Automation and Robotics in Construction XVI © 1999 by UC3M

Design of an Object Oriented Environment for Planning Mobile Crane Lifts

K. Govinda Rao
Project Associate,
Dept of Civil Eng.
I.I.T. Madras, India

Koshy Varghese
Assistant. Professor,
Dept. of Civil Eng.
I.I.T. Madras, India

N. Ramesh Babu
Associate Professor,
Dept. of Mech. Eng.
I.I.T. Madras, India

K.N. Satyanarayana
Assistant. Professor,
Dept. of Civil Eng.
I.I.T. Madras, India

Abstract

This paper describes the work done towards designing a computer aided system for planning crane operations on a construction site using object-oriented programming concepts. The use of object-oriented programming concepts for design is important as it facilities robust design, distributed development, modularity, reusability and easy extension of features. Based on the planning and representation requirements, an object-oriented analysis and design was done using the Booch method. The platform selected for implementation was AutoCAD Rel.14 with the AutoCAD Runtime Extension (ObjectARX) in the Windows 95/NT operating system. This platform was found to support all important object-oriented constructs as well as the component object model (COM) which ensures plug-in capability and independent extensibility for various modules of the system. A detailed architecture of the system components within this environment has been developed and potential usage of this system is illustrated.

1.0 Introduction

On industrial projects, the use of cranes to perform heavy lifts is a common activity. These lifts involve the installation/replacement of plant equipment such as compressors weighting 30t to large process columns weighing up to 800t. As industrial sites are congested with closely space equipment, overhead and underground piping, and structural steel, the planning and execution of the lift can be a complex task.

Past work in the area has addressed issues on identifying the requirements of heavy lift planning systems, and developing software environments to assist the planner [Brown & Root 91][Bennet 94][Varghese 97]. However, as most of these systems were developed on propriety platforms, their usage is limited. Further, the design and implementation of these systems are based on conventional concepts and technologies that are now being replaced by object oriented design concepts and implementation environments.

The objective of this work is to develop a computer aided heavy lift-planning environment using object-oriented design methodologies and tools which support the implementation of the design. A planning environment implemented based on object-oriented concepts will posses a robust design and facilitate the extension of system features without modification to the basic design decisions. A detailed coverage of object oriented design concepts are presented by Booch [Booch 94]. In an attempt to make the environment easily available to planners, it was decided to develop it on a hardware/software platform that is widely available in construction organizations.

This paper is organized into six sections. The following section reviews the past work done in computer aided planning of crane lifts. In the third section, the various classes required for the system and relationships between classes are discussed. Next, the development environment selected and its features are discussed and an implementation architecture for the system is presented. The fifth section illustrates the usage of the system and finally the conclusions drawn from the current effort and focus of future work are presented.

2.0 Past Work

A computer aided lift planning system called "Computer Aided Rigging" (CAR) was developed by a team of programmers and experienced lift planners at Brown & Root [CAR 90] [Alexander 92]. This system was developed within the Microstation CAD package using the Microstation Development Language {MDL}. Using the system interface a user can enter the dimension of the load and radius of lift and the system will list the cranes which can lift the load without exceeding capacity and interfering with the load. In addition, the rigging details can also be designed by the system. A drawing of the lift configuration for field planning is also generated by the system.

In a concurrent effort, research was conducted at the University of Texas at Austin to classify the components of a lift plan, identify the features required of a Heavy Lift Planning Systems and developing a prototype Heavy Lift Planning System called HELPS [Wolfhope 91] [Hornaday 93] [Varghese 97]. This system was developed in the visualization environment Walkthru. Features of Walkthru were enhanced to support heavy lift planning. HELPS permitted the user to select different crane, site and load configurations and

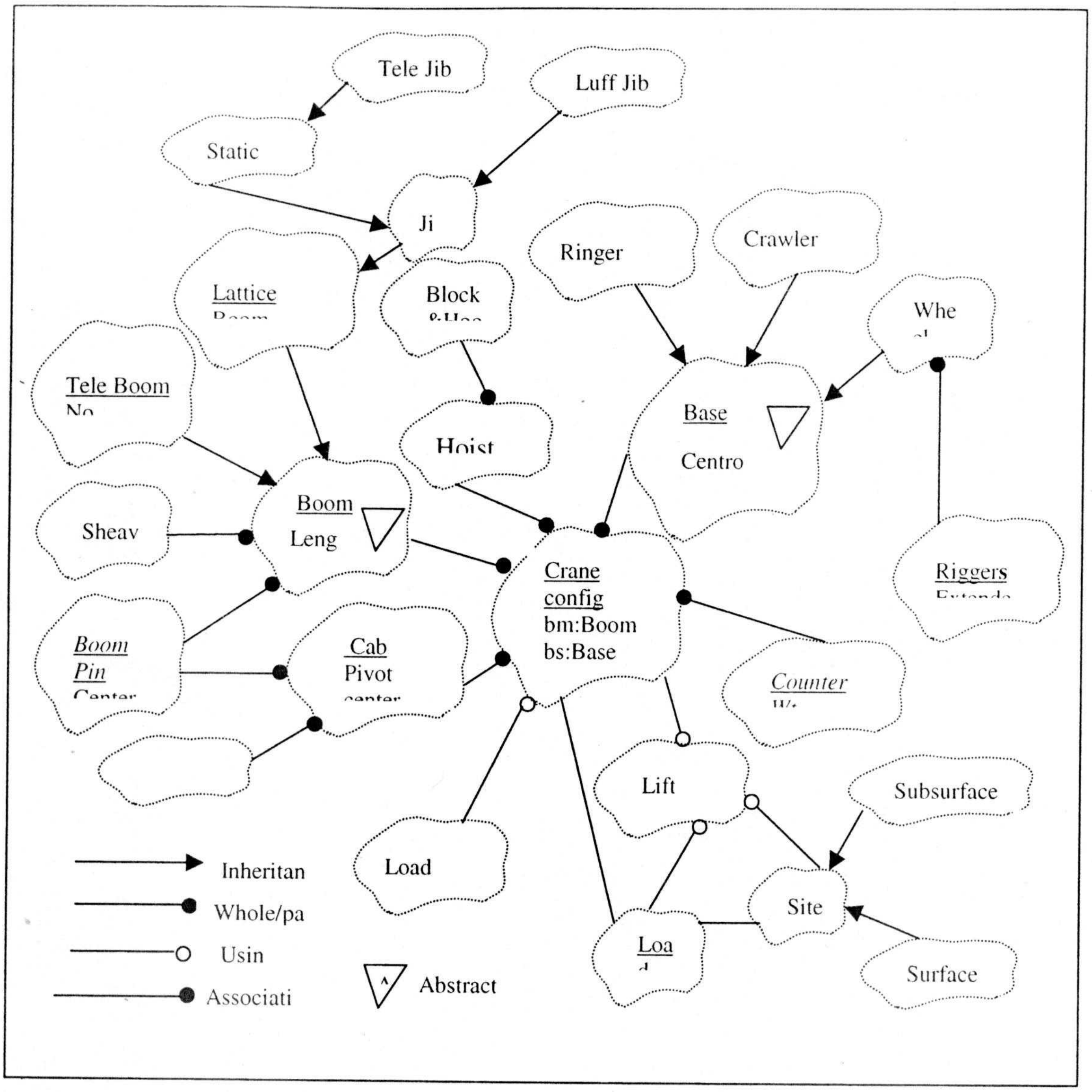


Figure 1: System Design

simulate the entire lift operation while monitoring the lift parameters.

At Bechtel, a lift planning software named Automated Lift Planning System (ALPS) was developed using a customized visualization environment [Benett and Ditlinger 94]. This software also can be used to design the rigging assembly for a given load, select a suitable crane, and visualize the lift environment for making decisions on the lift parameters.

Further research at the University of Texas resulted in the development of an interactive planning system for optimizing multiple heavy lifts on a single job site [Lin 96]. This system was developed in the Microstation environment. It incorporates algorithms that assist the planner to locate the crane from a position where a maximum number of multiple lifts can be performed.

While the basic features of the above system are relevant to lift planning domain, the incorporation of all the decision support and automation features required for lift planning is a immense task and has to be incorporated in an incremental fashion. The software design and architectures of the above systems are based on conventional software design methodologies. This limits the ease with which new features and algorithms can be added to the software. By utilizing an object-oriented approach, it is expected that the translation of the problem requirements to software structure will be more natural, the ensuring design more robust and the ease of extensibility enhanced.

3.0 System Design

The preliminary step in developing an object oriented design is to identify the classes or "key abstractions", which are required to represent the problem. The classes are identified based on the presence of physical objects, roles, events and interactions that are a part of the problem domain. Next, the main attributes of the classes are identified. These attributes reflect the properties, which the object instantiated from the class will possess. Finally, the relationships between the classes are defined. Figure-1 shows a representation of the design.

There are three key physical systems interacting in the lift planning process. They are the crane, the load and the site. Each of these is a high level abstraction in the representation. Physically a crane is a composition of sub components such as the base, cab, boom, counterweights, line, etc. The sub-components have a "has a" (aggregation) relationship with the crane class and are modeled as classes themselves. The aggregation relationship is used further to model the hierarchical decomposition of the crane components to the level of detail required for planning the lift.

Some of sub-components in a crane can have specialized properties. For example, all cranes have booms but a particular boom has to be a lattice-boom or telescoping boom. Each type has distinct properties and behavior. Similarly the crane base is classified into different categories such as crawler, truck mounted, ringer etc. To capture these relationships the inheritance concept is used. The general category (e.g. boom) is represented as a parent class containing the common properties that will be inherited by the child classes. The specialized properties of the child classes (e.g. number of inserts for a lattice boom) are specified in the child class. To take advantage of polymorphism the parent classes are defined as abstract classes (in the c++ context). This allows common behaviors such as luffing & slewing to be inherited from the parent class while specialized behavior (e.g. extend_boom) to be declared as a virtual function in the parent class and redefined to exhibit specific behavior in the child classes.

A given crane model can have different configurations based on, boom attachments, counterweights, special attachments etc. The load chart used to determine the lifting capacity of a crane depends on the specific configuration. For example, the popular Manitowoc 4100W series of cranes has numerous configurations based on crawler position, boom specification (open throat etc.), jib attachments (static, luffing), base type (crawler, ringer), counterweight loaded and special attachments such as Max-Er. Once the configuration is fixed, the capacity depends on boom length and lift radius. Thus, the load chart is also abstracted to a class, which is a component of the crane configuration.

The lift area generally comprises of obstacles, which are on the surface or under the surface. The surface obstacles interfere directly with the lift operation, while the sub-surface obstacles affect the lift operation only when they are overloaded. The general attributes required to model surface obstacles include location, allowable clearance and installation schedule. For the sub-surface obstacles, attributes such as allowable stresses and soil properties have to be represented instead of allowable clearance. These two categories of obstacles are considered to make up the lift area. A separate class called Site has been defined to aggregate these obstacles.

The object, which is to be installed in the lifting operation, is represented as the class Load. This class undergoes three states during the planning process. It is initially an independent object at the pick location. Whenever the object is attached to the crane or placed in it's final position; it becomes a part of the crane or the site respectively. The relationships between the classes are represented in the Figure-1 by the association symbol.

The lift plan is developed using a selected crane configuration, load and lift area. The existence of the plan is encapsulated in the form of a separate class called Lift Plan. The basic attributes of this class include planning parameters such as crane location, pick location, lift path, minimum clearances etc. Different plan scenarios (objects) can be created from the Plan class. Each scenario can contain a different set of plan attributes for the same crane-site-load combination.

It can be seen that design using object-oriented concepts provides a natural representation of the lift-planning situation. As a result, the likelihood of changes in the basic representation when new features are added to the software is lower. It should be noted that in all of the physical classes identified above, shape is also an important attribute. Further no classes relating to the user-interface was discussed, as it is an implementation issue. The translation of this representation into a specific software implementation depends on the features of the development environment and the encoding language.

4.0 Development Environment

As discussed in previous studies, an integrated software system which can assist the lift planner to generate and test alternate lift plans will enhance planning productivity. The system should be implemented in an environment which can support object-oriented development and be easily available to construction planners. As the AutoCAD package is widely used in construction organizations and supports customization, it was decided to explore the capabilities of this package for developing the lift planning system. Numerous development alternatives are a vailable within AutoCAD. These include Auto Lisp, AutoCAD Development System (ADS),

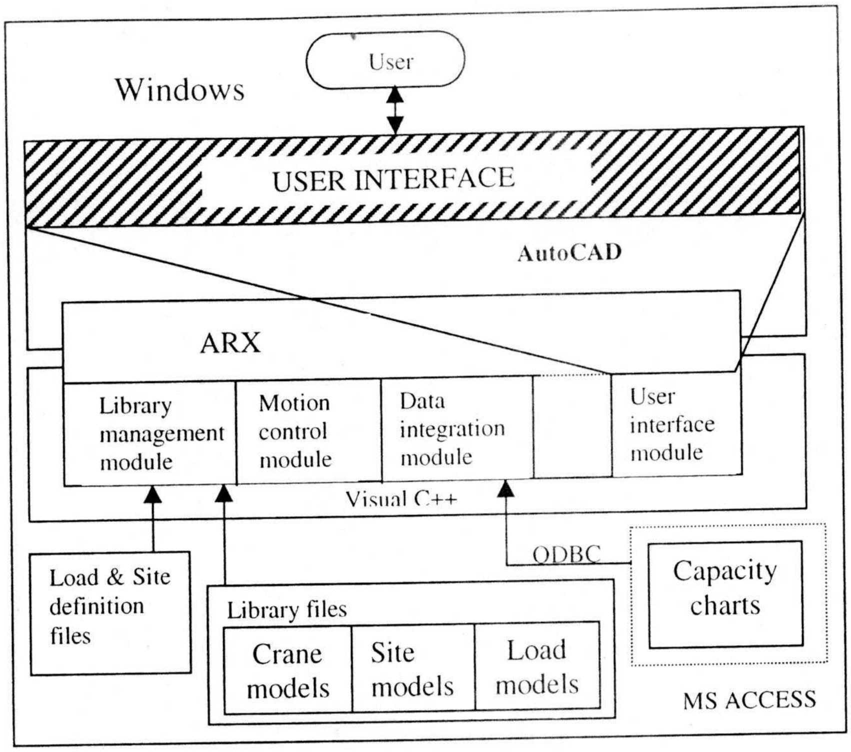


Figure 2: System Architecture

Visual Lisp, Visual Basic for applications (VBA), Visual Basic and Object ARX.

AutoLisp: AutoLisp is an artificial intelligence language and it is not generally well known outside of two specific communities- artificial intelligence and AutoCAD. AutoLisp is slow in execution and take even days to execute the complex programs.

ADS: It is C interface to AutoCAD. C is not inherently "object-oriented". The interface to the AutoCAD database is slower than with ObjectARX and C++.

Visual Lisp: It is an enhancement of the mature AutoLisp. Built with ObjectARX programming technology, Visual Lisp for AutoCAD Release 14 provides a new set of advanced, object-oriented customization tools. It dramatically reduces AutoLisp development time. The Visual Lisp compiler creates an ObjectARX application from your Lisp source files that loads and executes 3 to 10 times faster AutoLisp. It have the advanced features what ObjectARX have.

VBA: VBA enables customers to use AutoCAD Release 14 ActiveX Automation features. VBA is significantly faster than AutoLisp. The execution speed is very close to a compiled C++ ObjectARX. But it is not object-oriented.

Of these, the options which supported object-oriented development were VisualLisp and ObjectARX. Of these, ObjectARX was selected for developing the lift planning system, as it is the native environment for AutoCAD and utilizes the widely available language C++.

4.1 Object ARX Details

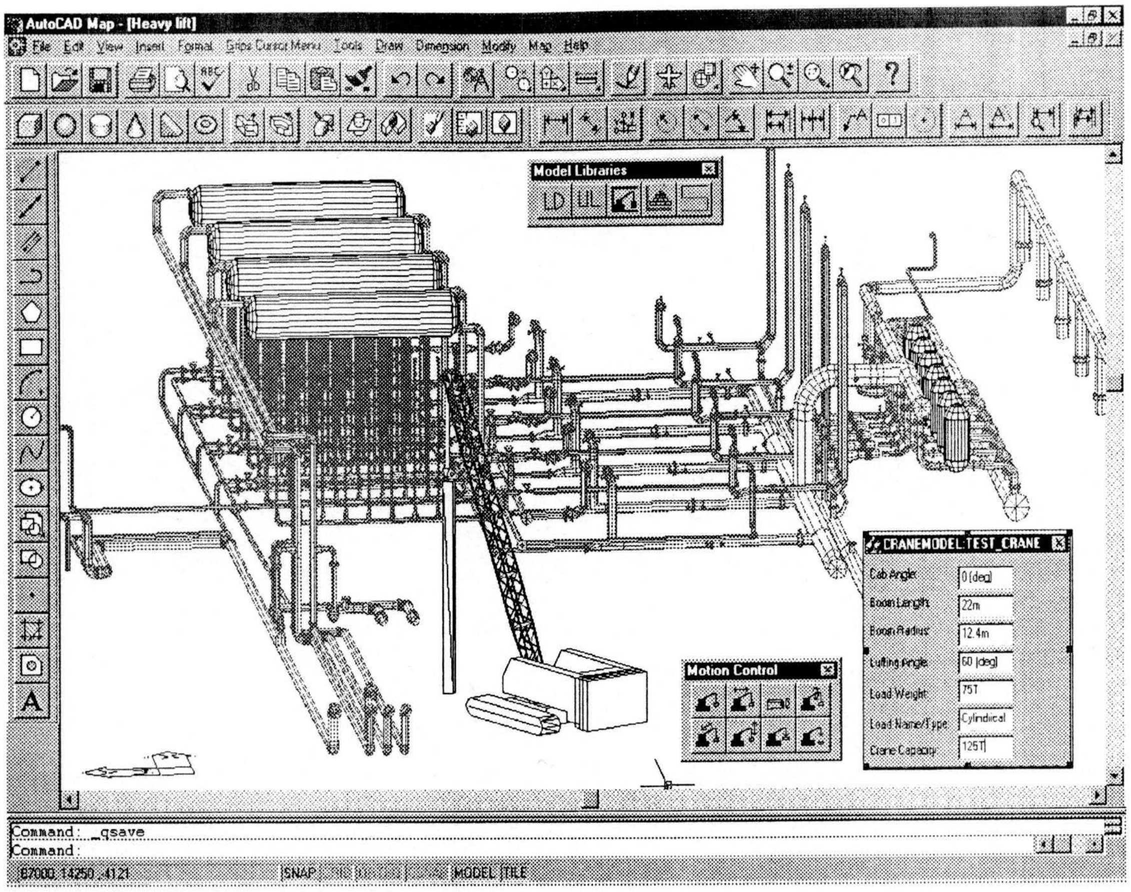
This is the product developed by Autodesk to program, customize and extend the capabilities of AutoCAD using the C++ language in a Windows environment. ARX functions are created as Dynamic Link Libraries (DLL file). These libraries are independent program modules, which can be loaded/unloaded into/from AutoCAD implicitly or explicitly. When these libraries are loaded into AutoCAD, the functions coded in the DLL files can be called from within AutoCAD thus extending its functionality. The entry point to the DLL files for an application has to be coded based on the specifications required by that application.

Dynamic Link Library files created using ARX specifications have a .ARX extension. They are usually created using appropriate settings in the Visual C++ development environment. These settings include the location of the Application Program Interface (API) for AutoCAD. The functions are then coded and compiled in the conventional manner. Once compiled, they can be distributed to other Win platforms and loaded into AutoCAD without any further compilation.

Development using ObjectARX permits the implementation of Object-Oriented Constructs in the C++ language. It permits the definition of custom classes and reactors. The custom classes can inherit properties from both the Microsoft Foundation Classes (MFC) as well as the classes defined within the AutoCAD package.

Class Libraries in Object ARX

The important class libraries available to ObjectARX modules are given below. The applicability of these libraries in the context of implementing the lift planning system is also discussed.


AutoCAD Map - [Heavy lift]
File Edit View Insert Format Grips Cursor Menu Tools Draw Dimension Modify Map Help
Model Libraries
LD UL
Motion Control
CRANEMODEL:TEST_CRANE
SNAP
MODEL TILE

Cab Angle	0 [deg]
Boom Length	22m
Boom Radius	12.4m
Lufting Angle	60 [deg]
Load Weight	75T
Load Name/Type	Cylindical
Crane Capacity	125T

```
Command: _qsave
Command:
87000,14250,-4121
```

Figure 3: System Interface

AcRx: This is a top-level library of classes used for runtime class registration and identification. It provides a set of C++ macros to declare and define new ARX classes, which are always derived from AcRx Object class. For example, while deriving a new class from any of the AutoCAD standard entity classes ACRX_DECLARE_MEMBERS () macro is used in the declaration of classes that are to be a part of the ARX runtime tree.

AcEd: It is a library of classes for registering native commands. To register the user-friendly commands for the present application, it is necessary to use AcEd functions. For example, to make the crane travel in a particular direction, we can use our own commands such as "MOVECRANE" by selecting appropriate items from the interface.

AcGi: This is a graphics interface library classes, used to draw AutoCAD entities. This class is needed to display the drawing entity dynamically when it is subjected to any standard transformations like moving, rotating, scaling etc.

AcDb: This is a library of database classes giving direct access to drawing entities. This class is needed to model all graphical elements such as the boom of the crane is made up of solid elements of the class type AcDb3dsolid.

AcGe: This is a library of geometry utilities which provides classes for points, vectors and matrices etc. To do any type of standard transformations on any of the drawing entities we need to use these geometry utility class libraries. For example, in case of swinging cab of the crane, it is necessary to specify three parameters, namely, center of rotation (point), axis of rotation (vector), and angle of rotation.

AcBr: This is a library of classes for inquiring into the boundary representation of solids. It is necessary to interact with AcBr class library, to get information about topological elements with in the solid model. For example, to detect interference between the crane/load and site objects we need to invoke the methods defined in these classes.

As ARX modules share the memory space with AutoCAD, it has direct access to the AutoCAD entities. This makes the execution of ARX programs faster than the alternate development methods. However this direct access to memory also exposes AutoCAD to greater risk of crashing, if the Object ARX module crashes.

4.2 System Architecture

Figure 2 shows the architecture of the system within the AutoCAD environment. Key modules developed within ObjectARX environment are shown. The features of these modules along with the basic features of AutoCAD provide the system functionality.

The Library Management Module is a key module that provides seamless access to the library of cranes, loads and sites. This module ensures that appropriate graphics files are loaded, corresponding objects constructed and initialized with relevant attribute information. The graphics files are created off-line and stored in appropriate directories as library files.

The Motion Control Module deals with various motions of the crane. This module will define the master-slave relationships between the various components of the crane and ensure that the motions along the degrees of freedom of the model selected are natural. The functions to check for interference between the crane, load and site are also specified in this module.

The Data Integration Module is used to link the state of the crane's graphical image with the load charts. As the boom is lowered or raised the lifting capacity of the crane will change. This module will keep track of the geometrical properties and read the corresponding capacity from the charts. As the most convenient way to store the crane table is through a standard database such as MS Access. Object Data Base Connectivity is used to establish the link between the lift planning system and the database. This also ensures that the load chart can be stored using any standard database.

The user interface module will permit the user to utilize the added functionality in an efficient manner. It makes use of relevant MFC as well as AutoCAD classes and presents a customized lift-planning interface in the AutoCAD environment. The various events created in the interface are linked to functions within the other modules.

5.0 System Usage

The system is currently under development, a brief description of the usage is given in this section. Figure 3 shows a typical screen from the planning system. Using the options in library management menus the user can invoke relevant dialog boxes to select and specify different crane configurations, load geometry and site.

Using the motion menus the crane is moved to a location and the lift operation involving the picking and placing of the load is attempted. In addition to the usage of above menus the arrow keys on the keyboard can be used, for example, to move the crane in all directions. During the pick and place operation the user is notified if there are any obstructions or if the load exceeds the capacity of the crane. If there are any impediments, the user can alter the crane location, lift path or pick location and make another attempt.

If after numerous tries a workable plan cannot be developed with a particular crane-load-site combination. A different crane configuration can be selected and the planning process continued. For some scenarios, it might be required to alter the properties of the load or site. This can also be done through the planning system. Thus the environment will allow the planner to conduct what-if analyses with little effort while visualizing the details of the plan. Once the user is satisfied with a lift plan he can store the scenario in a plan file. Numerous such plan files can be stored for review at a later date.

6.0 Summary and Conclusions

A heavy lift planning system is a complex software package. The features required for effective use of the package evolve. Adding the new features should not cause any major changes in the fundamental design of the system. An object-oriented approach to design promises a robust design to which new functions can be added incrementally.

A preliminary design for the problem was developed using Object-Oriented concepts. The concepts provide a mechanism for natural translation of the real-world situation into a software representation. However there are no objective tests to check the validity and robustness of the design.

The Object ARX environment provides basic development features suitable for the implementation of the system. Further work on implementation is required to make a detailed assessment of the strengths and limitations of this environment for this work.

Acknowledgements

This work was supported by the Department of Science and Technology, vide Grant No. III.5(110)/97-ET

References

1) Alexander, S. (1992) "Avoiding Trouble With Rigorous Planning: Load Lift Modeled On Microstation", MicroStation Mgr., 2(8).
2) Bennett, C. and Ditlinger, S. (1994) "Bechtel Automated Lift Planning System" Robotics for challenging environments; Proc., ASCE Spec. Conf., ASCE, New York, N.Y.
3) Booch, G. (1994). "Object Oriented Analysis And Design With Applications", Addition-Wesley Publishing Company.
4) "Computer Aided Rigging (CAR)"(1990). Brown & Root Braun, Houston, Tex.
5) Hornaday, W. C., Haas, C.T., O'Connor, J.T. and Wen, J. (1993). "Computer-aided planning for heavy lifts", J. Constr. Engrg. and Mgmt., ASCE, 119(3), 498-515.
6) Lin, K., and Haas, C.T., (1996). "Multiple Heavy Lifts Optimization" J. Constr. Engrg. And Mgmt., ASCE, 354-362.
7) Kruglinski, D. (1997). "Programming Microsoft Visual C++ Fifth Edition", Microsoft Press.
8) Wolfhope, J. S. (1991). "Design for computerized heavy lift planning system for construction", MS Thesis, Univ. of Texas, Austin, Tex.
9) Owen Ransen (1997). "AutoCAD programming in C/C++", John Wiley & Sons Ltd.
10) Varghese, K., Dharwardkar, P., Wolfhope, J. and O'Connor, J.T. (1997) "A Heavy Lift Planning System for Crane Lifts", Microcomputers in Civil Engineering, 12, 31-42.