to objects. Message sending is a form of indirect procedure call. Instead of naming a procedure to perform an operation on an object, one sends the object a message. A selector in the message specifies the kind of operation. Objects respond to messages by their own methods (procedures) to perform operations [Stefik and Bobrow 85].

The message sending concept is used at this stage of the dynamic sequencer to send messages to activities to perform the needed operations of schedule calculation. The dynamic sequencer sends messages to the activities from an action side of rules defined to perform this task. There are two rules in the system that send messages to activities. The first rule is concerned with sending messages to activities to perform the early schedule calculation. The second rule is concerned with sending messages to activities to perform the late schedule calculation.

There are two methods defined for each activity to respond to messages. The first method is called “to-schedule-early”. The second method is called “to-schedule-late”. The procedural code of the two methods are defined by an ART function. The methods are attached to each activity via two slots called: “calc-early” and “calc-late”, respectively. When a rule sends a message to an activity, it activates the procedural code of the method defined in the slot that received the message.

The rules are designed to send messages to the activities only if all the preceding activities, in the case of early schedule rule, or the succeeding activities, in the case of late schedule rule, have been scheduled. An activity is called early scheduled, if its early start and early finish have been calculated based on the message that has been already sent to it. Similarly, an activity is called late scheduled, if its late start and late finish have been calculated based on the message that has already been sent to it. The following
discussion describes the operations involved in the early schedule rule to perform the forward pass calculation of the schedule. Similar operations are applicable in the case of the late schedule rule. However, different attributes are used to perform the backward pass calculation of the schedule.

The activity schema includes different slots that are used by the scheduling rules. Part of the activity schema structure that is involved in this process looks as follows:

(Schema ACT-1
  (calc-early to-schedule-early)
  (calc-late to-schedule-late)
  (scheduled-early No)
  (scheduled-late No)
  (early-start)
  (early-finish)
  (late-start)
  (late-finish)
  (act-dur 20))

The "calc-early" and "calc-late" are slots that contain the name of the methods to be performed if a message is sent to the activity. The "scheduled-early" and "scheduled-late" are slots that define the status of the activity regarding the scheduling process. If the slot has a value of No, that means the activity has not been scheduled yet for a certain scheduling pass (early or late). If the value is Yes, that means the activity has been scheduled for that scheduling pass. The early-start and early-finish slots contain the information regarding the early schedule of the activity. The late-start and late-finish slots contain the information regarding the late schedule of the activity. The "act-dur" slots
defines the duration of the activity. In the current version of the KNOW-PLAN prototype system, the duration of the activities are entered by the user in the central database. Refer to Chapter 7 Section 7.2.1 regarding the activity durations.

The early schedule rule performs the forward pass of the scheduling process. The first step is to list all the activities of the project in the "act-list" of the "control-data" schema. The scheduling process progresses by selecting an activity from the "act-list". If the activity is a start activity, the rule sends a "calc-early" message to the activity. Sending a message to the activity activates the "to-schedule-early" method that performs the early schedule calculation for the activity. If the activity has predecessors, the predecessor activities are checked if they have been scheduled or not. If all the preceding activities have been scheduled, the message is sent to the activity. If one or more of the preceding activity has not been scheduled, the program ignores the current activity and selects a new activity from the "act-list". If a message is sent to an activity, the schedule calculations are performed. Then the activity is retracted from the "act-list". The process continues until the "act-list" becomes empty. If the "act-list" is empty, this means all the activities of the project have been scheduled for the forward pass.

The "to-schedule-early" is the name of the method that is activated by the early schedule rule. The procedural code of this method is defined by the "to-schedule-early" ART function. The function initially assumes that the early start of the activity is "Zero". Then it selects each predecessor activity and finds its early finish. If the early finish of the predecessor activity is greater than the current early start value of the activity, the early start of the activity is modified to have the same value as the early finish of the predecessor. The operation continues until all preceding activities are checked. The last value of the early start is the value that will be stored in the "early-start" slot of the activity.
The "early-finish" of the activity is simply calculated by adding the activity duration to its early start value.

Figure 8.24 depicts a flow chart for the operations that are performed by the early schedule rule. The flow chart of the procedural code of the ART function "to-schedule-early" is also included in the figure. Figure 8.25 depicts a flow chart for the late schedule rule with its related ART function "to-schedule-late". The source code listing of the schedule calculation rules and associated ART functions (methods) are included in Appendix C8.

8.6. Creating the Schedule Interface Files

The last step to be performed by the dynamic sequencer is creating two ASCII files that contain the logic data among different activities and the schedule dates of different activities. The rule that creates these two files is called "schedule-reporting-rule". The name of the two files that are created by this rule are "logic.dat" and "schdate.dat". Chapter 9 describes these two files in more details. The source code listing of the schedule reporting rule is included in Appendix C9.
Figure 8.24: Early Schedule Rule and the Related Object Method
Figure 8.25: Late Schedule Rule and the Related Object Method
8.7. Summary of the Dynamic Sequencer Module

The dynamic sequencer module is the center piece of the KNOW-PLAN prototype system. The power of the KNOW-PLAN system comes from the capability of the dynamic sequencer of providing a reasoning process to generate construction plans. The dynamic sequencer takes as input a description of the facility to be planned as defined by extracted geometric data from a 3D computer modeling system. The dynamic sequencer progresses by using the extracted data in conjunction with geometric knowledge and other knowledge sources to come up with a project plan. The plan generation process is based on a powerful reasoning process. The reasoning process attempts to satisfy all the constraints defined by the different knowledge sources in the system based on the priority of these constraints. Figure 8.26 depicts an overall picture of the processes that are involved in the dynamic sequencer’s reasoning process. The data flow in the dynamic sequencer and the data flow out of the dynamic sequencer are schematically presented in the figure.

The dynamic sequencer is basically a KGB planning system. The sequencer adopts many AI concepts in the implementation in addition to the concepts of knowledge-based systems technology. Among these concepts are: least-commitment principle, state-space representation, and object-oriented programming approach. The dynamic sequencer is implemented using the ART toolkit which is running under a DOS environment.

The dynamic sequencer’s reasoning process is divided into two main phases. The first phase involves the reasoning process which is concerned with generating relationships (logic) among different activities of the project. These relationships are based on the
Figure 8.26: Data Flow in the Dynamic Sequencer
different constraints defined by the knowledge that resides in the different knowledge sources of the system. The second phase involves refining the generated logic to optimize the project plan network. This reasoning process is called the refinement reasoning process. Also, the second phase performs schedule calculations based on the finalized project plan.

In the first phase, many steps are involved. First, an interface with the outside environment is provided. The dynamic sequencer has many external data interfaces that enable it to read and import the data resident in the central database of the KNOW-PLAN prototype system. Then, a zone allocation rule is executed to assign each object, which is defined as an activity, to different zones and to find the zones with which the object shares a common surface. This process is based on geometric data comparison between the objects' geometric data and the zones' geometric data. The final step in this phase is the geometric reasoning process. This process is concerned with asserting potential relationships between activities based on their geometry and the classes to which they belong. The potential relationships are asserted in the knowledge-base in the form of execution facts. If other knowledge sources are included in the system, the reasoning process at this phase asserts potential relationships between activities based on the different constraints defined by these additional knowledge sources.

The second phase is mainly divided into three stages. In the first stage, a preliminary logic refinement reasoning process is performed. This reasoning process attempts to eliminate execution facts that are redundant or oppositely defined by another execution fact in the knowledge-base. The second stage is the final logic refinement reasoning process. The input to this stage are the execution facts that survive the preliminary logic refinement reasoning process. The result of the final reasoning process is the final project
plan that can be viewed as a network with activities as nodes and relationships between different activities as links. The third stage is the process of calculating the schedule dates of each activity based on the generated final project plan by the final refinement reasoning process. Also, at this stage, the files needed by the central database to be able to generate a Record file for the visual simulation construction process are created. These files define the logic and schedule dates of each activity. The process of generating these files is described in Chapter 9.

The next chapter introduces the visual simulation construction process that is performed using the WALKTHRU system. The visual simulation process is based on the generated project plan by the dynamic sequencer reasoning process.
9. The Visual Simulation Module

The primary advantage of the KNOW-PLAN prototype system over other knowledge-based planning systems is its ability to communicate with a 3D computer modeling system. The purpose of this communication is to use geometric data that resides in 3D computer models in generating project plans. Also, to visually simulate the construction process on a graphics display. The simulated construction sequence is based on the project plan generated by the dynamic sequencer's reasoning process.

The visual simulation allows for visual presentation of the construction process to better view unfavorable logic in the generated project plan. The visual simulation can achieve a greater understanding and awareness of the project by management and site personnel. During the actual construction, the simulation process can help the site personnel to visualize how and what should be constructed in a coming progress period.

This chapter introduces the fourth module of the KNOW-PLAN prototype system. This module is called the visual simulation module. The interface and communication procedures between different components of the visual simulation process are discussed.
9.1. The Central Database Postprocessor

In Chapter 6 the central database of the KNOW-PLAN prototype system is introduced as the primary user interaction environment with the KNOW-PLAN prototype system. The central database acts as the link between the different applications integrated by KNOW-PLAN. It provides the necessary interface between these applications to accomplish an efficient data exchange.

Processing within the central database is divided into two main phases. The first phase is the Preprocessor phase. The preprocessor prepares the data and knowledge needed by the dynamic sequencer to perform the plan generation reasoning process. The preprocessor has been introduced in Chapter 7. The second phase is the Postprocessor phase. The postprocessor is the component of the central database that manipulates and organizes the data needed by the visual simulation module of the KNOW-PLAN prototype system.

The primary objective of the Postprocessor phase is to read the project plan generated by the dynamic sequencer module. Then, to process the data involved in the plan to create a Record file needed by the WALKTHRU system to perform the visual simulation process. The postprocessor requires the user to input specific information needed to accomplish its objectives.

The postprocessor can be invoked in the central database by selecting the second option of the main menu "Postprocessor Data Manipulation". The postprocessor menu currently includes the following options:

9. The Visual Simulation Module
1. Input viewing parameters;
2. Read generated logic and schedule data;
3. Generate EARLY "Record" file to WALKTHRU;
4. Generate LATE "Record" file for WALKTHRU.

Figure 9.1 depicts the menus of the central database for the postprocessor phase. Sample screens are included in the figure to illustrate the data needed or created by each option.

9.1.1. Viewing Parameters

The WALKTHRU (3D Visual Simulation) system is used in the KNOW-PLAN prototype system to provide the simulation environment for the construction process. The WALKTHRU system is a graphics system that displays the objects of a 3D computer model on graphics display. The display of the 3D computer model at each frame is controlled by viewing parameters that characterize each frame of display.

A WALKTHRU Record file has to be created to perform the visual simulation of the construction process. This file is used by the Play-back routine of the WALKTHRU system to initiate a simulation session. The Record file includes parameters that define the configuration of the display at each frame of time. Time frames are called "Key Frames" in the WALKTHRU system. The structure of the Record file is described in detail in section 9.1.3.

At each key frame of the Record file, specific viewing parameters are defined. These parameters define the x, y, and z position of the body that represents the position of the user in the 3D computer model. The direction of travel, the head orientation, clipping
Figure 9.1: The Central Database Menu Structure (Postprocessor Phase)
planes, and the perspective angle of the view at each key frame are also included in the key frame parameters. The following is a short list for the viewing parameters needed to be defined at each key frame:

**Body's position:** is for control of the x, y, and z coordinates of the viewer's body in the model space.

**Direction of Travel:** is for control of the direction of the viewer's travel through the model. Two parameters define the direction of travel. These parameters are: the inclination of the viewer's body in degrees, and the heading of the body in degrees.

**Head Orientation:** is for control of the orientation of the viewer's head relative to the viewer's body. There is a horizontal pan of the head relative to the body in degrees. And there is a vertical pan of the head relative to the body in degrees.

**View Clipping:** control the distance from the viewer's body to the near and far clipping planes in model units. The near clipping plane is a plane in space that no geometric entities in front of it is displayed. The far clipping plane is a plane in space that no geometric entities behind it are displayed.

**Angle of perspective:** is for control of the angle of perspective to be used in displaying the 3D computer model (zooming).

Accordingly, the user has to define the viewing parameters to be used in the Record file. The central database provides a data entry form to be filled by the user to define the viewing parameters for a specific Record file. These parameters are stored in a database.
file called "walkdat.dbf". The user is required to provide only one set of viewing parameters. This set is used by all key frames in the Record file. Figure 9.1 depicts a sample set of viewing parameters that is used in the visual simulation process of the culvert project.

A future version of the system will allow the user to define different sets of viewing parameters to be used by different key frames of the same Record file. This will allow the user to view the simulation process from different locations in the same session. Also, the viewing parameters can be extracted automatically from an already existing Record file of the 3D computer model. This feature will eliminate the necessity for user input.

9.1.2. Read Generated Logic and Schedule Data

The final outcome of the dynamic sequencer reasoning process is a project plan. This plan includes the data that defines a list of the project's activities and their relationships. Also, it includes the schedule dates of each activity. This data is needed by the postprocessor to create the Record file. Accordingly, the data has to be transferred from the Knowledge-base of the dynamic sequencer to the central database. The dynamic sequencer has a rule that generates two ASCII files to be used in the data exchange process.

The first ASCII file, generated by the schedule reporting rule, is the schedule file. This file includes a list of activities and their schedule dates. The data included in the schedule file are read by the postprocessor and mapped to the activity database file in the central database. This data is later used in creating the Record files for the visual simulation process.
The second ASCII file is the logic file. This file includes a list of the relationships (logic) between the different activities of the project. These relationships are based on the generated project plan by the dynamic sequencer. This file is not required to create the Record files. However, the data included in this file can be used to provide an interface with any CPM processor that can read ASCII files. The file’s data is mapped to a new database file called logic database file. This data file consists of two fields: preceding activity and succeeding activity.

9.1.3. Record File Structure

A Record file in the WALKTHRU system contains a sequence of one or more key frames. Each key frame contains view configuration data and optional object orientation data about the 3D computer model at a particular moment in time. The data for one object is contained on one line. The structure of a Record file with the definition of each attribute of a key frame is depicted in Figure 9.2.

Each key frame starts with an attribute that defines the delta time since the last key frame in units of time. The viewing parameters for the key frame are listed at the top of the key frame. The display bit masks of a key frame indicates which objects are displayed in the key frame. There is one bit mask for every 32 objects in the 3D computer model. If an object is displayed in the key frame, the bit related to the object is set to (1). If the object is not displayed, then the bit is set to (0). The bit number of an object equals to the object number minus (1) which defines the bit’s position in the binary format. Therefore the first object is related to the first bit at position (0).
Figure 9.2: The WALKTHRU Record File Structure
Assume there are 6 objects in the model. If object#1, object#3, object#5, and object#6 are displayed. And object#2 and object#4 are not displayed. Then one display bit mask is required. The display bit mask will have a (110101) binary value which is the number (53). The decimal value of the display bit mask is calculated by transforming its binary value to the decimal value.

A list of all objects which are displayed with their position and orientation are listed in the key frame each on one line. Figure 9.1 includes a sample key frame with its attribute values. A Record file contains a sequence of key frames each contains view configuration data. The time between the key frames is defined by the delta-time attribute of the key frames.

9.1.4. Creating a Record File

The KNOW-PLAN prototype system utilizes the Record and Replay functions of the WALKTHRU system to perform the visual simulation of the construction process. The postprocessor uses the activities’ schedule data to create the different key frames of Record files. The postprocessor creates a key frame in the record file whenever an activity starts or an activity finish. The result is a record file that contains a sequence of key frames that represents the time at which each activity is to be displayed during the visual simulation process.

The postprocessor menu has two options to create Record files. The first option creates a Record file based on the calculated early schedule of the project. The second option creates a Record file based on the late schedule of the project. In the KNOW-PLAN prototype system, an object is displayed in a current key frame whenever its start is after
the cumulative time from the first key frame to the current key frame during the simulation process. Therefore, the visual simulation process displays objects based on their start time.

The Record file creation process is divided into two main steps. The first step creates a frame time database file called "frmtime.dbf". This file contains records defining the time parameters of the different key frames required to perform the simulation process. This step is initiated when the user selects a Record file creation option from the postprocessor menu. The start time and finish time of each activity are used to create the potential time records in the frame time database file. The result is a list of time records that defines the critical dates when changes in the project status occur. A change in the project status is defined as the time at which an activity is started or an activity is finished.

The different time records are sorted in an ascending order. The sorted time records in the frmtime.dbf represents the time data of the key frames of the Record file to be created. For each key frame, the value of its display bit masks are calculated to reflect the activities that are to be displayed at that specific key frame.

The second step in the Record file creation process is to create the ASCII Record file. The postprocessor uses the data contained in the activity.dbf, walkdat.dbf, and the frmtime.dbf to create this ASCII file.

The postprocessor starts by opening a text file. It reads the first record in the frmtime.dbf. A key frame is initiated with a delta time equal to (0). The postprocessor writes the delta time, the viewing parameters, and the values of the display bit masks of the key frame to the opened text file.
The second key frame is now selected. The time difference between this key frame and the previous key frame is calculated and written to the text file as the "delta_time" value for the current key frame. The remaining attribute values of the current key frame are written to the text file. The process continues until all the key frames of the frmtime.dbf are written to the text file. The result is an ASCII Record file which contains the different key frames that define the view configuration of the visual simulation process.

Figure 9.3 depicts a flow chart for the procedure described in the first step of the Record file creation process. Figure 9.4 depicts a flow chart for the procedure described in the second step of the creation process. A sample Record file for the culvert project is presented in Chapter 10.

9.2. The Visual Simulation Process

The 3D visual simulation capability of the WALKTHRU system is incorporated in the KNOW-PLAN prototype system to simulate the construction process. The simulation process provides a visual tool to present the sequence of executing the project activities to the user. The activities of the project are defined by the 3D objects of the 3D computer model of the project. Displaying the objects on graphics display according to time scale, that is based on the generated logic by the dynamic sequencer, enhances the construction planning process.

The visual simulation of the construction process is performed using the Record file that represents the sequencing of the project's objects. The simulation is performed by re-
Figure 9.3: Creating the Key Frame Data for the Record File
Figure 9.4: Generating an ASCII Record File
playing the Record file using the Replay routine of the WALKTHRU system. The re-
playing process is based on the same concept used in VCR's. The Record file represents
different frames which describe the view configuration of the display at different intervals
of time. The system linearly interpolates the following values between frames:

1. X, Y, and Z positions
2. heading and inclination
3. horizontal and vertical pan
4. perspective angle
5. object motion (if it is included in the Record file).

Selecting the Replay option from the WALKTHRU menu brings the Replay menu on
the screen. The menu has the following functions:

File -
Frames/Seconds -> 10
Control-Obj/Path/View
Once Through
Start
Exit

The file function prompts the user for the name of the Record file to be used in the re-
play process. The frames/seconds lets the user set the number of frames per second at
which to replay the motion. The frame per second value is the number of intermediate
frames to be calculated for each unit of time in the Record file. The control function
defines the mode of motion. The Obj/Path/View mode replays everything recorded in the
Record file. The once through option in the Replay menu allows replaying the Record file once through. The start option initiates the visual simulation (Replay) process.

The result is a visual simulation of the construction process based on the generated sequence and calculated schedule by the dynamic sequencer. The user can control the time of the simulation process by using the frames/seconds option in the Replay menu. The KNOW-PLAN prototype system provides visual simulation sessions for the entire project period from start to finish. A current research at Virginia Tech has been initiated to provide more flexibility in viewing (simulating) the construction process. Refer to [Skolnick, Morad and Beliveau 90] for the description of this research.

The visual simulation of the construction process provides construction planners a tool for a greater understanding and awareness of the actual construction process. By simulating the construction process before starting the project, out of sequence activities can be detected. Many “What-If?” planning scenarios can be conducted based on the outcome of the simulation process.

Based on the outcomes of the visual simulation process, new rules and execution facts can be added to the knowledge-base of the dynamic sequencer. This additional knowledge can be considered as an additional knowledge source in the knowledge-base. The reasoning process can be initiated again to provide new execution facts to the knowledge-base. These new execution facts can be viewed as an additional network in the knowledge-base. The refinement reasoning process can be executed to combine this network with already existing networks from the previous reasoning process. The result is a modified final project plan that reflects the new added constraints. The visual simulation process can be repeated to simulate the modified project plan.

9. The Visual Simulation Module
The KNOW-PLAN prototype system does not provide recycling to the overall reasoning process of the dynamic sequencer to incorporate the additional network described in the previous paragraph. However, a minor modification to the system can be implemented to consider the recycling of the reasoning process. The recycling of the reasoning process is the fourth stage in the KNOW-PLAN Model described in Chapter 5.

9.3. Summary of the Visual simulation Module

The visual simulation module is the fourth module in the KNOW-PLAN prototype system. It is the module that performs the visual simulation of the construction process based on the generated sequence by the dynamic sequencer module. This module provides a practical approach for representing construction plans on a simple and realistic fashion. The strength of the KNOW-PLAN prototype system comes from its ability of integrating knowledge-based system technology with 3D computer modeling technology to generate and the visually simulate project plans.

The visual simulation modules takes as input two ASCII files that are generated by the dynamic sequencer. The primary component of the visual simulation module is the postprocessor of the central database. The postprocessor is a processing environment that create the necessary data needed to perform the visual simulation process. The postprocessor reads the ASCII files generated by the dynamic sequencer. The postprocessor creates new database files or modifies existing database files based on the data included in the ASCII files. Then, the postprocessor starts the process of creating the Record file.
The Record file is the basic element of the interface between the central database and the WALKTHRU (3D visual simulation) system. The Record files are used by the Replay routine of the WALKTHRU to initiate a visual simulation session. The Record files are created by the postprocessor based on the data included in the different database files in the central database.

The visual simulation of the construction process provides a simulation tool to present the sequence of installing the objects (activities) of the project on graphics display. The simulation process shows when each object should be installed based on the early or late schedule of the project. The simulation process is viewed according to a time scale which is controlled by the user.

The visual simulation process has many benefits. In general, it enhances the planning process by providing efficient tools to present construction plans to the users. The outcome of the visual simulation process can be an input to the dynamic sequencer to incorporate new facts or constraints in the project plan. Figure 9.5 depicts an overall picture of the data flow in the visual simulation module.
Figure 9.5: Data Flow in the Visual Simulation Module
10. Sample Culvert Project

This chapter presents the solution for the planning problem of the sample culvert project introduced in Chapter 6. The solution is based on utilizing the KNOW-PLAN prototype system to solve the problem. A sensitivity analysis is included to assess the impact of changing some parameters, such as data or knowledge, on the overall construction plan of the project.

Five cases are presented in the following sections. The first case provides the basis for other cases. The first case provides a reasonable construction plan for the culvert project. The second case assesses the impact of changing the system's knowledge about the zoning of the project on the construction plan. The third case assesses the impact of changing the duration of the activities on the construction plan. The fourth case assesses the impact of changing the user's defined knowledge on the construction plan. The final case assesses the impact of changing the knowledge about the classes of the activities on the construction plan.
For each case, the knowledge and data provided in the case are depicted. The generated logic network are also presented. The schedule dates calculated based on the generated plan with the associated Record file, to be used in the visual simulation process, are presented.

10.1. CASE 1: Hypothetical Project Knowledge

The first case presents the basis for other cases represented in this chapter. Figure 10.1 depicts the different knowledge and data needed to be entered by the user of the KNOW-PLAN prototype system. Figure 10.1 (a) lists the different activities of the culvert project with their durations, associated object numbers, and the classes to which they are related.

Figure 10.1 (b) lists the geometric data of the different zones of the project. The project in this case is divided into three zones. The first zone in the west zone which contains all the west structural elements and the west barrel. The second zone includes only the center barrel. The third zone contains all the east structural elements and the east barrel.

Figure 10.1 (c) lists all the connection data between the different activities. All the connections are assumed to be of the type "tg" which represents the default type of connection among objects.

Figure 10.1 (d) defines the different classes to which the activities are related. For each class the direction of installation in the different coordinate directions is defined. Also, the priorities of installation and the description of the class is included. Figure 10.1 (e) lists the dominating class relations between the different classes of the project.
Finally, user's defined knowledge is presented. In this case the user is defining a constraint to be considered during the generation of the construction plan. This constraint is defined in the form of an execution fact. The execution fact asserts that the East Barrel (ACT_3) should be installed before the Center Barrel (ACT_2) with a priority value of 60.

Figure 10.2 depicts the logic generated by the dynamic sequencer based on the input knowledge and data to the system. Figure 10.3 (a) lists the schedule dates of the project activities based on the generated logic. In Figure 10.3 the early and late Record files that are generated by the postprocessor of the central database are depicted.

10.2. CASE 2: Changing Zoning Data

This case assesses the impact of changing the definition of the zone break-down of the project on the construction plan generated by the first case. In this case, the whole project is defined as one zone. Figure 10.4 depicts the knowledge and data needed to run this case. The only difference between Figure 10.4 and Figure 10.1 is in the zones' geometric data.

The dynamic sequencer generates the same network presented in the first case with the same schedule dates and Record files. Accordingly, The zoning definition doesn't affect the logic of the construction plan nor the schedule dates of the activities. The impact of changing the zones parameters will affect the computing speed and the number of rules fired during the reasoning process.
Figure 10.1: Sample Project Input Knowledge and Data (Case #1)
Figure 10.2: Sample Culvert Project Network (Case #1)
(a) Schedule Dates File

<table>
<thead>
<tr>
<th>Activity</th>
<th>BS</th>
<th>LS</th>
<th>EF</th>
<th>LF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_23</td>
<td>20</td>
<td>29</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_22</td>
<td>20</td>
<td>29</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_20</td>
<td>20</td>
<td>16</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>ACT_19</td>
<td>20</td>
<td>29</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_18</td>
<td>20</td>
<td>29</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_17</td>
<td>20</td>
<td>29</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_16</td>
<td>16</td>
<td>16</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>ACT_15</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>ACT_14</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>ACT_13</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_12</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>ACT_11</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>ACT_10</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_9</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_8</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_7</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_6</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_5</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_4</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_3</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_2</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>ACT_1</td>
<td>25</td>
<td>25</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

(b) Early Record File

<table>
<thead>
<tr>
<th>0.000</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>393216</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>987136</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>5242520</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>5242520</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>6291451</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>8388607</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>8388607</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>8388607</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Late Record File

Figure 10.3: Sample Project Schedule and Record Data (case #1)
(a) Activity Data

<table>
<thead>
<tr>
<th>Activity Name</th>
<th>Activity Duration</th>
<th>Associated Object No.</th>
<th>Associated Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_1</td>
<td>16</td>
<td>19</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_2</td>
<td>16</td>
<td>20</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_3</td>
<td>16</td>
<td>18</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_4</td>
<td>4</td>
<td>21</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_5</td>
<td>4</td>
<td>22</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_6</td>
<td>4</td>
<td>1</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_7</td>
<td>4</td>
<td>2</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_8</td>
<td>4</td>
<td>3</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_9</td>
<td>4</td>
<td>7</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_10</td>
<td>4</td>
<td>6</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_11</td>
<td>5</td>
<td>15</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_12</td>
<td>5</td>
<td>5</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_13</td>
<td>4</td>
<td>9</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_14</td>
<td>5</td>
<td>4</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_15</td>
<td>5</td>
<td>8</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_16</td>
<td>4</td>
<td>13</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_17</td>
<td>4</td>
<td>14</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_18</td>
<td>4</td>
<td>23</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_19</td>
<td>4</td>
<td>12</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_20</td>
<td>4</td>
<td>17</td>
<td>C2</td>
</tr>
<tr>
<td>ACT_21</td>
<td>4</td>
<td>11</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_22</td>
<td>4</td>
<td>10</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_23</td>
<td>4</td>
<td>16</td>
<td>C5</td>
</tr>
</tbody>
</table>

(b) Zones' Geometric Data

<table>
<thead>
<tr>
<th>Zone Name</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>-55</td>
<td>220</td>
<td>-30</td>
</tr>
</tbody>
</table>

(c) Connection Data

All connections are defined as the default value "tg". The connections defined between activities are:

- ACT_1 with 16, 17, 18, 19, 10, 11, 12
- ACT_3 with 20, 21, 22, 23, 13, 14, 15
- ACT_16 with 11, 12, 17, 18, 19
- ACT_20 with 14, 15, 21, 22, 23
- ACT_10 with 5
- ACT_11 with 4
- ACT_12 with 6
- ACT_13 with 8
- ACT_14 with 9
- ACT_15 with 7

(d) Class Definition Data

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Direction of Installation</th>
<th>Priority of Installation</th>
<th>1st</th>
<th>2nd</th>
<th>Class Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>C2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>C3</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>C4</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>C5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

(e) Domination Class Data

<table>
<thead>
<tr>
<th>Class No 1</th>
<th>Class No 2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
<td>C5</td>
<td>C2</td>
</tr>
<tr>
<td>C1</td>
<td>C5</td>
<td>C1</td>
</tr>
<tr>
<td>C1</td>
<td>C3</td>
<td>C1</td>
</tr>
<tr>
<td>C3</td>
<td>C4</td>
<td>C3</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>C2</td>
</tr>
</tbody>
</table>

User Defined Knowledge

EXECUTE ACT_3 ACT_2 USER 60
*Install the East barrel before the Center Barrel*

Figure 10.4: Sample Project Input Knowledge and Data (Case #2)
10.3. CASE 3: Changing the Duration of the Activities

This case assesses the impact of changing the duration of the activities on the generated logic and the schedule dates of the activities. The current version of the KNOW-PLAN prototype system doesn’t use the activity duration in the reasoning process. Future extensions are expected to include the activity duration in the reasoning process. Figure 10.5 depicts the knowledge and data needed to run this case. The only difference between Figure 10.5 and Figure 10.1 is in the activity list. Figure 10.5 (a) depicts the activity list of the current case. The activity durations are different than the activity durations listed in Figure 10.1 (a).

The dynamic sequencer generates the same network presented in the first case. However, the schedule dates and Record files are different. Accordingly, the Duration of the activities doesn’t affect the logic of the construction plan, but it affects the schedule dates of the activities. Figure 10.6 depicts the generated schedule dates and Record files in this case. The Record files presented in the figure don’t include all the key frames of the generated Record files because of space limitations.

10.4. CASE 4: Changing User’s Defined Knowledge

In the first case, an execution fact is defined to assert the execution of the East Barrel (ACT_3) before the Center Barrel (ACT_2). In the fourth case, the impact of ignoring the user’s defined execution fact is assessed. Figure 10.7 depicts all the knowledge and data needed to run this case. The only difference between this figure and Figure 10.1 is in the definition of the user’s defined knowledge.
### Activity Data

<table>
<thead>
<tr>
<th>Activity Name</th>
<th>Activity Duration</th>
<th>Associated Object No.</th>
<th>Associated Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_1</td>
<td>14</td>
<td>19</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_2</td>
<td>8</td>
<td>20</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_3</td>
<td>8</td>
<td>18</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_4</td>
<td>6</td>
<td>21</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_5</td>
<td>3</td>
<td>22</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_6</td>
<td>3</td>
<td>1</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_7</td>
<td>5</td>
<td>2</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_8</td>
<td>3</td>
<td>3</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_9</td>
<td>5</td>
<td>7</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_10</td>
<td>6</td>
<td>6</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_11</td>
<td>3</td>
<td>15</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_12</td>
<td>5</td>
<td>5</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_13</td>
<td>4</td>
<td>9</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_14</td>
<td>3</td>
<td>4</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_15</td>
<td>5</td>
<td>8</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_16</td>
<td>5</td>
<td>13</td>
<td>C2</td>
</tr>
<tr>
<td>ACT_17</td>
<td>7</td>
<td>14</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_18</td>
<td>4</td>
<td>23</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_19</td>
<td>5</td>
<td>12</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_20</td>
<td>3</td>
<td>17</td>
<td>C2</td>
</tr>
<tr>
<td>ACT_21</td>
<td>6</td>
<td>11</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_22</td>
<td>7</td>
<td>10</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_23</td>
<td>4</td>
<td>16</td>
<td>C5</td>
</tr>
</tbody>
</table>

### Zones' Geometric Data

<table>
<thead>
<tr>
<th>Zone Name</th>
<th>X min</th>
<th>X max</th>
<th>Y min</th>
<th>Y max</th>
<th>Z min</th>
<th>Z max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>-55</td>
<td>55</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z2</td>
<td>55</td>
<td>110</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z3</td>
<td>110</td>
<td>220</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>

### Connection Data

All connections are defined as the default value "tg". The connections defined between activities are:

- ACT_1 with 16, 17, 18, 19, 10, 11, 12
- ACT_3 with 20, 21, 22, 23, 13, 14, 15
- ACT_16 with 11, 12, 17, 18, 19
- ACT_20 with 14, 15, 21, 22, 23
- ACT_10 with 5
- ACT_11 with 4
- ACT_12 with 6
- ACT_13 with 8
- ACT_14 with 9
- ACT_15 with 7

### Class Definition Data

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Direction of Installation</th>
<th>Priority of Installation</th>
<th>Dominating Class</th>
<th>Class Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>yes yes yes</td>
<td>X  Y</td>
<td>barrels</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>yes yes yes</td>
<td>X  Y</td>
<td>headwalls</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>yes yes no</td>
<td>Z  X</td>
<td>wrapwalls and slabs</td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>yes yes no</td>
<td>Z  X</td>
<td>transition walls and slabs</td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>yes yes yes</td>
<td>X  Y</td>
<td>Headwall supports</td>
<td></td>
</tr>
</tbody>
</table>

### Domination Class Data

<table>
<thead>
<tr>
<th>Class</th>
<th>Dominating No 1</th>
<th>Dominating No 2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>C2</td>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
<td>C5</td>
<td>C2</td>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
<td>C5</td>
<td>C1</td>
<td>C1</td>
</tr>
<tr>
<td>C4</td>
<td>C3</td>
<td>C1</td>
<td>C1</td>
</tr>
<tr>
<td>C5</td>
<td>C2</td>
<td>C3</td>
<td>C2</td>
</tr>
</tbody>
</table>

### User Defined Knowledge

Form: Execution Fact

(EXECUTE ACT_3 ACT_2 USER 60)

"Install the East barrel before the Center Barrel"

**Figure 10.5: Sample Project Input Knowledge and Data (Case #3)**
### Schedule Dates File

<table>
<thead>
<tr>
<th>Activity</th>
<th>ES</th>
<th>LS</th>
<th>EF</th>
<th>LF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_23</td>
<td>11</td>
<td>20</td>
<td>15</td>
<td>33</td>
</tr>
<tr>
<td>ACT_22</td>
<td>11</td>
<td>26</td>
<td>18</td>
<td>33</td>
</tr>
<tr>
<td>ACT_21</td>
<td>11</td>
<td>27</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>ACT_20</td>
<td>8</td>
<td>17</td>
<td>11</td>
<td>33</td>
</tr>
<tr>
<td>ACT_19</td>
<td>19</td>
<td>28</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>ACT_18</td>
<td>19</td>
<td>29</td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td>ACT_17</td>
<td>19</td>
<td>26</td>
<td>26</td>
<td>33</td>
</tr>
<tr>
<td>ACT_16</td>
<td>14</td>
<td>14</td>
<td>19</td>
<td>33</td>
</tr>
<tr>
<td>ACT_15</td>
<td>11</td>
<td>20</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>ACT_14</td>
<td>11</td>
<td>22</td>
<td>14</td>
<td>25</td>
</tr>
<tr>
<td>ACT_13</td>
<td>16</td>
<td>26</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>ACT_12</td>
<td>19</td>
<td>19</td>
<td>24</td>
<td>30</td>
</tr>
<tr>
<td>ACT_11</td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>ACT_10</td>
<td>24</td>
<td>24</td>
<td>30</td>
<td>24</td>
</tr>
<tr>
<td>ACT_9</td>
<td>14</td>
<td>25</td>
<td>19</td>
<td>30</td>
</tr>
<tr>
<td>ACT_8</td>
<td>21</td>
<td>30</td>
<td>24</td>
<td>30</td>
</tr>
<tr>
<td>ACT_7</td>
<td>16</td>
<td>25</td>
<td>21</td>
<td>33</td>
</tr>
<tr>
<td>ACT_6</td>
<td>24</td>
<td>27</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>ACT_5</td>
<td>30</td>
<td>30</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>ACT_4</td>
<td>22</td>
<td>24</td>
<td>28</td>
<td>33</td>
</tr>
<tr>
<td>ACT_3</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>ACT_2</td>
<td>14</td>
<td>25</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>ACT_1</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>33</td>
</tr>
</tbody>
</table>

### Early Record File

<table>
<thead>
<tr>
<th></th>
<th>0.000</th>
<th>89.500</th>
<th>259.250</th>
<th>217.233</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>262144</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>89.500</td>
<td>259.250</td>
<td>217.233</td>
</tr>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>393216</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>89.500</td>
<td>259.250</td>
<td>217.233</td>
</tr>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>997312</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>89.500</td>
<td>259.250</td>
<td>217.233</td>
</tr>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>6291451</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>89.500</td>
<td>259.250</td>
<td>217.233</td>
</tr>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>8388607</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>89.500</td>
<td>259.250</td>
<td>217.233</td>
</tr>
<tr>
<td></td>
<td>-39.600</td>
<td>-91.600</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>122.500</td>
<td>45.000</td>
<td></td>
</tr>
<tr>
<td>8388607</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Late Record File

|         | 1     | 89.500 | 259.250 | 217.233 |
|         | -39.600 | -91.600 | 0.000  | 0.000  |
|         | 1.000  | 122.500 | 45.000 |         |
|         | 6291455 | 0 |     |         |
|         | 2     | 89.500 | 259.250 | 217.233 |
|         | -39.600 | -91.600 | 0.000  | 0.000  |
|         | 1.000  | 122.500 | 45.000 |         |
|         | 8388607 | 0 |     |         |
|         | 3     | 89.500 | 259.250 | 217.233 |
|         | -39.600 | -91.600 | 0.000  | 0.000  |
|         | 1.000  | 122.500 | 45.000 |         |
|         | 8388607 | 0 |     |         |

*Figure 10.6: Sample Project Schedule and Record Data (Case #3)*

![Figure 10.6: Sample Project Schedule and Record Data (Case #3)](image-url)
Figure 10.8 depicts the generated logic network based on the knowledge and data defined in this case. The generated network is different from the network generated in the first case. In Figure 10.8, the subnetwork of the east side of the project is executed after the center barrel while in Figure 10.2 the east side activities has nothing to do with the center barrel activity. Figure 10.9 depicts the generated schedule dates and Record files. The Record files in the figure include all the key frames generated by the system.

10.5. CASE 5: Changing the Class Knowledge Definition

This case assesses the impact of changing the knowledge and data about the project’s classes on the generated construction plan. In this case, all the activities are assumed to be related to one class called C1: “Culvert Structure”. The direction of installation of this class is from left to right, forward to backward, and from bottom to top. No connection data or dominating class relations is needed in this case. However, one record of each type should be defined in order to avoid running errors. The defined records will not be used in the reasoning process. The user’s defined execution fact used in the first case is also used in this case. Figure 10.10 depicts the knowledge and data input for this case.

Figure 10.11 depicts the generated logic network based on this case. It is clear that the logic in this network is completely different from the logic generated in the first case. The class definitions are the primary factors in the reasoning process of the KNOW-PLAN prototype system. Changing the class definitions or the association of activities to classes will dramatically affect the generated construction plan.
### (a) Activity Data

<table>
<thead>
<tr>
<th>Activity Name</th>
<th>Activity Duration</th>
<th>Associated Object No.</th>
<th>Associated Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_1</td>
<td>16</td>
<td>19</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_2</td>
<td>16</td>
<td>20</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_3</td>
<td>16</td>
<td>18</td>
<td>C1</td>
</tr>
<tr>
<td>ACT_4</td>
<td>4</td>
<td>21</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_5</td>
<td>4</td>
<td>22</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_6</td>
<td>4</td>
<td>1</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_7</td>
<td>4</td>
<td>2</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_8</td>
<td>4</td>
<td>3</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_9</td>
<td>4</td>
<td>7</td>
<td>C4</td>
</tr>
<tr>
<td>ACT_10</td>
<td>4</td>
<td>6</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_11</td>
<td>5</td>
<td>15</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_12</td>
<td>5</td>
<td>5</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_13</td>
<td>5</td>
<td>9</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_14</td>
<td>5</td>
<td>4</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_15</td>
<td>5</td>
<td>8</td>
<td>C3</td>
</tr>
<tr>
<td>ACT_16</td>
<td>4</td>
<td>13</td>
<td>C2</td>
</tr>
<tr>
<td>ACT_17</td>
<td>4</td>
<td>14</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_18</td>
<td>4</td>
<td>23</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_19</td>
<td>4</td>
<td>12</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_20</td>
<td>4</td>
<td>17</td>
<td>C2</td>
</tr>
<tr>
<td>ACT_21</td>
<td>4</td>
<td>11</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_22</td>
<td>4</td>
<td>10</td>
<td>C5</td>
</tr>
<tr>
<td>ACT_23</td>
<td>4</td>
<td>16</td>
<td>C5</td>
</tr>
</tbody>
</table>

### (b) Zones' Geometric Data

<table>
<thead>
<tr>
<th>Zone Name</th>
<th>X min</th>
<th>X max</th>
<th>Y min</th>
<th>Y max</th>
<th>Z min</th>
<th>Z max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>-55</td>
<td>55</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z2</td>
<td>55</td>
<td>110</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z3</td>
<td>110</td>
<td>220</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>

### (c) Connection Data

All connections are defined as the default value "tg". The connections defined between activities are:

- ACT_1 with ACT_7, ACT_8, ACT_9, ACT_10, ACT_11, ACT_12, ACT_13, ACT_14, ACT_15, ACT_16
- ACT_3 with ACT_4, ACT_5, ACT_6, ACT_7, ACT_8, ACT_9, ACT_10, ACT_11, ACT_12, ACT_13, ACT_14, ACT_15, ACT_16
- ACT_16 with ACT_17, ACT_18, ACT_19, ACT_20, ACT_21
- ACT_20 with ACT_21, ACT_22, ACT_23
- ACT_10 with ACT_11
- ACT_12 with ACT_13
- ACT_14 with ACT_15

### (d) Class Definition Data

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Direction of Installation</th>
<th>Priority of Installation</th>
<th>Dominating Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>C1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>C2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>C3</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>C4</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>C5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

### (e) Domination Class Data

<table>
<thead>
<tr>
<th>Class</th>
<th>Class</th>
<th>Dominating No 1</th>
<th>Dominating No 2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>C1</td>
<td>C1</td>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
<td>C5</td>
<td>C2</td>
<td>C2</td>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
<td>C4</td>
<td>C3</td>
<td>C3</td>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
<td>C3</td>
<td>C4</td>
<td>C4</td>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
<td>C3</td>
<td>C5</td>
<td>C5</td>
<td>C5</td>
</tr>
</tbody>
</table>

### User Defined Knowledge

**NO USER DEFINED KNOWLEDGE**

---

**Figure 10.7: Sample Project Input Knowledge and Data (Case #4)**

249
Figure 10.8: Sample Culvert Project Network (Case #4)
Figure 10.9: Sample Project Schedule and Record Data (Case #4)
Figure 10.12 depicts the schedule dates and Record files generated by the dynamic sequencer in this case. In the figure, the Record files presented don't contain all the key frames generated by the system.
### (a) Activity Data

<table>
<thead>
<tr>
<th>Activity Name</th>
<th>Activity Duration</th>
<th>Associated Object No.</th>
<th>Associated Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT_1</td>
<td>16</td>
<td>19</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_2</td>
<td>16</td>
<td>20</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_3</td>
<td>16</td>
<td>18</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_4</td>
<td>4</td>
<td>21</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_5</td>
<td>4</td>
<td>22</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_6</td>
<td>4</td>
<td>1</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_7</td>
<td>4</td>
<td>2</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_8</td>
<td>4</td>
<td>3</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_9</td>
<td>4</td>
<td>7</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_10</td>
<td>4</td>
<td>6</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_11</td>
<td>5</td>
<td>15</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_12</td>
<td>5</td>
<td>5</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_13</td>
<td>4</td>
<td>9</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_14</td>
<td>5</td>
<td>4</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_15</td>
<td>5</td>
<td>8</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_16</td>
<td>4</td>
<td>13</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_17</td>
<td>4</td>
<td>14</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_18</td>
<td>4</td>
<td>23</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_19</td>
<td>4</td>
<td>12</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_20</td>
<td>4</td>
<td>17</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_21</td>
<td>4</td>
<td>11</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_22</td>
<td>4</td>
<td>10</td>
<td>Cl</td>
</tr>
<tr>
<td>ACT_23</td>
<td>4</td>
<td>16</td>
<td>Cl</td>
</tr>
</tbody>
</table>

### (b) Zones' Geometric Data

<table>
<thead>
<tr>
<th>Zone</th>
<th>X min</th>
<th>X max</th>
<th>Y min</th>
<th>Y max</th>
<th>Z min</th>
<th>Z max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>-55</td>
<td>55</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z2</td>
<td>55</td>
<td>110</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>Z3</td>
<td>110</td>
<td>220</td>
<td>-30</td>
<td>55</td>
<td>0</td>
<td>30</td>
</tr>
</tbody>
</table>

### (c) Connection Data

All connections are defined as the default value "tg". The connections defined between activities are:

**No Connection Data is needed**

"a single record is needed to avoid run errors"

### (d) Class Definition Data

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Direction of Installation</th>
<th>Priority of Installation</th>
<th>1st</th>
<th>2nd</th>
<th>Class Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cl</td>
<td>X Y Z</td>
<td>1st 2nd</td>
<td></td>
<td></td>
<td>Culvert Structure</td>
</tr>
</tbody>
</table>

### (e) Domination Class Data

No Dominating Class Relations is Defined

"a single record is defined to avoid run errors"

### User Defined Knowledge

EXECUTE ACT_3 ACT_2 USER 60

"Install the East barrel before the Center Barrel"

**Figure 10.10: Sample Project Input Knowledge and Data (Case #5)**
Figure 10.11: Sample Culvert Project Network (Case #5)
Figure 10.12: Sample Project Schedule and Record Data (Case #5)
11. Summary, Recommendations and Conclusion

The research presented in this dissertation provides a new approach for solving the planning problem. This approach is based on utilizing geometric-based data in the reasoning process of providing project plans. The research integrates capabilities of advanced computer technologies, both in hardware and software. The product of such an integration is an advanced system that can provide a solution for the planning problem. The research provides a model for what a developed working system would look like.

A KNOW-PLAN prototype system has been developed to prove that geometric-based system can provide for a generic dynamic sequencing system. The KNOW-PLAN prototype system is a significant effort in the overall dissertation.

This chapter provides a summary and conclusion for the dissertation. The KNOW-PLAN prototype system is summarized. The benefits of implementing the KNOW-PLAN model in practical construction planning applications are outlined. Recommendations and directions for future work to meet the overall objectives of the KNOW-PLAN model are listed. Finally conclusions regarding the research are provided.
11.1. Summary of the KNOW-PLAN Prototype System

The KNOW-PLAN prototype system is a partial implementation of the proposed KNOW-PLAN model. It consists of the crucial components of the proposed model as they are implemented by this research. The current version of the system provides the basis for implementing the overall proposed model in the future. The implementation effort concentrates on the development of the components which define the concepts of the KNOW-PLAN model. The center piece of the KNOW-PLAN prototype system is the dynamic sequencer. The dynamic sequencer is a knowledge and geometric based (KGB) system that performs the automatic generation of the project plan primarily based on the geometric data and other knowledge available in the knowledge-base of the system. A user interaction subsystem is developed to provide the interface between the 3D computer modeling system and the KGB system. The KNOW-PLAN prototype system is divided into four modules. The developed modules are:

1. the 3D computer modeling module;
2. the knowledge and data module;
3. the dynamic sequencing module;
4. the visual simulation module.

Figure 11.1 depicts an overall picture of the KNOW-PLAN prototype system as it is currently implemented.

The first module in the KNOW-PLAN prototype system is concerned with the generation of a 3D computer model of the project. Chapter 6 provides the details of this task. The interface between different CAD systems and the WALKTHRU system, used as a
**3D Computer Modeling Module**

- User
- CAD Systems
- 3D Computer Modeling
- Extracted Geometric Data
- ASCII
- Central Database Files
- POSTPROCESSOR
  - Read Generated Logic
  - Manipulate Schedule
  - Generate Record File
  - WALKTHRU Model File
  - WALKTHRU Simulation Module
  - User

**Knowledge and Data Module**

- User
- Optimizers
- Activity Data
- Class Data
- Zoning Data
- Connection Data
- Activity/Class Data
- Dominating Class Data
- Special Class Rel Data
- ASCII
- External Data Interface Files
- External Data Interface
  - Zone Allocation
  - Geometric Reasoning
  - Other Logic Assertion
- Preliminary Logic Refinement
- Final Logic Refinement
- Schedule Calculation
- Generating Logic Files
- Knowledge base
  - FACTS
  - SCHE-MATA
- Dynamic Sequencing Module
- Knowledge Sources

**Figure 11.1: The KNOW-PLAN Prototype System's Modules**
modeling system, are described in the chapter. The generated 3D computer model is used to provide the geometric data needed to perform the definition of the spatial relationships among various objects of the model. The geometric data is stored in the central database for further processing.

The second module in the KNOW-PLAN prototype system is the knowledge and data module. The purpose of this module is to outline the knowledge and data needed in the reasoning process for generating a construction plan. This knowledge and data are stored in the central database of the system. The central database contains information about the definition of different activities of the project, their geometric data, the definition of the different classes included in the database, their relationships, and definition of the zoning data.

The central database, during the preprocessor phase, functions as a mean of communication for passing data from the 3D computer modeling system to the dynamic sequencer. The central database provides the tools to manipulate the geometric data acquired from the 3D computer model, and the data defined in the database files that represents the available knowledge.

The dynamic sequencing module is the center piece of the KNOW-PLAN prototype system. The power of the system comes from the capability of the dynamic sequencer to provide a reasoning process to generate construction plans. The dynamic sequencer takes as input a description of the facility as defined by the acquired geometric data from a 3D computer modeling system. The dynamic sequencer progresses by using the geometric data in conjunction with the geometric knowledge and other knowledge sources to come up with a project plan. The reasoning process attempts to satisfy all the constraints defined by the different knowledge sources in the system based on the priority
of these constraints. The dynamic sequencer is basically a KGB planning system. The sequencer adopts many AI concepts in the implementation in addition to the concepts of knowledge-based technology. Among these concepts are: least-commitment principle, state-space representation, and object-oriented programming approach.

The dynamic sequencer's reasoning process is divided into two main phases. The first phase involves the reasoning process which is concerned with asserting relationships among different activities of the project. These relationships are based on the different constraints defined by the knowledge that resides in the different knowledge sources of the system. The second phase involves refining the asserted relationships to resolve conflicts, loops and implied logic to generate the optimized final project plan. This reasoning process is called the refinement reasoning process. Also, the second phase performs schedule calculations based on the finalized project plan.

The fourth module in the KNOW-PLAN prototype system is the visual simulation module. It is the module that performs the visual simulation of the construction process based on the generated sequence by the dynamic sequencer. The visual simulation module takes as input two ASCII files that are generated in the second phase of the reasoning process of the dynamic sequencer. The primary component of the visual simulation module is the postprocessor of the central database. The postprocessor is a processing environment that creates the necessary data needed to perform the visual simulation process using the WALKTHRU system.
11.2. Software and Hardware

The KNOW-PLAN prototype system has been implemented using advanced computer software and hardware technology. This section lists the different software and hardware systems that are utilized in the implementation of the KNOW-PLAN prototype system.

11.2.1. Software

Many advanced software packages and systems are utilized in the implementation of the KNOW-PLAN prototype system. The software systems range from CAD packages to Artificial Intelligence based systems. The communication between the CAD and AI based systems has been implemented using a database management system. The following is a list of the different software used in the implementation:

- **WALKTHRU**: Bechtel’s visual simulation system. It is utilized as a 3D computer modeling and 3D visual simulation system.
- **CADAM**: Computer Aided Design and Manufacturing system is used in generating a 3D computer model for the sample culvert project.
- **ART-IM**: Inference Corporation’s Artificial Intelligence tool kit. It is used in developing the knowledge-based component of the KNOW-PLAN prototype system.
- **Clipper-87**: is used in developing the central database of the KNOW-PLAN prototype system.
• 'C' language is used in developing the Path-Finder system within the WALKTHRU system.

11.2.2. Hardware

The KNOW-PLAN prototype system implementation has been conducted on several hardware platforms. The hardware platforms are running under different operating systems. The following is a list of the different operating systems and their associated hardware:

• A Silicon Graphics IRIS 4D/80GT and SUN 4/260 workstations are used as graphics workstations running under UNIX operating system. The two workstations are used mainly for 3D computer modeling and visual simulation processes. The SUN 4/260 is loaded with advanced graphics hardware and software manufactured and developed by Raster Technology.

• The CADAM system is running on an IBM 5080 mainframe that runs under VM/CMS operating system.

• The knowledge-based system and the central database are implemented on a 386 IBM/PC based system which runs under the MS-DOS operating system.

The different computer platforms communicate with each another through the campus Ethernet and through the IBM/ROLM 9571 voice/data campus telephone network.
11.3. Contributions and Benefits

Construction planning for complex projects can greatly benefit from the integration of CAD and 3D computer modeling technology with Artificial Intelligence technology to automatically generate construction plans and to simulate the construction process on a graphics display. The research provides a step forward in accomplishing such an integration to enhance the construction planning process.

The research formulates an advanced planning model which uses the geometric data of the project components as the basis for generating construction plans using a knowledge-based system. The geometric data is used in defining the spatial relationships among the different components of the project. These spatial relationships are used in the reasoning process of a dynamic sequencer to assert logical relationships among the activities of the project.

The KNOW-PLAN model utilizes 3D computer models as the source of knowledge that can be used by the dynamic sequencer to generate construction plans. This capability enables it to be a suitable planning approach for a generic project with minimal user interaction. Once a certain project is done, a similar project can be done with minimal user interaction.

The power of implementing the KNOW-PLAN model in real life applications is strengthened by the power of the technologies that are utilized by the model. The KNOW-PLAN model makes use of the capabilities of CAD and 3D computer modeling, 3D visual simulation, and Artificial Intelligence technologies.
CAD and 3D computer modeling technologies provide powerful graphics support for construction planning. This support is concerned with the pictorial representation of the project description and components. The utilization of advanced CAD and 3D computer modeling systems to perform 3D visual simulation of the construction process contributes to better planning. This is accomplished by providing tools to visually present the construction process which can lead to greater understanding and awareness of the project by management and project personnel. Also, these tools provide capabilities to check for the constructability of different planning scenarios.

Artificial Intelligence technology as represented by knowledge-based systems provides capabilities to automate the process of generating construction plans based on different constraints defined by the knowledge stored in the knowledge-base of the system. The utilization of this advanced technology provides suitable tools to generate more accurate and consistent construction plans.

In summary, the research provides an additional step in the direction of automating the process of generating construction plans. This step is based on the integration of advanced technologies to automatically generate construction plans primarily based on the geometric data of the different components that compose the project.

11.4. Recommendations and Directives

The KNOW-PLAN prototype system is a partial implementations of the KNOW-PLAN overall model. The implementation effort has been concentrated on the crucial compo-
ments of the overall model. Future extensions to the KNOW-PLAN prototype system are needed to meet the main objectives of the KNOW-PLAN model.

The major extension to the system is to capture planners' knowledge and expertise. This knowledge about construction should be added to the knowledge-base of the system to lend support to the knowledge used in the reasoning process of the dynamic sequencer to come up with a project plan. The construction knowledge needed, other than geometric knowledge, can be classified as: resource requirements knowledge, procurement knowledge, method of construction knowledge, safety consideration knowledge, etc...

This section describes future research needed to enhance the capabilities of the KNOW-PLAN prototype system to become a practical solution for the construction planning problem. The discussion is divided into two subsections. The first subsection describes research efforts that have been initiated at Virginia Tech to support the KNOW-PLAN prototype system. The second subsection describes general recommendations and directives for future extensions of the system.

11.4.1. Initiated Research Efforts

Two research efforts have been initiated at Virginia Tech to enhance the capabilities of the KNOW-PLAN prototype system. The first research effort enhances the capabilities of the dynamic sequencer of the KNOW-PLAN system. The second research provides advanced visual simulation features for construction process presentation.

The objective of the first research is to generate project plans for the linear components of high rise building projects using a knowledge-based system. The system allows the
user to perform incremental scheduling by interactively modifying the graphical output
directly from the display screen [Thabet 90].

Knowledge acquisition techniques will be utilized to capture the construction planning
and scheduling knowledge from multiple experts in the domain of high rise construction.
This elicited knowledge will be incorporated within the knowledge-based system being
implemented. This new knowledge-based system will be considered as an additional
knowledge source in the KNOW-PLAN prototype system.

The incremental scheduling capability of the model will be added to the planning and
scheduling process by allowing the user to interactively modify the graphical output di-
rectly on the screen in order to perform "what-if" scheduling scenarios. These "what-if"
scenarios will facilitate the ability to access the impact of various alternatives such as:
construction methods, availability of resources, constraints, delays, and others on the
generated schedule.

The second research effort is called VSS "Visual Schedule simulation" has been initiated
to enhance the visual simulation of the construction process [Skolnick, Morad, and
Beliveau 90]. The VSS system reads project schedule data. The user is allowed to select
the type of schedule to be simulated (actual versus planned), the schedule period, and
the early or late schedule simulation.
11.4.2. Future Extensions

Several issues became obvious during the development of the KNOW-PLAN prototype system. These issues were enumerated as undeveloped within the text. A synopsis of this future work is presented below.

- Acquiring more knowledge about the construction process. A significant effort will be required to collect knowledge about resources, productivity, safety, procurement, method of construction, weather-constraints,... etc. The acquired knowledge will be used in the reasoning process to optimize the generated construction plan. Upon collecting this knowledge, the overall KNOW-PLAN system would become more of an expert system.

- Defining the class definition and relations for different types of projects based on a knowledge acquisition process. The acquired knowledge can be stored in generic files to be used with different types of projects.

- Designing the system to have a hierarchical levels of plan abstraction. This means to have different levels of project plan in which an activity in a higher level will have a subnetwork of many activities in a lower level.

- Mapping an object in the 3D computer model of the designed facility to more than one activity in the project plan. At the same time allowing an activity to be associated with more than one object. This will lead to many-to-many type of relation between activities and 3D objects.

- Breaking down the object’s volume boundary to different volumes that approximate the shape of the object instead of defining it as a single volume. This will lead to more accurate spatial relationship definitions.
among different objects of the project. The same thing can be applied to the definition of the zones' boundaries.

- Building the Optimizers to optimize the data included in the database files of the central database. The optimizers are knowledge-based systems that can eliminate data redundancy and resolve conflicts between the different instances of each entity type or relationships, if such conflicts exist.

- Adopting a coding system to name the objects during the design phase. The coding system can be the source of asserting the classes to which each object is related. The assertion process can be based on a reasoning process to be performed by the dynamic sequencer. Accomplishing this task will minimize the user input to the system.

- Automate the assertion of the type of connection between the different objects of the project. This can be accomplished by embedding an expert system within the WALKTHRU system. The expert system will analyze the simulation of moving the object in different directions and concluding the type of the connection that exists between the object and other objects around it.

- Many processes performed by the dynamic sequencing process can be performed using an object-oriented approach. Accordingly, the object-oriented programming approach will play an important role in future extensions of the system.

- Building the central database and the knowledge-based components of the KNOW-PLAN system to be executed from a main 'C' program which acts as an integrated environment.

- Providing graphical presentations for the dynamic sequencing process. These graphical presentation can present the current priority cycle, the
status of the opened and closed lists, and the members of the current final project plan during the reasoning process.

- Implementing the interactive schedule modification stage of the KNOW-PLAN model introduced in Chapter 5.
- Implementing the conventional reporting stage of the KNOW-PLAN model by linking the system to a CPM processor. This will provide the capabilities of presenting the project plan in the form of bar-charts, time-scaled networks, and logic networks.

11.5. Conclusion

KNOW-PLAN is envisaged as an advanced planning and scheduling system that integrates CAD technology with AI technology and overcomes some of the limitations that exist in current planning techniques. The system generates construction plans using a KGB system. The generated sequence is based in part on the geometric data that defines the spatial relationships among various components of the designed facility which is acquired from the 3D computer model of the facility. The generated sequence also depends on the different knowledge rules defined in the knowledge-base. The planning process is enhanced by providing visual tools to simulate the construction process graphically on the workstation.

Construction planning for complex projects can greatly benefit from the integration of CAD systems and knowledge-based systems to generate construction plans and to visually simulate the construction process. More accurate construction schedules will be
generated by using the stored and structured construction knowledge. Greater understanding and awareness of the project by management can be achieved by visually simulating the construction process. During actual construction, the simulation process will help the site personnel to visualize how and what should be constructed. Visual simulation of the construction process will be a valuable tool to support and defend construction claims in the future.

The KNOW-PLAN model provides a significant step in developing more effective planning and scheduling techniques. The model integrates the capabilities of two powerful technologies, Artificial Intelligence and Computer-Aided Design. This integration is used to solve the construction planning problem. The research provides an advanced approach to solve the construction planning problem.
Bibliography


Inference Corp., (1989), "ART-IM in the DOS Environment.", Los Angeles, CA.


Kartam, N., and Levitt, R., (1989), "Planning and Execution in Uncertain Domains.", 6th Conference in Computing in Civil Engineering, September, Atlanta, GA.


Stefik, M., (1981), "Planning with Constraints.", Artificial Intelligence, 16(2), pp 111-140.


Tate, A., (1975), "Interacting Goals and Their Use.", Proc. of International, Joint Conference on Artificial Intelligence, Tbilisi, U.S.S.R.

Tate, A., (1977), "Generating Project Networks.", Proc. of the 5th IJCAI, pp 888-893.

Thabet, W., (1990), "A knowledge-based Planning System.", Draft copy of a proposal submitted to the faculty of Virginia Tech in partial fulfilment of the requirements of the Ph.D. degree, Blacksburg.


Wong, E., and Fu, K., "A Hierarchical Orthogonal Space Approach to Collision-Free Path Planning.", School of Electrical Engineering, Purdue University.

Appendix A. The Planning Problem

Appendix A1: Means-Ends Technique: Sample Solution

The solution for the jigsaw puzzle problem using the Means-Ends technique based on the linear scheduling scenario is as follow:

No = Not installed
Yes = installed

Initial State

PC#(1) = NO
PC#(2) = NO
PC#(3) = NO
PC#(4) = NO
PC#(5) = NO
PC#(6) = NO
PC#(7) = NO
PC#(8) = NO
PC#(9) = NO

Goal State

PC#(1) = YES
PC#(2) = YES
PC#(3) = YES
PC#(4) = YES
PC#(5) = YES
PC#(6) = YES
PC#(7) = YES
PC#(8) = YES
PC#(9) = YES

Actions (Operators)

Action
Install PC#(i)

Preconditions
PC#(i) = No and All rules satisfied

Constructive effect
PC#(i) = Yes

Destructive effect
PC#(i) = NO
Rules (to define interlocking constraints)

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Conditions</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC#(2) = Yes or PC#(4) = Yes</td>
<td>Install PC#(1)</td>
</tr>
<tr>
<td>2</td>
<td>PC#(1) = Yes or PC#(3) = Yes or PC#(5) = Yes</td>
<td>Install PC#(2)</td>
</tr>
<tr>
<td>3</td>
<td>PC#(2) = Yes or PC#(6) = Yes</td>
<td>Install PC#(3)</td>
</tr>
<tr>
<td>4</td>
<td>PC#(1) = Yes or PC#(5) = Yes or PC#(7) = Yes</td>
<td>Install PC#(4)</td>
</tr>
<tr>
<td>5</td>
<td>PC#(2) = Yes or PC#(4) = Yes or PC#(6) = Yes or PC#(8) = Yes</td>
<td>Install PC#(5)</td>
</tr>
<tr>
<td>6</td>
<td>PC#(3) = Yes or PC#(5) = Yes or PC#(9) = Yes</td>
<td>Install PC#(6)</td>
</tr>
<tr>
<td>7</td>
<td>PC#(4) = Yes or PC#(8) = Yes</td>
<td>Install PC#(7)</td>
</tr>
<tr>
<td>8</td>
<td>PC#(5) = Yes or PC#(7) = Yes or PC#(9) = Yes</td>
<td>Install PC#(8)</td>
</tr>
<tr>
<td>9</td>
<td>PC#(6) = Yes or PC#(8) = Yes</td>
<td>Install PC#(9)</td>
</tr>
</tbody>
</table>

Note: The order by which the rules are listed affects the sequence of the solution. Additional rules can be included to satisfy specific objectives. Also, the order of the actions affects the way the system solves the problem.

In the linear scheduling scenario, it is assumed that the element to be installed does not need to have a physical relationship with the last installed element. However, the system can be modified to count for continuous path of installation.

Example

Assume the order of the actions is as follow:

- #1 install PC#(9)
- #2 install PC#(5)
#3 install PC#(6)
#4 install PC#(2)
#5 install PC#(4)
#6 install PC#(1)
#7 install PC#(7)
#8 install PC#(8)
#9 install PC#(3)

Starting with PC#(3) as the first element to be installed, the following is a description for the problem solving process:

Step#1: Fact PC#(3) = Yes

Step#2: Action #1 cannot be considered because it doesn’t satisfy the conditions of rule #2 and rule #6
Action #2 cannot be considered
Action #3 can be considered because it satisfy the conditions of rule # 6
   => PC#(6) = Yes

Step#3: Action #1 can be considered (rule #9) => PC#(9) = Yes

Step#4: Action #2 can be considered (rule #5) => PC#(5) = Yes

Step#5: Action #3 can be considered
   .
   .
   .

The final solution (sequence) will be:

Install PC#(3)
Install PC#(6)
Install PC#(9)
Install PC#(5)
install PC#(2)
Install PC#(4)
Install PC#(1)
Install PC#(7)
Install PC#(8)
Appendix A2: Production Systems: Sample Solution

The following includes the source code listing of a production system developed to generate a linear schedule for the installation sequence of the (3 X 3) jigsaw puzzle. The system has been developed using VP-Expert. VP-Expert is an expert system tool which utilizes a Backward Chaining (goal-directed) control mechanism for navigating the knowledge-base which contains mainly procedural knowledge (rules).

Source Code Listing

RUNTIME;

ACTIONS

PC#1 = NOT_INSTALLED
PC#2 = NOT_INSTALLED
PC#3 = NOT_INSTALLED
PC#4 = NOT_INSTALLED
PC#5 = NOT_INSTALLED
PC#6 = NOT_INSTALLED
PC#7 = NOT_INSTALLED
PC#8 = NOT_INSTALLED
PC#9 = NOT_INSTALLED
FIND R
DISPLAY "SCHEDULING ........ IN PROGRESS"
DISPLAY "STEP ........1 AND 2"
FIND GOAL
RESET N
DISPLAY "STEP ........3"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP ........4"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP ........5"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP ........6"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP ........7"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP ........8"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "STEP .......9"
FIND GOAL
RESET GOAL
RESET N
DISPLAY "THE DESIRED FINAL GOAL IS ACHIEVED";

RULE 1
IF      PC#2 = INSTALLED
   OR    PC#4 = INSTALLED
   AND   PC#1 = NOT_INSTALLED
THEN    PC#1 = INSTALLED
        N  = (N + 1)
        DISPLAY "INSTALL PIECE NO. 1";

RULE 2
IF      PC#1 = INSTALLED
   OR    PC#3 = INSTALLED
   OR    PC#5 = INSTALLED
   AND   PC#2 = NOT_INSTALLED
THEN    PC#2 = INSTALLED
        N  = (N + 1)
        DISPLAY "INSTALL PIECE NO. 2";

RULE 3
IF      PC#2 = INSTALLED
   OR    PC#6 = INSTALLED
   AND   PC#3 = NOT_INSTALLED
THEN    PC#3 = INSTALLED
        N  = (N + 1)
        DISPLAY "INSTALL PIECE NO. 3";

RULE 4
IF      PC#1 = INSTALLED
   OR    PC#5 = INSTALLED
   OR    PC#8 = INSTALLED
   AND   PC#4 = NOT_INSTALLED
THEN    PC#4 = INSTALLED
        N  = (N + 1)
        DISPLAY "INSTALL PIECE NO. 4";

RULE 5
IF      PC#2 = INSTALLED
   OR    PC#4 = INSTALLED
   OR    PC#6 = INSTALLED
   OR    PC#8 = INSTALLED
   AND   PC#5 = NOT_INSTALLED
THEN    PC#5 = INSTALLED
        N  = (N + 1)
        DISPLAY "INSTALL PIECE NO. 5";
RULE 6
IF PC#3 = INSTALLED
OR PC#5 = INSTALLED
OR PC#9 = INSTALLED
AND PC#6 = NOT_INSTALLED
THEN PC#6 = INSTALLED
N = (N + 1)
DISPLAY "INSTALL PIECE NO. 6";

RULE 7
IF PC#4 = INSTALLED
OR PC#8 = INSTALLED
AND PC#7 = NOT_INSTALLED
THEN PC#7 = INSTALLED
N = (N + 1)
DISPLAY "INSTALL PIECE NO. 7";

RULE 8
IF PC#5 = INSTALLED
OR PC#7 = INSTALLED
OR PC#9 = INSTALLED
AND PC#8 = NOT_INSTALLED
THEN PC#8 = INSTALLED
N = (N + 1)
DISPLAY "INSTALL PIECE NO. 8";

RULE 9
IF PC#6 = NOT_INSTALLED
OR PC#8 = INSTALLED
AND PC#9 = NOT_INSTALLED
THEN PC#9 = INSTALLED
N = (N + 1)
DISPLAY "INSTALL PIECE NO. 9";

RULE 10
IF FIRST <> 0
AND N = 9
THEN GOAL = ACHIEVED
DISPLAY "The goal is achieved";

RULE A1
IF FIRST = PC#1
THEN PC#1 = INSTALLED
R = 1;

RULE A2
IF FIRST = PC#2
THEN PC#2 = INSTALLED
R = 1;

RULE A3
IF FIRST = PC#3
THEN PC#3 = INSTALLED
   R  = 1;

RULE A4
IF FIRST = PC#4
THEN PC#4 = INSTALLED
   R  = 1;

RULE A5
IF FIRST = PC#5
THEN PC#5 = INSTALLED
   R  = 1;

RULE A6
IF FIRST = PC#6
THEN PC#6 = INSTALLED
   R  = 1;

RULE A7
IF FIRST = PC#7
THEN PC#7 = INSTALLED
   R  = 1;

RULE A8
IF FIRST = PC#8
THEN PC#8 = !INSTALLED
   R  = 1;

RULE A9
IF FIRST = PC#9
THEN PC#9 = INSTALLED
   R  = 1;

ASK FIRST: "WHAT IS THE FIRST ELEMENT TO BE INSTALLED?";

CHOICE FIRST: PC#1, PC#2, PC#3, PC#4, PC#5, PC#6, PC#7, PC#8, PC#9
Appendix B. The Path-Finder System

The Path-Finder system finds the optimum path of an object from one position and orientation in a 3D computer model to another position and orientation. It optimizes the path subject to motion constraints on mechanisms which manipulate the object as well as ensuring that there are no interferences along the path between the moving object and manipulation mechanisms and other objects in the model.

The program operates from the WALKTHRU system. It utilizes AI concepts to find the optimum path for moving an object from an initial location to a desired location in the 3-D computer model. The following subsections provide a brief description of the problem representation and the search control strategy used by the system to find a solution for the path problem.

1. Problem Environment Representation

The problem environment is formally represented by using the State-Space Representation. The process of developing the state-space representation in the Path-Finder is described as follows:

**Definition of the Start State (Start Node):** The attributes of the start state are defined by the initial conditions of the problem. The initial position and orientation of the object to be moved and its manipulation mechanisms define the attributes of the start state. The initial position of the object is the position in the 3D computer model from which the motion process is initiated as defined by the user. The attributes which define the start state are:

- Position \((x, y, z)\) of the moving object.
- Orientation \((\theta_x, \theta_y, \theta_z)\) of the moving object.
- Position(s) \((x, y, z)\) of the manipulation mechanism(s).
- Orientation(s) \((\theta_x, \theta_y, \theta_z)\) of the manipulation mechanism(s).
- Total cost associated with the start state.

**Definition of the Goal State (End Node):** The attributes of the goal state are defined by the conditions of the desired solution. The final position and orientation of the moving object define the attributes of the goal state. The final position of the object is as defined in the static (original) configuration of the 3D computer model. The attributes which define the goal state are:

- Position \((x, y, z)\) of the moving object.
• Orientation ($\theta_1$, $\theta_2$, $\theta_3$) of the moving object at the final position.

Definition of the Operators (Actions): The operators are the actions which transfer the position and orientation of the moving object and its manipulation mechanisms from one state (parent node) to another state (new node). Each operator defines a step change in the direction of one degree of freedom (transitional or rotational). The step size is variable, and is a function of the relative distance between the current position of the moving object and the final position.

Accordingly, each state in the state-space represents a discrete position and orientation of the object in space. Successive states represent the position of the object after many step changes in the direction of different degrees of freedom. The degrees of freedom are defined by the user from the WALKTHRU object motion menus. The user can define a unit cost for moving along each degree of freedom. The number of degrees of freedom depends on the configuration of the manipulation mechanisms. Also, the user can define a different penalties (cost) for changing the direction of the degree of freedom between two successive steps.

2. The Objective Function

The objective of the search process is to find an optimum collision free path for moving the object from its initial position to the desired position. The objective function of the optimization is minimum total cost. The total cost for a specific state (node) in the search space is a function of two cost components: actual cost and estimated cost. Figure B.1 depicts the cost equations for different states.

- Total Cost = Actual Cost + Estimated Cost
- The actual cost is the accumulated cost due to applying the operators from the start node.
- Actual cost = $\sum$ (applied operators cost) + $\sum$ (change in direction cost)
- Change in direction cost is added when any two successive operators are not for the same degree of freedom.
- Estimated cost is a function of the distance between the current position of the current state and the desired position of the moving object as defined by the goal state.

3. The Search Strategy

The program uses the Generate and Test approach as the problem solving strategy. The search process starts from the start state (start node). The generator generates all possible successor states from the start state using the applicable operators (which represents the different degrees of freedom allowed by the manipulation mechanisms). The total cost of each generated node is calculated based on the cost of the applied operators, the direction of motion, and the relative distance to the final position. All generated nodes are posted to a list called the open-list. For each generated node the following conditions are checked:

- If the node already exists in the open or close lists.
- If the position of the manipulation mechanisms are within the allowed motion limits.
Cost Calculation

Actual Cost(I) = Actual Cost(O) \times \text{Cost OP(X)}
Estimated Cost(I) = \text{Cost of L(I)}

Actual Cost(P) = Actual Cost(I) \times \text{Cost OP(Y)} + \text{Change Dir Cost}
Estimated Cost(P) = \text{Cost of L(P)}

Actual Cost(N) = Actual Cost(P) \times \text{Cost OP(X)} + \text{Change Dir Cost}
Estimated Cost(N) = \text{Cost of L(N)}

Figure B.1: PATH-FINDER: Cost Calculation
The next step is to select a node from the open-list which has the minimum total cost. Then the evaluator will check the following conditions:

- If the goal state is reached (by comparing the new position and orientation with the final position and orientation). The goal is achieved if the differences are within the accuracy levels defined by the user.
- If there is an Interference between the moving object and other objects in the 3D computer model as a result of moving the object by the manipulation mechanism to the position defined by the selected state.
- If there is an Interference between the manipulation mechanism of the moving object and other objects in the 3D computer model as a result of moving the object to the position defined by the selected state.

Future extensions of the system will incorporate several constraints which will be imposed on the search process. These extensions will include: equipment integrity (tipping, etc.), control of object velocity, and safety of the manipulation process.

The search will stop if the final position is reached. If the final position has not been reached, the current node will be posted to the close-list and a new set of nodes, which represent new states, will be generated from the current node. The new set of nodes will be posted to the open-list. In case of interference, the path will be terminated with no new nodes to be generated from the current node. The search process will backtrack one level. Then, the generator will select a node from the open-list to continue the search process.

The described search strategy can be classified as a depth-first control search strategy with a data-driven reasoning direction. However, it is modified to consider the total cost as the criteria for selecting the next node in the search space from the open-list and not the level at which the node is created within the nodes hierarchy.

The output of the search process is a list of the nodes along the selected path. The position and orientation of the 3D computer model objects at each node are saved in a Record file. The record file contains frames of the objects' position and orientation at each node. This record file will be executed by the Play Back routine of WALKTHRU to visually simulate the generated path using the 3D computer model of the facility.

Figure B.2 depicts a search tree representing the status of the search space at an intermediate stage of the search process. In the Figure, node A is the start node and node Z is the finish node. These two nodes represent the initial and goal states, respectively. The nodes: B, F, H, J, and I are already searched nodes. They represent the close-list. The nodes: C, D, E, G, K, L, and M are not searched nodes. They represent the open-list. Node L is the current node at this stage of the search process. The path for the current partial solution is all the ancestors of the node L. The ancestor ordered list is: A, B, F, H, and I.

Figure B.3 depicts the graph after finding a solution. The open list consists of the following nodes: C, D, E, G, K, M, N, and Q. The close-list consists of the following nodes: A, B, F, H, J, I, L, O, and P. The node P represents the final position of the object which satisfy the conditions at the goal state. Accordingly, the solution of the problem is the found path from node A to node P. The ordered sequence of the solution path is: A, B, F, H, I, L, O, and P.
Open List: \((C, D, E, G, K, M, L)\)

Closed List: \((A, B, F, H, I, I)\)

Ancestors of Node "L": \((A \rightarrow B \rightarrow F \rightarrow H \rightarrow I)\)

*Figure B.2: The Search Space at an Intermediate State*
Open List: (C, D, E, G, K, M, N, Q)

Closed List: (A, B, F, H, J, I, L, O, P)

Ancestors of Node "P": (A → B → F → H → I → L → O)

"An Acceptable Solution"

Figure B.3: The Search Space at the Goal State
4. Applications for the Path-Finder

The primary use of the Path-Finder is to determine during design and pre-construction if objects can be maneuvered into their final destinations. This testing will be done using the WALKTHRU system. As complex objects are designed or planned, the method of placement can be modeled from the laydown yard to final location. The system would model the equipment required to move the object as well as the normal equipment maneuvering capabilities. The equipment maneuvering capabilities would be via a set of degrees of freedom for each rotation and lifting potential. The Path-Finder program would identify whether the object and the equipment required could actually install the object given the existing objects within the model. The visual output would be an animation of the equipment and object as they travel through the 3D computer model from original laydown to final positioning.

The system has great potential to enhance and enrich the construction planning process by improving the constructability of construction schedules. This improvement is accomplished by visually simulating the construction process during the design phase to avoid future constructability problems. By using 3D computer model of the designed facility, it will be possible to check for the constructability of different sequencing scenarios of various components prior to construction start.

The constructability checking of different sequencing scenarios can be achieved by simulating the movement of critical objects from their initial positions to their final positions in the 3D computer model. This will check if it is possible to install these objects without interfering or hitting other objects in the work space. The outcomes of the constructability checking process can be used to enhance and enrich the automated process of generating construction schedules by knowledge-based systems.
Appendix C. The Dynamic Sequencer Rules

Appendix C1: External Data Interface Functions and Rules

Activity External Data Interface:
(def-external-data act-string
  "Activity record is read as STRING" 60
  (all 0 60 :char))

(def-external-data act-data
  "Activity record is read as FIELDS" 60
  (act-name 0 8 :char :symbol)
  (act-dur 8 4 :char :integer)
  (min-X 12 8 :char :float)
  (max-X 20 8 :char :float)
  (min-Y 28 8 :char :float)
  (max-Y 36 8 :char :float)
  (min-Z 44 8 :char :float)
  (max-Z 52 8 :char :float))

(defglobal ?*act-buf* = (alloc-buffer act-string))

defrule read-act-rec
  "Read the activity geometric data from ACTIVITY.DAT file"
  (declare (salience (- ?*firel* 100)))
  = >
  (bind ?stream (open "\art\phd\data\activity.dat" "r"))
  (while (not (feof ?stream))
    DO
    (bind ?line-buf (read-line ?stream))
    (IF (not (equal ?line-buf *eof*))
      THEN
      (poke ?*act-buf* act-string all ?line-buf)
      (bind ?sname (peek ?*act-buf* act-data act-name))
      (IF (not (schematic ?sname))
        THEN
        (schematic ?sname t)
        (slotc ?sname act-dur t)
        (slotc ?sname max-X t)
        (slotc ?sname min-X t)
(slotc ?sname max-Y t)
(slotc ?sname min-Y t)
(slotc ?sname max-Z t)
(slotc ?sname min-Z t)
(slotc ?sname act-zone t)
(slotc ?sname act-zon-bor t)
(slotc ?sname early-start t)
(slotc ?sname late-start t)
(slotc ?sname early-finish t)
(slotc ?sname late-finish t)
(slotc ?sname act-name)
(map-schema ?*act-buf* act-data ?sname))
(retract (schema ?sname (act-name)))
(assert (schema ?sname (is-a activity)))
(assert (schema ?sname (scheduled-early no)))
(assert (schema ?sname (scheduled-late no)))
(assert (schema ?sname (calc-early to-schedule-early)))
(assert (schema ?sname (calc-late to-schedule-late))))
(close ?stream))

Class External Data Interface:
(def-external-data class-string
  "Class record is read as STRING" 44
  (all 0 44 :char))

(def-external-data class-data
  "Class record is read as FIELDS" 44
  (cls-name 0 4 :char :symbol)
  (cls-desc 4 25 :char :string)
  (X-dir 29 3 :char :symbol)
  (Y-dir 32 3 :char :symbol)
  (Z-dir 35 3 :char :symbol)
  (hi-1 38 3 :char :symbol)
  (hi-2 41 3 :char :symbol))

(defglobal ?*cls-buf* = ( alloc-buffer class-string))

(defrule read-class-rec
  "Read the class data from class.dat file"
  (declare (salience (- ?*file* 300)))
  = >
  (bind ?stream (open "\art\phd\data\class.dat" "r"))
  (while (not (feof ?stream)))
    do
      (bind ?line-buf (read-line ?stream))
      (IF (not (equal ?line-buf *eof*))
      THEN
        (poke ?*cls-buf* class-string all ?line-buf)
        (bind ?sname (peek ?*cls-buf* class-data cls-name))
        (IF (not (schemap ?sname))
        THEN

Appendix C. The Dynamic Sequencer Rules
Zoning External Data Interface:
(def-external-data zone-string
 "Zone record is read as STRING" 52
 (all 0 52 :char))

(def-external-data zone-data
 "Zone record is read as FIELDS" 52
 (zon-name 0 4 :char :symbol)
 (min-X 4 8 :char :float)
 (max-X 12 8 :char :float)
 (min-Y 20 8 :char :float)
 (max-Y 28 8 :char :float)
 (min-Z 36 8 :char :float)
 (max-Z 44 8 :char :float))

(defglobal ?*zon-buf* = (alloc-buffer zone-string))

(defrule read-zone-rec
 "Read the zone geometric data from ZONE.DAT file"
 (declare (salience (- ?*fire1* 200)))
 (= >
 (bind ?stream (open "\|\art\|\phd\|\data\|\zone.dat" "r"))
 (while (not (feof ?stream)))
 DO
 (bind ?line-buf (read-line ?stream))
 (IF (not (equal ?line-buf *eof*))
 THEN
 (poke ?*zon-buf* zone-string all ?line-buf)
 (bind ?sname (peek ?*zon-buf* zone-data zon-name))
 (IF (not (schemap ?sname))
 THEN
 (schemac ?sname t)
 (slotc ?sname min-X t)
 (slotc ?sname max-X t)
 (slotc ?sname min-Y t)
 (slotc ?sname max-Y t)
 (slotc ?sname min-Z t)
 (slotc ?sname max-Z t)
)
(slotc ?sname zon-name)
  (map-schema ?*zon-buf* zone-data ?sname))
(retract (schema ?sname (zon-name)))
(assert (schema ?sname (is-a ZONE))))
(close ?stream))

Activity/Class External Data Interface:
(def-external-data act-cls-string
  "Activity class record is read as STRING" 12
  (all 0 12 :char))

(def-external-data act-cls-data
  "Activity class record is read as FIELDS" 12
  (act-name 0 8 :char :symbol)
  (act-class 8 4 :char :symbol))

(defglobal ?*act-cls-buf* = (alloc-buffer act-cls-string))

(defrule read-act-cls-rec
  "Read the activity/class data from ACTCLASS.DAT file"
  (declare (salience (- ?*firel* 400))))
= >
(bind ?stream (open "\|art\|phd\|data\|actclass.dat" "r"))
(while (not (feof ?stream))
  do
    (bind ?line-buf (read-line ?stream))
    (IF (not (equal ?line-buf *eof*))
      THEN
        (poke ?*act-cls-buf* act-cls-string all ?line-buf)
        (bind ?sname (peek ?*act-cls-buf* act-cls-data act-name))
        (bind ?class (peek ?*act-cls-buf* act-cls-data act-class))
        (IF (not (slotp ?sname act-class))
          THEN
            (slotc ?sname act-class t)
            (assert (schema ?sname (act-class ?class)))))
  (close ?stream))

Connection External Data Interface:
(def-external-data con-string
  "Activity connection record is read as STRING" 28
  (all 0 28 :char))

(def-external-data con-data
  "Activity connection record is read as FIELDS" 28
  (con-name 0 8 :char :symbol)
  (act-from 8 8 :char :symbol)
  (act-to 16 8 :char :symbol)
  (con-type 24 4 :char :symbol))

(defglobal ?*con-buf* = (alloc-buffer con-string))
(defrule read-con-rec
  "Read the connection data from CONTYPE.DAT file"
  (declare (salience (~ *fire1* 500)))
  = >
  (bind ?stream (open "\\art\\phdl\data\contype.dat" "r"))
  (while (not (feof ?stream)))
  DO
  (bind ?line-buf (read-line ?stream))
  (IF (not (equal ?line-buf *eof*))
    THEN
    ( poke ?*con-buf* con-string all ?line-buf)
    ( bind ?sname (peek ?*con-buf* con-data con-name))
    ( IF (not (schema ?sname))
      THEN
      ( schemac ?sname t)
      ( slotec ?sname act-from t)
      ( slotec ?sname act-to t)
      ( slotc ?sname con-type t)
      ( slote ?sname con-name)
      ( map-schema ?*con-buf* con-data ?sname))
    ( retract (schema ?sname (con-name)))
    ( assert (schema ?sname (is-a connect)))))
  (close ?stream))

Dominating Class External Data Interface:
(def-external-data cls-domin-string
  "Dominating class record is read as STRING" 21
  (all 0 21 :char))

(def-external-data cls-domin-data
  "Dominating class record is read as FIELDS" 21
  (cls-dom-name 0 5 :char :symbol)
  (cls-1 5 4 :char :symbol)
  (cls-2 9 4 :char :symbol)
  (con-type 13 4 :char :symbol)
  (dominating 17 4 :char :symbol))

(defglobal ?*cls-dom-buf* = (alloc-buffer cls-domin-string))

(defrule read-cls-dominating-rec
  "Read the dominating class relations DOMCLASS.DAT file"
  (declare (salience (~ *fire1* 600)))
  = >
  (bind ?stream (open "\\art\\phdl\data\domclass.dat" "r"))
  (while (not (feof ?stream)))
  DO
  (bind ?line-buf (read-line ?stream))
  (IF (not (equal ?line-buf *eof*))
    THEN
    ( poke ?*cls-dom-buf* cls-domin-string all ?line-buf)
    ( bind ?sname (peek ?*cls-dom-buf* cls-domin-data

Appendix C. The Dynamic Sequencer Rules 293
(IF (not (schemap ?sname))
  THEN
  (schemac ?sname t)
  (slotc ?sname cls-1 t)
  (slotc ?sname cls-2 t)
  (slotc ?sname con-type t)
  (slotc ?sname dominating t)
  (slotc ?sname cls-dom-name)
  (map-schema ?*cls-dom-buf* cls-dom-data ?sname))
  (retract (schema ?sname (cls-dom-name)))
  (assert (schema ?sname (is-a class dominator))))
(close ?stream))

Special Class Relation External Data Interface:
(def-external-data cls-rel-string
  "Class relation record is read as STRING" 33
  (all 0 33 :char))

(def-external-data cls-rel-data
  "Class relation record is read as FIELDS" 33
  (cls-rel-name 0 5 :char :symbol)
  (cls-l 5 4 :char :symbol)
  (cls-2 9 4 :char :symbol)
  (con-type 13 4 :char :symbol)
  (rel-X 17 4 :char :symbol)
  (rel-Y 21 4 :char :symbol)
  (rel-Z 25 4 :char :symbol)
  (cls-install 29 4 :char :symbol))

(defglobal ?*cls-rel-buf* = (alloc-buffer cls-rel-string))

(defrule read-cls-relation-rec
  "Read the class relations from RELCLASS.DAT file"
  (declare (salience (- ?*fire1* 700)))
  = >
  (bind ?stream (open "\|art\\phd\\data\\relclass.dat" "r"))
  (while (not (feof ?stream)))
  DO
  (bind ?line-buf (read-line ?stream))
  (IF (not (equal ?line-buf *eof*)))
    THEN
      (poke ?*cls-rel-buf* cls-rel-string all ?line-buf)
      (bind ?sname (peek ?*cls-rel-buf* cls-rel-data cls-rel-name))
  (IF (not (schemap ?sname))
    THEN
      (schemac ?sname t)
      (slotc ?sname cls-1 t)
      (slotc ?sname cls-2 t)
      (slotc ?sname con-type t)
(slotc ?sname rel-X t)
(slotc ?sname rel-Y t)
(slotc ?sname rel-Z t)
(slotc ?sname cls-install t)
(slotc ?sname cls-rel-name)
(map-schema ?*cls-rel-buf* cls-rel-data ?sname)
(retract (schema ?sname (cls-rel-name)))
(assert (schema ?sname (is-a CLASS-RELATION))))
(close ?stream))
Appendix C2: Zone Allocation Rule

;* Defining the rule to find the zones in which an activity is located.
;* Also, find the zones with which an activity shares a common surface.
;* The zoning parameter plays an important role in the level of details
;* of the sequence to be generated.

(deffun allocate-act-zone-and-borders
  "Allocate the zones in which the activity or object is located"
  (declare (salience (+ ?*fire! 1 1000)))

  (schema ?current-zone ;; Attach the name of a zone and its slots
    (is-a zone); ;* to variables to be used in the geometric
    (min-X ?ZminX) ;; data comparison with activities' data
    (max-X ?ZmaxX)
    (min-Y ?ZminY)
    (max-Y ?ZmaxY)
    (min-Z ?ZminZ)
    (max-Z ?ZmaxZ))

  (schema ?current-act; ;; Attach the name of an activity and its
    (is-a activity) ;; slots to variables to be used in the
    (min-X ?AminX) ;; geometric data comparison with zones' data
    (max-X ?AmaxX)
    (min-Y ?AminY)
    (max-Y ?AmaxY)
    (min-Z ?AminZ)
    (max-Z ?AmaxZ))

  =>

  (IF (not (or (> = ?ZmaxX ?AminX))) ;; To check if the activity is
       (> = ?ZmaxY ?AminY)); ;* located in the current zone
       (> = ?ZmaxZ ?AminZ)
     (IF (not (or (> < ?ZmaxX ?AminX))) ;; To check if the activity shares
       (< = ?ZmaxY ?AminY) ;; a common surface with the zone
       (< = ?ZmaxZ ?AminZ); ;* and the activity and the zone
       (> = ?ZminX ?AmaxX) ;; are not overlapping
       (> = ?ZminY ?AmaxY)
       (> = ?ZminZ ?AmaxZ)) THEN

     (IF (or (< = ?ZmaxX ?AminX)
       (< = ?ZmaxY ?AminY)
       (< = ?ZmaxZ ?AminZ)
       (> = ?ZminX ?AmaxX)
       (> = ?ZminY ?AmaxY)
       (> = ?ZminZ ?AmaxZ)) THEN

     (assert (schema ?current-act (act-zone ?current-zone))))))
Appendix C3: Logic Definition Rules (Geometric Reasoning)

Logic Between Activities of the Same Class and Same Zone

(defrule logic-same-zone-class
  "Defining logic between activities of the same class in the same zone"
  (declare (salience (- ?*fire1* 2000)))
  (schema ?act-A ;* Select the first activity located
    (is-a activity) ;* in the zone ?Z
    (act-zone ?Z)
    (act-class ?C)
    (min-X ?minXA)
    (max-X ?maxXA)
    (min-Y ?minYA)
    (max-Y ?maxYA)
    (min-Z ?minZA)
    (max-Z ?maxZA))

  (is-a activity) ;* Select the first activity located
  (act-zone ?Z) ;* in the zone ?Z
  (act-class ?C)
  (min-X ?minXB)
  (max-X ?maxXB)
  (min-Y ?minYB)
  (max-Y ?maxYB)
  (min-Z ?minZB)
  (max-Z ?maxZB))

(test (not (or (< ?maxXA ?minXB)) ;* Test: if the two activities
  (< ?maxYA ?minYB)); ;* NOT (NOT Overlapping) which
  (< ?maxZA ?minZB)); ;* means they are overlapping
  (> ?minXA ?maxXB)); ;* or sharing a common surface
  (> ?minYA ?maxYB)
  (> ?minZA ?maxZB))))

(schema ?C
  (is-a class) ;* binding the slot values of the class
  (X-dir ?X-dir) ;* schema to variables.
  (Y-dir ?Y-dir)
  (Z-dir ?Z-dir)
  (hi-1 ?hi-1)
  (hi-2 ?hi-2))

= >

;* Check which activity is lower and which is
;* higher in the three X, Y, Z coordinates
(IF (or
    (> = ?minXA ?maxXB)
  (and (> ?minXA ?minXB)
    (> = ?maxXA ?maxXB)))

Appendix C. The Dynamic Sequerencer Rules

297
(and (= ?minXA ?minXB) ;* Checking in the X direction
  (> ?maxXA ?maxXB))
(and (> ?minXA ?minXB)
  (<= ?maxXA ?maxXB)))
THEN
(bond ?low-in-X ?act-B)
(bond ?hi-in-X ?act-A)
else
(IF (and (= ?minXA ?minXB)
          (= ?maxXA ?maxXB))
    THEN
    (bind ?low-in-X BOTH)
    else
    (bind ?low-in-X ?act-A)
    (bind ?hi-in-X ?act-B))

(IF (or
     (> = ?minYA ?maxYB) ;* Checking in the Y direction
     (and (> ?minYA ?minYB)
          (> = ?maxYA ?maxYB))
     (and (= ?minYA ?minYB)
          (> ?maxYA ?maxYB))
     (and (> ?minYA ?minYB)
          (<= ?maxYA ?maxYB)))
    THEN
    (bind ?low-in-Y ?act-B)
    (bind ?hi-in-Y ?act-A)
    else
    (IF (and (= ?minYA ?minYB)
             (= ?maxYA ?maxYB))
        THEN
        (bind ?low-in-Y BOTH)
        else
        (bind ?low-in-Y ?act-A)
        (bind ?hi-in-Y ?act-B))

(IF (or
     (> = ?minZA ?maxZB) ;* Checking in the Z direction
     (and (> ?minZA ?minZB)
          (> ?maxZA ?maxZB))
     (and (= ?minZA ?minZB)
          (> ?maxZA ?maxZB))
     (and (> ?minZA ?minZB)
          (<= ?maxZA ?maxZB)))
    THEN
    (bind ?low-in-Z ?act-B)
    (bind ?hi-in-Z ?act-A)
    else
    (IF (and (= ?minZA ?minZB)
             (= ?maxZA ?maxZB))
        THEN
        (bind ?low-in-Z BOTH)
        else

Appendix C. The Dynamic Sequencer Rules  298
(bind ?low-in-Z ?act-A)
(bind ?hi-in-Z ?act-B))

;* The following IF statements assert the logic
;* between activities based on the direction of
;* installation and the priority directions for
;* the class to which the activities are related

(IF (eq ?hi-1 X) ;* if the 1st prio is in X
  THEN
  (IF (not (eq ?low-in-X BOTH)))
  Then
  (IF (eq ?X-dir Y)
    THEN
    (assert (execute ?low-in-X ?hi-in-X GEOM 50))
    else
    (assert (execute ?hi-in-X ?low-in-X GEOM 50)))
  else

(IF (eq ?hi-2 Y) ;* if the 1st prio in X and the 2nd in Y
  THEN
  (IF (not (eq ?low-in-Y BOTH))
    THEN
    (IF (eq ?Y-dir Y)
      THEN
      else
    else
    (IF (not (eq ?low-in-Z BOTH))
      THEN
      (IF (eq ?Z-dir Y)
        THEN
        else
        (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
      else
        (assert (execute ?act-A ?act-B GEOM 50)))
    else
    ;* if the 1st prio in X and the 2nd in Z
  (IF (not (eq ?low-in-Z BOTH))
    THEN
    (IF (eq ?Z-dir Y)
      THEN
      else
      (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
    else
      (IF (not (eq ?low-in-Y BOTH))
        THEN
        (IF (eq ?Y-dir Y)
THEN
else
else
(assert (execute ?act-A ?act-B GEOM 50)))

;* if the 1st prio in Y

(IF (eq ?hi-1 Y)
THEN
(IF (not (eq ?low-in-Y BOTH))
Then
(IF (eq ?Y-dir Y)
THEN
else
else
;* if the 1st prio in Y and the 2nd in X

(IF (eq ?hi-2 X)
THEN
(IF (not (eq ?low-in-X BOTH))
THEN
(IF (eq ?X-dir Y)
THEN
(assert (execute ?low-in-X ?hi-in-X GEOM 50))
else
(assert (execute ?hi-in-X ?low-in-X GEOM 50))
else
(IF (not (eq ?low-in-Z BOTH))
THEN
(IF (eq ?Z-dir Y)
THEN
else
else
(assert (execute ?act-A ?act-B GEOM 50)))
else
;* if the 1st prio in Y and the 2nd in Z

(IF (not (eq ?low-in-Z BOTH))
THEN
(IF (eq ?Z-dir Y)
THEN
else
else
(IF (not (eq ?low-in-X BOTH))
THEN
(IF (eq ?X-dir Y)
THEN

Appendix C. The Dynamic Sequencer Rules
(assert (execute ?low-in-X ?hi-in-X GEOM 50))

else
 (assert (execute ?hi-in-X ?low-in-X GEOM 50))
else
 (assert (execute ?act-A ?act-B GEOM 50)))

;* if the 1st prio in Z

(IF (eq ?hi-1 Z)
 THEN
 (IF (not (eq ?low-in-Z Both))
  Then
 (IF (eq ?Z-dir Y)
    THEN
  else
  else
  ;* if the 1st prio in Z and 2nd prio in X

(IF (eq ?hi-2 Y)
 THEN
 (IF (not (eq ?low-in-Y BOTH))
  THEN
 (IF (eq ?Y-dir Y)
    THEN
  else
  else
 (IF (not (eq ?low-in-X BOTH))
   THEN
 (IF (eq ?X-dir Y)
    THEN
 (assert (execute ?low-in-X ?hi-in-X GEOM 50))
  else
 (assert (execute ?hi-in-X ?low-in-X GEOM 50))
  else
 (assert (execute ?act-A ?act-B GEOM 50))))

;* if the 1st prio in Z and 2nd prio in X

else
 (IF (not (eq ?low-in-X BOTH))
  THEN
 (IF (eq ?X-dir Y)
    THEN
 (assert (execute ?low-in-X ?hi-in-X GEOM 50))

else
 (assert (execute ?hi-in-X ?low-in-X GEOM 50))
else
 (IF (not (eq ?low-in-Y BOTH))
  THEN
 (IF (eq ?Y-dir Y)
THEN
else
else
   (assert (execute ?act-A ?act-B GEOM 50)))))))

Logic between Activities of the Same Class and Different Zones

(defrule logic-zone-same-class
   "Defining logic between activities of the same class and different zones"
   (declare (salience (- *fire1* 2000)))

   (schema ?act-A ;* Select the first activity
      (is-a activity)
      (act-zon-bor ?Z-bor)
      (act-class ?C)
      (min-X ?minXA)
      (max-X ?maxXA)
      (min-Y ?minYA)
      (max-Y ?maxYA)
      (min-Z ?minZA)
      (max-Z ?maxZA))

      (is-a activity) ;* Select the second activity
      (act-class ?C) ;* which in a zone that shares
      (act-zone ?Z-bor) ;* a common surface with the
      (min-X ?minXB) ;* first activity
      (max-X ?maxXB)
      (min-Y ?minYB)
      (max-Y ?maxYB)
      (min-Z ?minZB)
      (max-Z ?maxZB))

   (test (and (not (or (< ?maxXA ?minXB) ;* Test if the two activities
               (< ?maxYA ?minYB) ;* share a common surface
               (< ?maxZA ?minZB) ;* only. No overlapping
               (> ?minXA ?maxXB)
               (> ?minYA ?maxYB)
               (> ?minZA ?maxZB)))
   (or (< = ?maxXA ?minXB)
       (< = ?maxYA ?minYB)
       (< = ?maxZA ?minZB)
       (> = ?minXA ?maxXB)
       (> = ?minYA ?maxYB)
       (> = ?minZa ?maxZB))))

   (schema ?C ;* bind the name of the class and
      (is-a class);* and its slots values to variables
      (X-dir ?X-dir))
(Y-dir ?Y-dir)
(Z-dir ?Z-dir)
(hi-1 ?hi-1)
(hi-2 ?hi-2)

= >

;* Check which activity is lower and which is higher in the three X, Y, Z coordinates

(IF (or (> = ?minXA ?maxXB)
        (and (> ?minXA ?minXB)
             (> = ?maxXA ?maxXB))
        (and (= ?minXA ?minXB)
             (> ?maxXA ?maxXB))
        (and (> ?minXA ?minXB)
             (< = ?maxXA ?maxXB)))

THEN

(bind ?low-in-X ?act-B)
(bind ?hi-in-X ?act-A)
else

(IF (and (= ?minXA ?minXB)
         (= ?maxXA ?maxXB))

THEN

(bind ?low-in-X BOTH)
else

(bind ?low-in-X ?act-A)
(bind ?hi-in-X ?act-B))

(IF (or (> = ?minYA ?maxYB)
        (and (> ?minYA ?minYB)
             (> = ?maxYA ?maxYB))
        (and (= ?minYA ?minYB)
             (> ?maxYA ?maxYB))
        (and (> ?minYA ?minYB)
             (< = ?maxYA ?maxYB)))

THEN

(bind ?low-in-Y ?act-B)
(bind ?hi-in-Y ?act-A)
else

(IF (and (= ?minYA ?minYB)
         (= ?maxYA ?maxYB))

THEN

(bind ?low-in-Y BOTH)
else

(bind ?low-in-Y ?act-A)
(bind ?hi-in-Y ?act-B))

(IF (or (> = ?minZA ?maxZB)
        (and (> ?minZA ?minZB)
             (> = ?maxZA ?maxZB))
        (and (= ?minZA ?minZB)
             (> ?maxZA ?maxZB)))

THEN

(bind ?low-in-Z BOTH)
else

(bind ?low-in-Z ?act-B)
(bind ?hi-in-Z ?act-B))

Appendix C: The Dynamic Sequencer Rules

303
(and (> ?minZA ?minZB)
    (< = ?maxZA ?maxZB)))
THEN
(bind ?low-in-Z ?act-B)
(bind ?hi-in-Z ?act-A)
else
(IF (and (= ?minZA ?minZB)
        (= ?maxZA ?maxZB))
    THEN
    (bind ?low-in-Z BOTH)
    else
    (bind ?low-in-Z ?act-A)
    (bind ?hi-in-Z ?act-B))

:* The following IF statements assert the logic
:* between activities based on the direction of
:* installation and the priority directions for
:* the class to which the activities are related

(IF (eq ?hi-1 X) ;* if the 1st prio is in X
    THEN
    (IF (not (eq ?low-in-X BOTH))
        Then
        (IF (eq ?X-dir Y)
            THEN
            (assert (execute ?low-in-X ?hi-in-X GEOM 50))
            else
            (assert (execute ?hi-in-X ?low-in-X GEOM 50)))
        else
        (IF (eq ?hi-2 Y) ;* if the 1st prio in X and the 2nd in Y
            THEN
            (IF (not (eq ?low-in-Y BOTH))
                THEN
                (IF (eq ?Y-dir Y)
                    THEN
                    else
                else
                (IF (not (eq ?low-in-Z BOTH))
                    THEN
                    (IF (eq ?Z-dir Y)
                        THEN
                        else
                        (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
                    else
                    (assert (execute ?act-A ?act-B GEOM 50)))
                else
                ;* if the 1st prio in X and the 2nd in Z
(IF (not (eq ?low-in-Z BOTH))
  THEN
  (IF (eq ?Z-dir Y)
    THEN
    else
      (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
  else
    (IF (not (eq ?low-in-Y BOTH))
      THEN
        (IF (eq ?Y-dir Y)
          THEN
          else
        else
          (assert (execute ?act-A ?act-B GEOM 50)))
    ;* if the 1st prio in Y
  (IF (eq ?hi-1 Y)
    THEN
      (IF (not (eq ?low-in-Y Both))
        Then
          (IF (eq ?Y-dir Y)
            THEN
            else
          else
            ;* if the 1st prio in Y and the 2nd in X
      (IF (eq ?hi-2 X)
        THEN
          (IF (not (eq ?low-in-X BOTH))
            THEN
              (IF (eq ?X-dir Y)
                THEN
                  (assert (execute ?low-in-X ?hi-in-X GEOM 50))
                else
                  (assert (execute ?hi-in-X ?low-in-X GEOM 50)))
              else
                (IF (not (eq ?low-in-Z BOTH))
                  THEN
                    (IF (eq ?Z-dir Y)
                      THEN
                      else
                        (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
                    else
                      (assert (execute ?act-A ?act-B GEOM 50)))
                  ;* if the 1st prio in Y and the 2nd in Z
          (IF (not (eq ?low-in-Z BOTH)))
        ;* if the 1st prio in X and the 2nd in Z
      ;* if the 1st prio in X and the 2nd in Y
    ;* if the 1st prio in X and the 2nd in Z
  ;* if the 1st prio in Y and the 2nd in Z
)
THEN
  (IF (eq ?Z-dir Y)
    THEN
    else
      (assert (execute ?hi-in-Z ?low-in-Z GEOM 50)))
  else
    (IF (not (eq ?low-in-X BOTH))
      THEN
        (IF (eq ?X-dir Y)
          THEN
            (assert (execute ?low-in-X ?hi-in-X GEOM 50))
          else
            (assert (execute ?hi-in-X ?low-in-X GEOM 50)))
        else
          (assert (execute ?act-A ?act-B GEOM 50)))
      ;* if the 1st prio in Z
    )
(IFT (eq ?hi-1 Z)
  THEN
    (IF (not (eq ?low-in-Z Both))
      THEN
        (IF (eq ?Z-dir Y)
          THEN
          else
        else
          ;* if the 1st prio in Z and 2nd prio in X
        )
      )
    )
(IFT (eq ?hi-2 Y)
  THEN
    (IF (not (eq ?low-in-Y BOTH))
      THEN
        (IF (eq ?Y-dir Y)
          THEN
          else
        else
          (IF (not (eq ?low-in-X BOTH))
            THEN
              (IF (eq ?X-dir Y)
                THEN
                  (assert (execute ?low-in-X ?hi-in-X GEOM 50))
                else
                  (assert (execute ?hi-in-X ?low-in-X GEOM 50))
              else
                (assert (execute ?act-A ?act-B GEOM 50)))
            ;* if the 1st prio in Z and 2nd prio in X
            )
          )
        )
      )
    )
(IFT (not (eq ?low-in-X BOTH))
  THEN
    (assert (execute ?act-A ?act-B GEOM 50)))
THEN
(IF (eq ?X-dir Y)
 THEN
 (assert (execute ?low-in-X ?hi-in-X GEOM 50))
 else
 (assert (execute ?hi-in-X ?low-in-X GEOM 50)))
else
(IF (not (eq ?low-in-Y BOTH))
 THEN
 (IF (eq ?Y-dir Y)
 THEN
 else
 else
 (assert (execute ?act-A ?act-B GEOM 50)))
))

Logic between Activities of Different Classes and Different or Similar Zones

(defrule logic-clas-zone-diff
 "Defining logic between activities of different classes"
 (declare (salience (- ?*fire1* 1990)))

 (schema ?con ;* Finding a connection schema
 (is-a connect)
 (act-from ?act-A)
 (act-to ?act-B & ~ ?act-A)
 (con-type ?T))

 (schema ?act-A ;* Finding the act-from activity
 (is-a activity);
 (act-class ?C1)
 (min-X ?minXA)
 (max-X ?maxXA)
 (min-Y ?minYA)
 (max-Y ?maxYA)
 (min-Z ?minZA)
 (max-Z ?maxZA))

 (schema ?act-B ;* Finding the act-to activity
 (is-a activity);
 (act-class ?C2 & (not (eq ?C1 ?C2)))
 (min-X ?minXB)
 (max-X ?maxXB)
 (min-Y ?minYB)
 (max-Y ?maxYB)
 (min-Z ?minZB)
 (max-Z ?maxZB))

 (schema ?cls-rel ;* Finding a special class

Appendix C. The Dynamic Sequencer Rules

307
(is-a class-relation) ;* relation schema that exists
(con-type ?T) ;* between the two classes of the
(cls-1 ?CA & ?C1 | ?C2) ;* two activities
(rel-X ?relX)
(rel-Y ?relY)
(rel-Z ?relZ)
(cls-install ?install))

(schema ?dom-name) ;* Finding a special class
(is-a class-dominator) ;* relation schema that exists
(con-type ?T) ;* between the two classes of the
(cls-1 ?CE & ?C1 | ?C2) ;* two activities
(dominating ?cls-dom))

=>

;* Finding which activity is higher
;* in the X, Y, and Z directions

(IF (or
    (> = ?minXA ?maxXB)) ;* Checking higher in X
  (and (> = ?minXA ?minXB)
       (> = ?maxXA ?maxXB))
  (and (> = ?minXA ?minXB)
       (< = ?maxXA ?maxXB)))
THEN
(bind ?hi-in-X ?C1)
else
(bind ?hi-in-X ?C2))

(IF (or
    (> = ?minYA ?maxYB)
  (and (> = ?minYA ?minYB)
       (> = ?maxYA ?maxYB))
  (and (> = ?minYA ?minYB)
       (< = ?maxYA ?maxYB)))
THEN
(bind ?hi-in-Y ?C1)
else

(IF (or
    (> = ?minZA ?maxZB)
  (and (> = ?minZA ?minZB)
       (> = ?maxZA ?maxZB))
  (and (> = ?minZA ?minZB)
       (< = ?maxZA ?maxZB)))
THEN
(bind ?hi-in-Z ?C1)
else

(IF (and (eq ?relX ?hi-in-X) ;* Comparing the hi values with
         (eq ?relY ?hi-in-Y) ;* the special class relation
         ...)

Appendix C. The Dynamic Sequencer Rules
(eq ?relZ ?hi-in-Z)) ;* values in the X, Y, & Z dir
then  ;; ;* If the special class relation conditions applies,
;; ;* assert the logic based on the special class relation
(EIF (eq ?install ?C1)
  THEN
  (assert (execute ?act-A ?act-B GEOM 50))
  else
  (assert (execute ?act-B ?act-A GEOM 50)))
else
  ;; ;* If the special class relation doesn’t apply, assert
  ;; ;* logic based on the dominating class relation
(EIF (eq ?cls-dom ?C1)
  THEN
  (assert (execute ?act-A ?act-B GEOM 50))
  else
  (assert (execute ?act-B ?act-A GEOM 50))))
Appendix C4: Preliminary Logic Refinement Rules

Refine Same Logic Relations:
(defrule refine-same-logic-diff-priority
  "Retracting Execution Facts of similar logic and lower priorities"
  (declare (salience (- ?*fire1* 5000)))
  (test (not (eq ?prio1 ?prio2)))
  =>
  (IF (> ?prio1 ?prio2)
    THEN
    (retract ?f2)
    else
    (retract ?f1)))))

Refine Opposite Logic Relations:
(defrule refine-opposite-diff-priority
  "Retract Execution Facts of opposite logic and lower priorities"
  (declare (salience (- ?*fire1* 5000)))
  =>
  (IF (> ?prio1 ?prio2)
    THEN
    (retract ?f2)
    else
    (IF (> ?prio2 ?prio1)
      THEN
      (retract ?f1)
      else
Appendix C5: Creating Link, O-Node, and H-Node Schemata

Creating the Link (arrow or logic) Schemata Structure:
(defrule create-links-schemata
  "Creating a LINK schema for Each Execution Fact"
  (declare (salience (- ??fire1* 6000)))
  (bind ?LNK (string-to-symbol (string-append "LNK-" ?ACT-A "-" ?ACT-B)))
  (schema ?LNK)
  (assert (schema ?LNK) ; ;* For each execution fact a
    (is-a LNK) ; ;* link schema is created.
    (from-act ?ACT-A) ; ;* The link schema has slots
    (to-act ?ACT-B) ; ;* that stores the name of the
    (type ?TYPE) ; ;* activities that are related
    (prio ?PRI0); ;* by the link. The priority of
    (used NO) ; ;* the link. And the status of the
    (opposite ))); ;*in the final plan.
  (assert (schema control-data (prio-data ?prio)))
  (retract ?fl))

Opposite Links Association Rule:
(defrule assert-opposite-slot-in-links
  "Asserting the value of the opposite slot in the link schemata"
  (declare (salience (- ??fire1* 6100)))
  (bind ?LNK1 (string-to-symbol (string-append "LNK-" ?ACT-A "-" ?ACT-B)))
  (bind ?LNK2 (string-to-symbol (string-append "LNK-" ?ACT-B "-" ?ACT-A)))
  (assert (schema ?LNK1
    (opposite ?LNK2)) (retract ?fl))

Creating O-Node and H-Node Schemata Structures:
(defrule create-orig-and-hierarchy-nodes
  "Creating the O-Nodes and H-Nodes Schemata"
  (declare (salience (- ??fire1* 5000)))
  (schema ?ACT (is-a activity))
  = >
  (bind ?O-NODE (string-to-symbol (string-append "O-N-" ?ACT)))
  (schema ?O-NODE) ; ;* For each activity create an O-Node
  (assert (schema ?O-NODE
    (is-a O-NODE) ; ;* Assert the slots of the O-node
    (act ?ACT)
    (act-before)
    (act-after)
    (init-act-before)
    (init-act-after)
    (status)))
  (bind ?H-NODE (string-to-symbol (string-append "H-N-" ?ACT)))
  (schema ?H-NODE) ; ;* For each activity create an H-Node
  (assert (schema ?H-NODE (this-is H-NODE)
    (act ?ACT) (ancestor-N)))

Appendix C. The Dynamic Sequencer Rules 311
Appendix C6: Final Logic Refinement Reasoning Process

Maximum Priority Selection Rule:
(defrule find-max-prio-and-fix-links
 "Finding the Maximum priority value to be used in the 1st Priority Cycle"
 (declare (salience (- ?*fire1* 6900)))
 (schema control-data
   (initialized NO)) = >
 (IF (not (slot-null control-data prio-data))
   THEN
   (bind ?maxi 0)
   (FOR ?value in-slot-values-of control-data prio-data
     DO
       (IF (> ?value ?maxi)
         THEN
         (bind ?maxi ?value))
     (modify-schema-value control-data prio-max ?maxi)
     (modify-schema-value control-data initialized YES)
   (assert (to-recycle YES))))

Initial Logic assertion Rule (defrule assert-all-initial-logic
 "Initialize the schedule of the priority cycle"
 (declare (salience (- ?*fire1* 7000)))
 (schema ?LNK
   (is-a LNK) ;* only the links of priority value
   (from-act ?act-A) ;* equal the current priority of the
   (to-act ?act-B) ;* priority cycle are considered for
   (prio ?prio)); ;* analysis at the current cycle.
 (schema control-data ;* The original nodes of the
   (prio-max ?maxi)) ;* activities related by these links
 (test (eq ?maxi ?prio)) ;* are initialized with their
 (to-recycle yes) ;* init-act-after & init-act-before
 = >
 (bind ?O-NODE-1 (string-to-symbol (string-append "O-N--" ?act-A)))
 (bind ?O-NODE-2 (string-to-symbol (string-append "O-N--" ?act-B)))

Creating the Opened and Closed List Facts Rule:
(defrule define-open-close-node-lists
 "Create the opened-list and closed-list facts"
 (declare (salience (- ?*fire1* 7050)))
 (schema control-data
   (initialized YES))
 = >
 (assert (opened-list))
 (assert (closed-list))
 (assert (check-first-element opened-list)))

Finding the First Element in the Opened-List:
(defrule find-start-activities
"Find all start activities based on the selected links of the cycle"
(declare (salience (- ?*firel* 7100)))
?f1 < - (opened-list . ?OL)
(schema ?O-NODE
  (is-a O-NODE)
  (INIT-ACT-BEFORE)
  (act ?ACT))
(test (not (member. ?ACT ?OL)))
(test (slot-null ?O-NODE init-act-before))
(test (slot-null ?O-NODE status)) ;*** to avoid run loops
= >
(retract ?f1)
(assert (opened-list . ?OL ?ACT))
(assert (schema ?O-NODE
  (status start-act)))

Checking the First Element in the Opened-List:
(defrule check-first-element-in-open-list
  "Check if the first member in the opened-list is a start activity"
  (declare (salience (- ?*firel* 7200)))
?f1 < - (check-first-element opened-list)
?f2 < - (opened-list ?first-element . ?OL)
= >
(retract ?f1)
(bind ?O-NODE (string-to-symbol (string-append "O-N-" ?first-element)))
(IF (not (slot-null ?O-NODE init-act-before))
  THEN
  (retract ?f2)
  (retract-schema-value ?O-NODE status start-act)
  (assert (opened-list . ?OL)))

Final Logic Refinement Reasoning Rule:
(defrule logic-refinement
  "Asserting the final logic - initialize a priority cycle"
  (declare (salience (- ?*firel* 8000)))
?f1 < - (opened-list ?ACT-A . ?REM-OL)
?f2 < - (closed-list . ?CL)
(to-recycle YES)
= >
(bind ?O-N-A (string-to-symbol (string-append "O-N-" ?ACT-A)))
(bind ?H-N-A (string-to-symbol (string-append "H-N-" ?ACT-A)))
(bind ?prio (get-schema-value control-data prio-max))
(FOR ?ACT-B in-slot-values-of ?O-N-A init-act-after DO
  (bind ?LNK (string-to-symbol (string-append "LNK-" ?ACT-A ";" ?ACT-B)))
  (IF (eq ?prio (get-schema-value ?LNK prio))
      THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-append "O-N-" ?ACT-B)))
      (bind ?H-N-B (string-to-symbol (string-append "H-N-" ?ACT-B)))
      (IF (not (schema-value-p ?H-N-A is-a ?H-N-B))
       THEN
      (bind ?O-N-B (string-to-symbol (string-app
THEN
(bind ?case2 NO)
    ;* first case of implied logic
(FOR ?B-parent in-slot-values-of ?O-N-B act-before
    DO
            THEN
                (bind ?sname (string-to-symbol (string-append
                    "LNK-DEL-" ?B-parent "-" ?ACT-B)))
                (schemac ?sname)
                (assert (schema ?sname
                    (is-a LNK-DEL)
                    (from-act ?B-parent)
                    (to-act ?ACT-B))))
            ;* second case of implied logic
        (IF (schema-value-p ?H-N-B ancestor-N ?ACT-A)
            THEN
                (bind ?case2 YES))
            ;* third case of implied logic
        (FOR ?H-N-B-descendent in-schema-descendents-of ?H-N-B
            DO
                (bind ?B-descendent (get-schema-value
                    ?H-N-B-descendent act))
                    THEN
                        (bind ?sname (string-to-symbol (string-append
                            "LNK-DEL-" ?ACT-A "-" ?B-descendent)))
                        (schemac ?sname)
                        (assert (schema ?sname
                            (is-a LNK-DEL)
                            (from-act ?ACT-A)
                            (to-act ?B-descendent))))
                (IF (eq ?case2 NO)
                    THEN
                        (modify-schema-value ?LNK used YES)
                        (assert (schema ?O-N-A (act-after ?ACT-B)))
                        (assert (schema ?O-N-B (act-before ?act-A)))
                        (assert (schema ?H-N-B (is-a ?H-N-A)
                            (ancestor-N ?act-A))))))
            ;* modifying the state description
        (FOR ?sname in-schema-children-of LNK-DEL
            DO
                (bind ?A-from (get-schema-value ?sname from-act))
                (bind ?A-to (get-schema-value ?sname to-act))
                (bind ?LNK-NO (string-to-symbol (string-append
                    "LNK-" ?A-from "-" ?A-to)))
                (bind ?O-N-from (string-to-symbol (string-append "O-N-" ?A-from)))
                (bind ?H-N-from (string-to-symbol (string-append "H-N-" ?A-from)))
                (bind ?O-N-to (string-to-symbol (string-append "O-N-" ?A-to)))
                (bind ?H-N-to (string-to-symbol (string-append "H-N-" ?A-to)))
                (modify-schema-value ?LNK-NO used NO)

Appendix C. The Dynamic Sequencer Rules
(retract-schema-value ?O-N-from act-after ?A-to)
(retract-schema-value ?O-N-to act-before ?A-from))
(FOR ?sname in-schema-children-of LNK-DEL DO
  (retract ?sname))
(retract ?F2)
(assert (closed-list .?CL ?ACT-A))
(bind ?act-may-open (get-schema-value ?O-N-A act-after))
(FOR ?value in. ?act-may-open DO
  (IF (and (not (member. ?value ?CL))
    (not (member. ?value ?REM-OL))) THEN
    (put-schema-value control-data may-data ?value)))
(bind ?open-ok (get-schema-value control-data may-data))
(retract-all-schema-values control-data may-data)
(retract ?F1)
(assert (opened-list .?REM-OL .?open-ok))

Next Priority Value Selection Rule:
(defrule select-new-prio-value
  "Selecting a new priority value for the next priority cycle"
  (declare (salience (- ?*fire1* 9000)))
  ?F1 <- (opened-list ?)
  ?F2 <- (closed-list ?)
  (schema control-data
    (initialized YES)
    (prio-max ?maxi))
  ?F3 <- (to-recycle YES)
= >
  (retract ?F1)
  (retract ?F2)
  (retract ?F3)
  (retract-schema-value control-data prio-max ?maxi)
  (retract-schema-value control-data prio-data ?maxi)
  (assert (clean init-and-start-cond)))

Retracting Data from the O-Node Schemata Rule:
(defrule clean-init-and-start-cond
  "Preparing for the next priority cycle"
  (declare (salience (- ?*fire1* 9200)))
  ?F1 <- (clean init-and-start-cond)
  (schema O-node)
= >
  (FOR ?O-node in-schema-children-of O-node DO
    (retract-all-schema-values ?O-node init-act-before)
    (retract-all-schema-values ?O-node init-act-after)
    (IF (schema-value-p ?O-node status start-act) THEN
      (retract-schema-value ?O-node status start-act)))
  (retract ?F1)
  (modify-schema-value control-data initialized NO))
Appendix C7: Miscellaneous Definitions

(defglobal ?*fire1* = 10000) ;* for salience declaration

(defschema ACTIVITY) ;* parent schema for activities

(defschema CLASS) ;* parent schema for classes

(defschema ZONE) ;* parent schema for zones

(defschema CONNECT) ;* parent schema for connections

(defschema CLASS-DOMINATOR) ;* parent schema for class dominators

(defschema CLASS-RELATION) ;* parent schema for class relations

(defschema LNK) ;* parent schema for links

(defschema ACT-CLASS
  (instance-of relation)
  (cardinality multiple)) ;* slot as a relation (multiple)

(defschema ACT-ZONE
  (instance-of relation)
  (cardinality multiple)) ;* slot as a relation (multiple)

(defschema ACT-ZON-BOR
  (instance-of relation)
  (cardinality multiple)) ;* slot as a relation (multiple)

(defschema CON-TYPE
  (instance-of relation)
  (cardinality multiple)) ;* slot as a relation (multiple)

(defschema act-before
  (instance-of relation)
  (cardinality multiple)
  (inherits no))

(defschema act-after
  (instance-of relation)
  (cardinality multiple)
  (inherits no))

(defschema init-act-before
  (instance-of relation)
  (cardinality multiple)
  (inherits no))

(defschema init-act-after
  (instance-of relation)
(defschema ancestor-N
  (instance-of relation)
  (cardinality multiple)
  (inherits YES))

(defschema LNK-DEL)

(defschema may-data
  (instance-of slot)
  (cardinality multiple))

(defschema prio-data
  (instance-of slot)
  (cardinality multiple))

(defschema act-list
  (instance-of relation)
  (cardinality multiple))

(defschema control-data
  (may-data)
  (prio-data)
  (prio-max)
  (act-list)
  (project-finish 0)
  (initialized NO))
Appendix C8: Object-Oriented Schedule Calculation Process

ART Function (METHOD) for Early Schedule Calculation:
(def-art-fun to-schedule-early (?act)
  (bind ?start 0)
  (bind ?dur (get-schema-value ?act act-dur))
  (bind ?O-node (string-to-symbol (string-append "O-N-" ?act)))
  (FOR ?act-P in-slot-values-of ?O-node act-before
     DO
     (bind ?early-time (get-schema-value ?act-P early-finish))
     (IF (> ?early-time ?start)
      THEN
      (bind ?start ?early-time))
     (modify-schema-value ?act early-start ?start)
     (modify-schema-value ?act early-finish (+ ?start ?dur))
     (bind ?proj-finish (get-schema-value control-data project-finish))
     (IF (> (+ ?start ?dur) ?proj-finish)
      THEN
      (modify-schema-value control-data project-finish
       (+ ?start ?dur)))
  )

ART Function (Method) for Late Schedule Calculation:
(def-art-fun to-schedule-late (?act)
  (bind ?finish (get-schema-value control-data project-finish))
  (bind ?dur (get-schema-value ?act act-dur))
  (bind ?O-node (string-to-symbol (string-append "O-N-" ?act)))
  (FOR ?act-S in-slot-values-of ?O-node act-after
     DO
     (bind ?late-time (get-schema-value ?act-S late-start))
     (IF (< ?late-time ?finish)
      THEN
      (bind ?finish ?late-time))
     (modify-schema-value ?act late-finish ?finish)
     (modify-schema-value ?act late-start (- ?finish ?dur))
  )

Rule for Early Schedule Calculation:
(defrule send-for-early-schedule
 "Early schedule calculation rule"
 (DECLARE (SALIENCE (- ?*FIRE* 9501)))
 = >
 (FOR ?schema-child in-schema-children-of activity
   DO
   (assert (schema control-data (act-list ?schema-child))))
   (while (not (slot-null control-data act-list))
     DO
     (FOR ?act in-slot-values-of control-data act-list
       DO
       (bind ?O-node (string-to-symbol (string-append "O-N-" ?act)))
       (bind ?testing OK)
       (FOR ?act-P in-slot-values-of ?o-node act-before
         DO
         )
       )
   )
)

Appendix C. The Dynamic Sequencer Rules
(IF (schema-value-p ?act-P scheduled-early NO)
  THEN
  (bind ?testing FAIL))
(IF (eq ?testing OK)
  THEN
  (send calc-early ?act)
  (modify-schema-value ?act scheduled-early Yes)
  (retract-schema-value control-data act-list ?act)))

Rule for Late Schedule Calculation:
(defrule send-for-late-schedule
  "Late schedule calculation rule"
  (DECLARE (SALIENCE (- ?*FIRE1* 9502)))
  = >
  (FOR ?schema-child in-schema-children-of activity DO
    (assert (schema control-data (act-list ?schema-child))))
  (while (not (slot-null control-data act-list))
    DO
    (FOR ?act in-slot-values-of control-data act-list DO
      (bind ?O-node (string-to-symbol (string-append "O-N-" ?act)))
      (bind ?testing OK)
      (FOR ?act-S in-slot-values-of ?o-node act-after DO
        (IF (schema-value-p ?act-S scheduled-late NO)
          THEN
          (bind ?testing FAIL))
        (IF (eq ?testing OK)
          THEN
          (send calc-late ?act)
          (modify-schema-value ?act scheduled-late Yes)
          (retract-schema-value control-data act-list ?act))))
Appendix C9: The Schedule Reporting Rule

(defglobal ?*log-file* = (open "logic.dat" "w"))
(defglobal ?*sch-date-file* = (open "schdate.dat" "w"))

(defrule schedule-reporting-rule
  "Rule that generates the two ASCII files needed in the simulation"
  (declare (salience (- ?*FIRE1* 9800)))
  = >
   (FOR ?act in-schema-children-of activity
    DO
      (bind ?ES (get-schema-value ?act early-start))
      (bind ?LS (get-schema-value ?act late-start))
      (bind ?EF (get-schema-value ?act early-finish))
      (bind ?LF (get-schema-value ?act late-finish))
    (close ?*sch-date-file*))
  (FOR ?link in-schema-children-of lnk
    DO
      (bind ?stat (get-schema-value ?link used))
      (IF (eq ?stat YES)
        THEN
          (bind ?to (get-schema-value ?link to-act))
          (bind ?from (get-schema-value ?link from-act))
          (printf ?*log-file* "%-8S%-8S\n" ?FROM ?TO))
      (close ?*log-file*))
Vita

The author was born on February 18, 1961 in Kuwait. He received his Bachelor of Science degree (with honors) in Civil Engineering from Kuwait University in 1982. From 1982 to 1983, he joined an A/E consulting firm (Kuwaiti Engineer’s office) as a Time and Cost Engineer. In September 1983, he started to pursue graduate studies at the Center of Construction Research and Education (CCRE) at Massachusetts Institute of Technology (M.I.T.) where he was awarded a Master of Science degree in Civil Engineering in June 1984. From 1984 to 1985, he joined a construction management firm in Kuwait as a Control Systems Manager. From 1985 to 1987, he rejoined the Kuwaiti Engineer’s Office as a Contract Administrator. In September 1987, he continued his graduate studies in Construction Engineering and Management at Virginia Polytechnic Institute and State University (VPI&SU) where he was awarded a Doctor of Philosophy degree in Civil Engineering in May 1990.