CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Computer Models & Methods for a Disabled Access Analysis Design Environment

By

Charles S. Han
Kincho Law
John Kunz

CIFE Technical Report #123
July, 2000

STANFORD UNIVERSITY

Copyright © 2000 by

Center for Integrated Facility Engineering

If you would like to contact the authors, please write to:

c/o CIFE, Civil and Environmental Engineering
Stanford University
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020

SUMMARY
CIFE TECHNICAL REPORT #123

Title: Computer Models and Methods for a Disabled Access Analysis Design Environment

Authors: Charles S. Han, Post-doctorate Fellow, Department of Civil and Environmental Engineering, Stanford University,
Kincho H. Law, Professor, Department of Civil and Environmental Engineering, Stanford University, and
John C. Kunz, Senior Research Engineer, Center for Integrated Facilities Engineering, Stanford University

Publication Date: July 26, 2000

Funding Sources: CIFE, an Equipment Grant from Intel Corporation, and continuation funding from the National Science Foundation

1. **Abstract:**

The research develops a design-aid framework to support the disabled access analysis utilizing a design-intent or objective-based approach. This approach depends on the clear intent or objectives of the code or standard and the ability to decompose the intent and objectives into testable code-compliant or usability methods. In addition, as part of a performance-based approach, this research also utilizes motion-planning techniques to test the wheelchair usability of a facility. This new framework also provides an Internet-based design environment for users or designers to interact virtually with the design of a facility. An additional component of the framework, an interactive environment, also addresses the deficiencies of the performance-based approach related to the difficulties in accurately modeling all the relevant behaviors.

2. **Subject:**

This report describes a computer framework that assists the designer with an important aspect of disabled accessibility analysis of a facility design, specifically wheelchair access. This report investigates the following essential concepts needed to realize this framework:

- The capturing of the intent of a disabled accessibility building code in a computable format
- The description of a facility design in a computable format.
- The development of wheelchair movement “performance-based” or simulation methods.
- The delivery of this framework to the designer’s desktop via the Internet

With this framework, this research demonstrates the power of wheelchair simulation methods versus “prescriptive-based” wheelchair building code provisions as well as a proof-of-concept of the delivery of engineering services across the Internet in a modular and systematic fashion.

3. **Objectives/Benefits:**
   CIFE funded this research under two related CIFE Seed Proposals:
   - Performance-based Automated Building Code Checking
   - Internet-based Computer-aided Design.

   The design checking and approval process can be a critical activity that prolongs the construction and delays the operation of a facility. Automating this process has the potential to alleviate both the delays and the inconsistencies associated with manual code checking. This report demonstrates that a "performance-based" approach using simulation methods more-accurately captures the actions and behaviors of persons in a facility (in this case, wheelchair movement) than the "prescriptive-based" disabled access provisions that currently exist in building codes. Finally, Internet-based computer-aided design, the delivery of engineering such as the disabled access design-aid developed in this research, constitutes a major leveraging of heterogeneous and more powerful computing platforms across the Internet delivering these services to the desktop.

4. **Methodology:**
   The models and methods of the computing framework were developed by the authors. The framework was developed and tested against a bathroom facility as well as a Stanford University facility.

5. **Results:**
   The analysis of the test cases validated the developed computing framework for disabled access analysis. Based on the analysis of the Stanford University building, recommendations were made to the University to improve the accessibility of the building as well as recommendations on the design of a future building that will serve in the same capacity. Finally, based on the developed and test wheelchair simulation methods, recommendations were made to the existing disabled accessibility building code, the *Americans with Disabilities Act Accessibility Guidelines*.

6. **Research Status:**
   The developed computing framework is currently being tested between Stanford University and other educational institutions that are connected along the Internet2 backbone to determine the critical issues of this Internet-based analysis system with respect to the transfer of large facility models. The framework will be extended to integrate mobile computing platforms under CIFE Seed funding for the project entitled "Leveraging Mobile Computing Device Technology in a Distributed Engineering Services Environment." Future extensions include the incorporation of other aspects of disabled access besides wheelchair analysis as well as incorporation of other building-code-related and non-building code-related analyses. An NSF grant under the Digital Government Program has been awarded recently to develop a formal model for the accessibility codes.

Abstract

Designing a facility entails generating a configuration that satisfies a set of usability constraints. This research develops computer models and methods providing designers with disabled access analysis tools to test the usability of a facility. Among the usability constraints, building code regulations are considered the most critical. The design intent of building codes include the provision of minimum standards to safeguard life or limb, health, property and public welfare. The statements or provisions in building codes often contain "prescriptive" specifications of these objectives. However, prescriptive-based codes can be ambiguous, contradictory, complex, and unduly restrictive. In this research, performance-based methods are developed to provide designers with analysis complementary to the prescriptive-based disabled access code, the *Americans with Disabilities Act Accessibility Guidelines* (ADAAG).

The research develops a design-aid framework to support the disabled access analysis utilizing three models:

- A *design-intent model* that organizes computer methods to analyze a facility design.
- A *product model* that describes the facility design.
- A *document model* tightly integrated with the intent model that extracts the relevant provisions to inform the designer of code violations of a facility design.

The design-intent or objective-based approach used by the design-aid framework depends on the clear intent or objectives of the code or standard and the ability to decompose the intent and objectives into testable code-compliant or usability methods.

In addition, as part of the performance-based approach, this research also utilizes motion-planning techniques to test the wheelchair usability of a facility. This new framework also provides an Internet-based design environment for users or designers to interact virtually with the design of a facility. An additional component of the framework, an interactive environment, also addresses the deficiencies of the performance-based approach related to the difficulties in accurately modeling all the relevant behaviors.

Acknowledgments

This report is reproduced from a Stanford University doctoral dissertation by Charles S. Han. The doctoral committee comprised of Prof. Kincho H. Law (principal advisor), Dr. John C. Kunz (co-advisor), Prof. Jean-Claude Latombe, and Prof. Martin A. Fischer.

This research has been partially funded by the Center for Integrated Facilities Engineering at Stanford University, an Equipment Grant from Intel Corporation, and continuation funding from the National Science Foundation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Designing a facility entails generating a configuration that satisfies a set of usability constraints. Among these usability constraints, building code regulations are considered the most critical. The design intent or objective of building codes includes the provision of minimum standards to safeguard life or limb, health, property and public welfare [58]. Many of the provisions in building codes such as the *Uniform Building Code* (UBC) [58] and the *Americans with Disabilities Act Accessibility Guidelines* (ADAAG) [1] contain "prescriptive" specifications of these objectives. For example, the following ADAAG provision prescribes the acceptable width of a doorway for a wheelchair user:

4.13.5 Clear Width. Doorways shall have a minimum *clear opening of 32 in* (815 mm) with the door open 90 degrees, measured between the face of the door and the opposite stop (see Fig. 24(a), (b), (c), and (d)). Openings more than 24 in (610 mm) in depth shall comply with 4.2.1 and 4.3.3 (see Fig. 24(e)).

Advantages of using prescriptive provisions include straightforward evaluation of a design using the prescribed parameters and having the ability to make this evaluation without needing high-level engineering knowledge about the specific analysis. However, prescriptive-based codes can be ambiguous, contradictory, complex, and unduly restrictive [20,23,39]. Solutions constrained by prescriptive-based codes address only a fraction of the possible solutions that meet the design intent or objectives of these codes. Furthermore, instances exist where adhering to these prescriptive provisions produces a design that may not satisfy usability.

As a partial solution to the problems that exist in prescriptive-based building codes, many jurisdictions have adopted or are moving toward the adoption of “performance-based” codes [23,57]. As opposed to prescriptive-based codes that provide solutions abstracted from the design intent or objective of a building code, performance-based codes attempt to directly capture the behaviors that conform to the intent of the design codes or regulations. This direct performance-based approach accepts design solutions that satisfy the usability constraints, including those solutions that do not comply with the prescriptive-based constraints specified by a design code. On the other hand, when a performance-based approach accurately models usability, this approach will identify and reject design solutions that are not usable—as noted earlier, some unusable designs may be accepted by an analysis utilizing the prescriptive-based constraints specified by a design code.

Limitations of the performance-based approach include the difficulty in defining all the quantitative levels or modeling all the relevant behaviors of performance:

- If a particular behavior is inadequately modeled, the performance-based analysis may reject a design solution that satisfies the design intent or objectives of the code.
- If a particular behavior is incorrectly modeled, the performance-based analysis may accept a solution that does not satisfy the intent of the code or reject a solution that does satisfy the intent of the code.

Table 1.1: Prescription versus Performance of a Design Code (modified from [23]).

Code Type	Advantages	Disadvantages
Prescriptive Codes	Straightforward evaluation of compliance with established requirements No requirements of high level engineering expertise	Requirements specified without statement of objectives Complexity of and conflicts within the structure of existing codes Overly restrictive Sometimes compliance does not insure meeting of code intent
Performance Codes	Establishment of clear goals Facilitating use of new knowledge when available Non-complex documents Permitting prompt introduction of new technologies to the market place	Difficult to define quantitative levels of performance Difficult to capture relevant behavior Difficult to evaluate compliance with established requirements Need for validation of the tools used for quantification

Table 1.1 summarizes the advantages and disadvantages of prescriptive- and performance-based building codes, and Figure 1.1 depicts the relationship among the design solution space given a set of usability constraints and the solution spaces based on the prescriptive-based and performance-based formulations of these usability constraints. Designs that are in compliance with prescriptive-based codes may represent a small number of all possible designs, and not all of them necessarily meet design objectives. Designs based on performance-based codes, ideally, include a larger portion of desirable designs since they directly fulfill the design objectives and requirements.

This research develops computer models and methods providing designers with disabled access analysis tools to test the usability of a facility. Specifically, the developed computer environment allows designers to input the design of a facility and test the

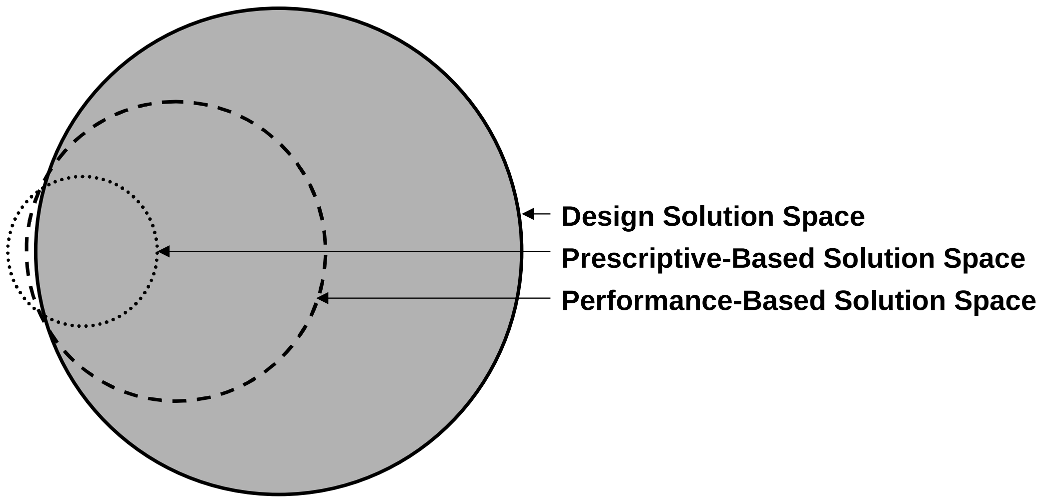

Design Solution Space
Prescriptive-Based Solution Space
Performance-Based Solution Space

Figure 1.1: The relationship among the three design solution spaces.

wheelchair usability of the design. The developed performance-based methods provide designers with analysis complementary to the prescriptive-based ADAAG. The research utilizes the performance-based approach to develop the usability analysis while minimizing the deficiencies associated with this approach and develops a design-aid framework that utilizes three models:

- A *design-intent model* that organizes computer methods to analyze a facility design.
- A *product model* that describes the facility design.
- A *document model* of the building code tightly integrated with the intent model that extracts the relevant provisions to inform the designer of code violations of a facility design.

The design-intent or objective-based approach used by the design-aid framework depends on the clear intent or objectives of the code or standard and the ability to decompose the intent and objectives into testable code-compliant or usability methods. The hierarchical structure of intents and the decomposition of these intents into sub-intents consist of analysis methods associated with prescriptive provisions, but the hierarchical organization of the intent or objectives of the code allows these prescriptive analysis methods to be substituted with performance-based methods. Furthermore, while several prescriptive provisions may be needed to describe a sub-intent of the code (often increasing the complexity and creating contractions among provisions of the code), as this research demonstrates, a single performance-based method can replace a set of prescriptive provisions thus alleviating possible complexity and contradiction. This new framework also provides an environment for users or designers to interact virtually with the design of a facility. An additional component of the framework, an interactive environment, also addresses the deficiencies of the performance-based approach related to the difficulties in accurately modeling all the relevant behaviors.

Working within this framework, this research develops performance-based methods to demonstrate wheelchair usability in a facility. Specifically, the research leverages motion-planning techniques in an attempt to directly capture wheelchair motion and behavior. The performance-based methods and the design-aid framework are demonstrated using a simple test case as well as with an analysis of a university facility.

1.2 Related Research

Design standards and product and process modeling have been active research areas for several decades. To review all relevant works is beyond the scope of this thesis. For a United States perspective, see [14]. This section presents both an overview and compares and contrasts the related research with this research effort. Section 1.2.1 presents an overview of building standards research. Section 1.2.2 presents the relevant product and process model research since the ability to automate the analysis of a designed facility is dependent on the description of the facility.

1.2.1 Building Standards Research

1.2.1.1 Decision Tables and Standards Analysis, Synthesis and Evaluation

Fenves demonstrated the use of decision tables to represent design standards provisions [13]. The structure of decision tables naturally lends itself to the organization of a set of conditions on data items that make up a provision. As shown in Table 1.2, a decision table is divided into four quadrants with each quadrant separated with a double line. The upper left quadrant contains the conditional stubs or the list of logical variables of the provision. The upper right quadrant contains the conditional entries of these variables. The lower left quadrant contains the action stubs, and the lower right quadrant contains the action entries.

Conditional entries use the following nomenclature [15]: “T” or “Y” means the condition must be true, and “F” or “N” means the condition must be false for the rule to apply. “I” or “.” means the condition is immaterial. “-” or “+” indicate that the condition is respectively implicitly false or true. An “X” in the action entry indicates the action to be taken for the given rule. Table 1.3 shows the use of the decision table nomenclature for a table concerning the maximum height of a building.

Table 1.4 illustrates a requirement for stair dimensions. For this table, if a stair does not satisfy all three conditions of stair width, riser height, and tread width (in this case, the conditions need to be false), the stair violates the requirement. Note that this table contains an “E” rule column representing the else rule (similarly, columns 5 through 10 in Table 1.3 can be replaced with an else rule column).

While decision tables can concisely represent provision requirements, an individual decision table alone does not provide a means to address the overall organization (including relationships among provisions) of a design standard. Fenves et al. addressed this deficiency with the development of the Standards Analysis, Synthesis and Evaluation (SASE) methodology [15]. SASE provided a formal methodology to represent individual provisions (with data items and decision tables), relationships among provisions (the

Table 1.2: Regions of a decision table (from [15]).

CONDITION STUB	CONDITION ENTRY		
ACTION STUB	ACTION ENTRY		

Table 1.3: Checking the maximum height (from [15]).

		1	2	3	4	5	6	7	8	9	10
1	Clearance < 6'	T	T	T	T	T	T	T	T	T	F
2	Sign Const. = closed	T	T	F	F	T	T	F	F	I	I
3	Bldg. Type = 1 or 2	T	-	T	-	T	-	T	-	F	I
4	Bldg. Type = 3	-	T	-	T	-	T	-	T	F	I
5	Height > 35'	I	F	I	I	-	T	+	+	I	I
6	Height > 50'	F	-	I	I	T	I	+	+	I	I
7	Height > 60'	-	-	I	F	I	I	+	T	I	I
8	Height > 100'	-	-	F	-	I	I	T	I	I	I
1	Height acceptable	X	X	X	X						
2	Height not acceptable					X	X	X	X	X	X

Table 1.4: A stair requirements decision table (variation of table from [15]).

		1	E
1	Stair width < 22 inches	F	
2	Riser height > 8 inches	F	
3	Tread width < 8 inches	F	
1	Requirement - satisfied	X	
2	Requirement - violated		X

information network), and the organization of the standard. Finally, SASE's information network structure connects data items in a standard through precedence relationships.

1.2.1.2 An Object-Oriented Approach

Garrett and Hakim point out two deficiencies in the SASE approach to modeling design standards which uses decision tables composed of data items as the primary means of logic representation [18]. First, with no formal model of the design objects, a standard becomes a "large, unwieldy" set of decision tables. Second, lack of evaluation methods other than decision tables leads to inefficiencies such as forcing conditional decision-making upon unconditional items such as design functions.

To address these issues, Garrett and Hakim developed an object-oriented approach organizing a design standard around design objects pertinent to the design standard. This object-oriented approach uses the following main groups of objects:

- A group of several hierarchies of objects merged together which defines design-specific attributes and hierarchies as well as attributes such as shape, function, and material that may not be influenced by the design standard in question.

- A performance-limitation hierarchy that defines the performance or behavioral constraints on the design objects.

- A data-item hierarchy that provides a structure for classifying and describing a piece of information (for example, a basic data item, a rule, or a function), and a data-item-instance network that expresses the relationships among data items.

Garrett and Hakim's incorporation of design objects within the analysis process provides a logical extension to the SASE methodology.

De Ward offers a slightly different object-oriented approach to design standard processing [9]. Garrett and Hakim tightly couple the design object structure to the particular design standard. De Waard first develops the *product model* (a description of design objects or building components and their relationships among one another) for residential buildings. Using the developed residential product model, de Waard contends that modeling design standards must consider the building model implicit in the regulations. To illustrate this point, de Waard models several provisions (referred to as *Building Decrees*) extending and adjusting the relationships developed in the residential product model.

1.2.1.3 The Hyper-Object-Logic Model

Yabuki uses an object-logic model to represent design standards for design application and standards analysis combined with a hypertext document model structure that can store relevant design standard provisions and related information [60]. Figure 1.2 shows the object-logic framework. The object-logic system consists of two sets of object-oriented hierarchies (the standards base that consists of a “member class” and a “method object” hierarchy and CAD object database that consists of an “object model” and “data objects” hierarchy). Yabuki’s research focused on the American Institute of Steel Construction (AISC) Load and Resistance Factor Design (LRFD) specification for steel construction [44]. In Yabuki’s implementation, a (steel) “member” is a subclass of a generic object. In addition, the object-logic model contains two design applications (a conformance-checking module and a component-design module), and a standards-analysis module.

The standards base includes method objects written in object-logic sentences that are associated with a class in a “member” class hierarchy (method objects represent the provisions of the design standard). The method object hierarchy has a “method” root object with a “requirements,” a “determinants,” and a “classifications” subclass, and Yabuki made the observation that a “requirements” method object class can only be associated with a leaf node in the member class hierarchy. The CAD object database uses object-logic sentences to express external constraints to a design member object.

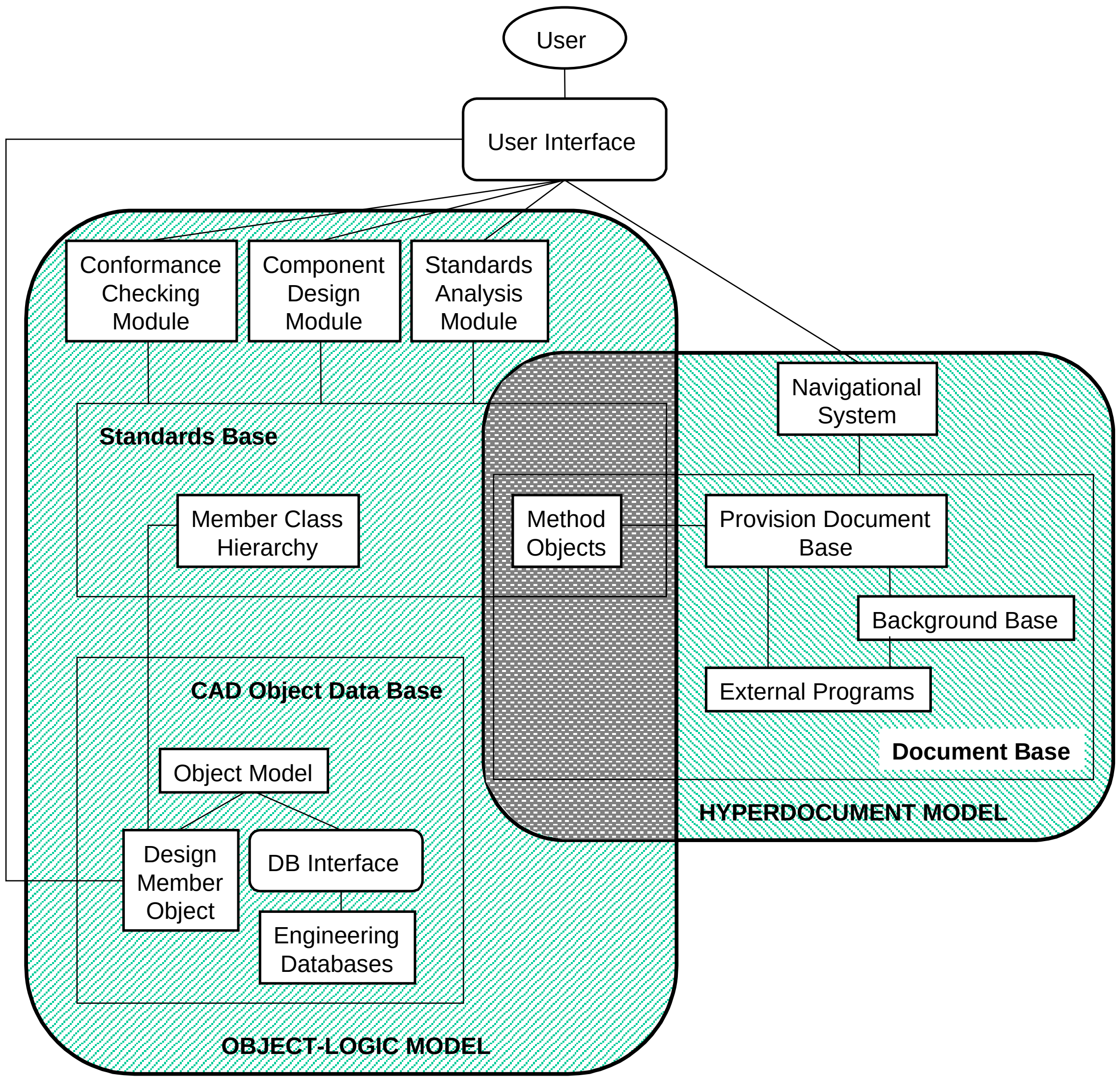


Figure 1.2: Overview of Yabuki's Hyper-Object-Logic model.

Yabuki implements two design applications and a standards-analysis module to validate the developed object-logic framework. The two design applications take a user-defined design object (a steel member) from the CAD object database and check the design for conformance in the standards base by traversing the member class hierarchy and using logic resolution and message passing among the method objects associated with each traversed class. Before checking for conformance, the component-design module generates a component based on heuristics derived from the requirement of the user's preliminary choice of a possible member, again utilizing the CAD object database for the member generation. The standards-analysis module does not depend on the CAD object database to check the completeness, uniqueness and correctness of the standard at both the provision level and the organization level—the module traverses the member class hierarchy examining the method objects associated with the traversed classes.

Finally Yabuki connects the object-logic model with a hyperlink-based document system he calls the "HyperDocument" system implemented using HyperCard on a Macintosh computer. The ubiquity of the World Wide Web has made hyperlink-based documents and the associated navigation model commonplace. However, Yabuki imposes a formal system of linking the method objects from the object-logic model to the relevant provisions in the document base. Thus, the standards base in the object-logic model associates the methods of instantiated method objects with classes in the standards base member class hierarchy, and, simultaneously, the HyperDocument system links these method object instantiations with provisions in the HyperDocument provisions document base. In addition, the provisions document base has links to a background base module and an extended program module that are both components of the HyperDocument system.

1.2.1.4 Performance- and Objective-based Codes

In 1972, the International Union of Testing and Research Laboratories for Materials and Structures (RILEM), the American Society for Testing and Materials (ASTM), and the International Council for Building Research Studies and Documentation (CIB) jointly sponsored a symposium entitled "Performance Concept in Buildings" [49]. Proceedings from this symposium represent the organization of the research around the time of the symposium to formalize the performance concept.

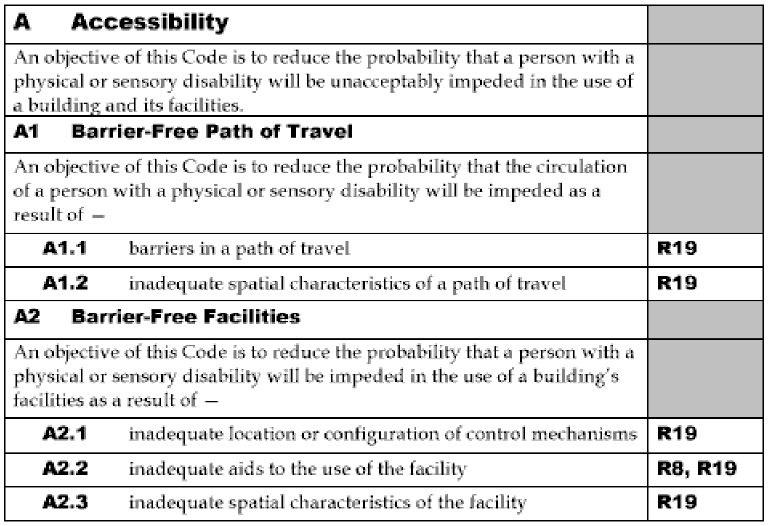
Code-related proceedings from this symposium include discussions on structural analysis [61], energy use [43], and fire-resistance [56]. Interestingly, many of the performance-based criteria seem prescriptive in nature. For example, Yokel prescribes criteria for load capacity using sums of factored loads [61], and Seigel proposes prescribing temperature limits for building materials [56]. Finally, Eberhard discusses computer-based code systems using the performance concept including proposals for simulation-based analysis [12].

More recently, there have been several research efforts related to performance-based analysis of buildings in the areas of structural, energy, and fire analysis. Krawinkler discusses the rationale and challenges to performance-based earthquake engineering (PBEE) [41]. Krawinkler holds the prescriptive codes accountable for stifling innovation because new concepts are difficult to fit into the rigid prescribed framework, and cites the example of the slow acceptance of base isolation.

Efforts in performance-based energy analysis include the Design Tools Project (DTP) at Pacific Northwest National Laboratory (PNNL). Currently, PNNL provides an energy analysis package that works within the AutoCAD environment, and the lab intends to incorporate the *Industry Foundation Class* product model into subsequent versions of the analysis package.[1] As noted earlier in this chapter, some jurisdictions have adopted or are moving toward the adoption of performance-based codes including California with respect to the *Title 24 Energy Efficiency Standards* [57]. With Title 24, users have the option of using either the well-established prescriptive-based calculation methods or a performance-based analysis method. Other research efforts include a web-based client/server energy calculation program [16].

Performance- and simulation-based analysis of fire is a widely researched area. Hadjisophocleous discusses building-code-related fire safety criteria based on deterministic and probabalistic approaches [23]. Recent simulation research includes fire and smoke simulation [7,47]. In addition, the Building and Fire Research Laboratory (BRFL) of the National Institute of Standards and Technology (NIST) is developing an

[1] See online documentation at http://www.energytech.pnl.gov:2080/dtp/dtp.html for a detailed description of PNNL's Design Tools Project.



A Accessibility	
An objective of this Code is to reduce the probability that a person with a physical or sensory disability will be unacceptably impeded in the use of a building and its facilities.	
A1 Barrier-Free Path of Travel	
An objective of this Code is to reduce the probability that the circulation of a person with a physical or sensory disability will be impeded as a result of –	
A1.1 barriers in a path of travel	**R19**
A1.2 inadequate spatial characteristics of a path of travel	**R19**
A2 Barrier-Free Facilities	
An objective of this Code is to reduce the probability that a person with a physical or sensory disability will be impeded in the use of a building's facilities as a result of –	
A2.1 inadequate location or configuration of control mechanisms	**R19**
A2.2 inadequate aids to the use of the facility	**R8, R19**
A2.3 inadequate spatial characteristics of the facility	**R19**

Figure 1.3: A proposal for the CCBFC objective-based disabled access code, Part A.

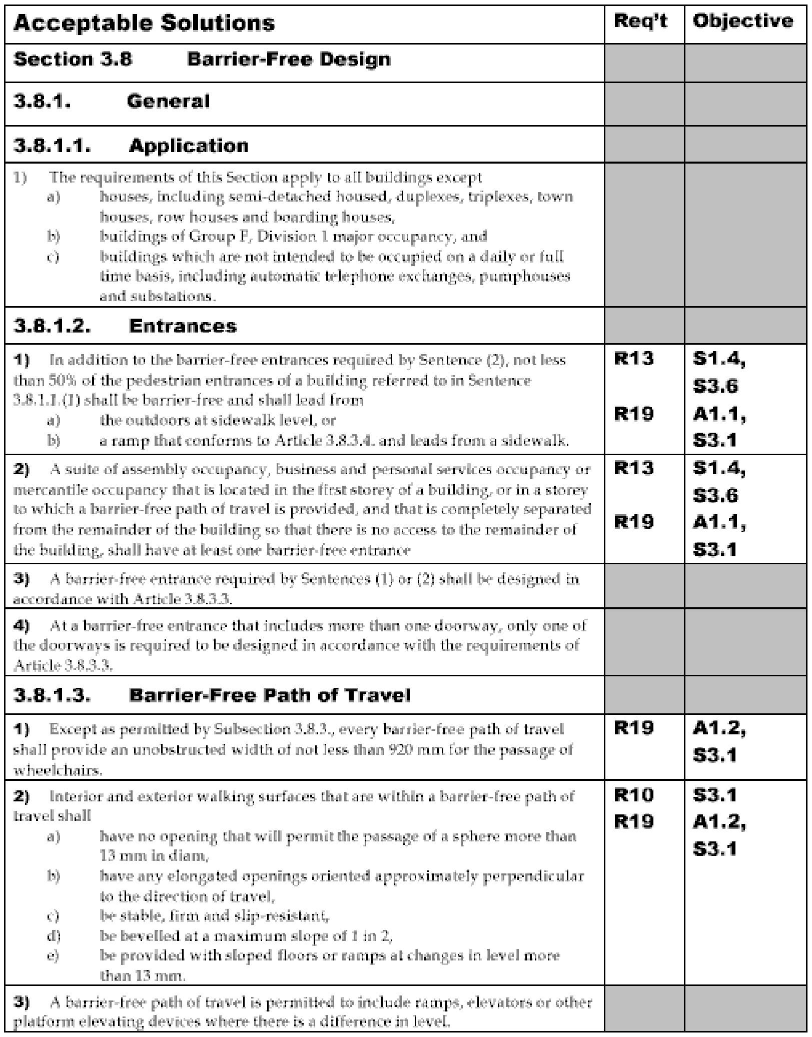
Industrial Fire Simulation System (IFSS) that models fire and fuel-burning and suppression characteristics and develops a data-exchange semantics.[2]

Several countries have begun to develop performance-based building codes. As the realization of a 1995 Strategic Plan, the Canadian Commission on Building and Fire Codes (CCBFC) called for the migration of its prescriptive-based building codes to an objective-based format, and toward this goal, the CCBFC formed the Task Group on Planning For Objective-Based Codes.[3] The Task Group's current vision of object-based codes includes a two-part document:

- Part A (shown in Figure 1.3) will describe the objectives that the code addresses and the qualitative functional requirements for solutions.

[2] See online documentation at http://www.bfrl.nist.gov/860/ps98/ifss.htm for a detailed description of IFSS.

[3] The Task Group on Planning For Objective-Based Codes publishes its documents online at http://www.nrc.ca/ccbfc/tgs/obc/index_E.shtml, and these online documents have contributed to the discussion on the CCBFC objective-based code development.



Acceptable Solutions	**Req't**	**Objective**
Section 3.8 Barrier-Free Design		
3.8.1. General		
3.8.1.1. Application		
1) The requirements of this Section apply to all buildings except a) houses, including semi-detached housed, duplexes, triplexes, town houses, row houses and boarding houses, b) buildings of Group F, Division 1 major occupancy, and c) buildings which are not intended to be occupied on a daily or full time basis, including automatic telephone exchanges, pumphouses and substations.		
3.8.1.2. Entrances		
1) In addition to the barrier-free entrances required by Sentence (2), not less than 50% of the pedestrian entrances of a building referred to in Sentence 3.8.1.1.(1) shall be barrier-free and shall lead from a) the outdoors at sidewalk level, or b) a ramp that conforms to Article 3.8.3.4. and leads from a sidewalk.	**R13** **R19**	**S1.4,** **S3.6** **A1.1,** **S3.1**
2) A suite of assembly occupancy, business and personal services occupancy or mercantile occupancy that is located in the first storey of a building, or in a storey to which a barrier-free path of travel is provided, and that is completely separated from the remainder of the building so that there is no access to the remainder of the building, shall have at least one barrier-free entrance	**R13** **R19**	**S1.4,** **S3.6** **A1.1,** **S3.1**
3) A barrier-free entrance required by Sentences (1) or (2) shall be designed in accordance with Article 3.8.3.3.		
4) At a barrier-free entrance that includes more than one doorway, only one of the doorways is required to be designed in accordance with the requirements of Article 3.8.3.3.		
3.8.1.3. Barrier-Free Path of Travel		
1) Except as permitted by Subsection 3.8.3., every barrier-free path of travel shall provide an unobstructed width of not less than 920 mm for the passage of wheelchairs.	**R19**	**A1.2,** **S3.1**
2) Interior and exterior walking surfaces that are within a barrier-free path of travel shall a) have no opening that will permit the passage of a sphere more than 13 mm in diam, b) have any elongated openings oriented approximately perpendicular to the direction of travel, c) be stable, firm and slip-resistant, d) be bevelled at a maximum slope of 1 in 2, e) be provided with sloped floors or ramps at changes in level more than 13 mm.	**R10** **R19**	**S3.1** **A1.2,** **S3.1**
3) A barrier-free path of travel is permitted to include ramps, elevators or other platform elevating devices where there is a difference in level.		

Figure 1.4: Portions of the CCBFC disabled-access code, Part B.

- Part B (shown in Figure 1.4) will describe the quantitative performance criteria with which solutions must comply.

Note, however, that in Part B, the quantitative performance criteria seem to be more prescriptive in nature as these quantities are taken from the current version of the code.

1.2.2 Product and Process Model Research

There have been several research efforts to develop object-oriented CAD systems and object-oriented building models that contain the necessary geometric, functional, and behavioral relationships of building components [9,28,36]. Eastman provides a comprehensive look at the context, history, and current efforts in product model research as it relates to building design and construction [11]. Eastman writes that the purpose and challenge of developing product models is:

> To develop an electronic representation of a building, in a form capable of supporting all major activities throughout the building lifecycle.

The goal of constructing product models is to exchange building information, and Eastman appropriately describes the emergence of several CAD systems along with early data exchange standards. Specifically, he describes DXF, the exchange of Autodesk's AutoCAD DWG file information, Autodesk's proprietary CAD file format [4], and IGES, a CAD/CAM data exchange effort initiated by Robert Fulton, General Electric, and Boeing, and transferred to the National Institute for Standards and Technology (NIST) [30].

Eastman reviews both past and current product modeling efforts [11]. He enumerates many early efforts and describes three object-based (a building as a set of related components) systems in detail:

- A system developed by SSHA focusing on housing unit design and housing estate site planning [5].

- OXSYS CAD, a system to support hospital design using the Oxford Method for a post-and-beam and slab system [27].

- GLIDE-II, a portable engineering database language and successor to GLIDE (Graphical Language for Interactive Design), an interpretive language and permanent storage system that allowed defining and storing building schemas with complex geometry [11].

1.2.2.1 Standard Exchange of Product Model Data (STEP) and Related Efforts

Many of the current product modeling projects use the concepts from the International Standards Organization (ISO) Standard for the Exchange of Product Model Data (STEP) effort [50]. Eastman dedicates several chapters to STEP and related technologies [11]. The objectives of STEP included the incorporation of object-oriented programming concepts and formal specifications of the defined structures, separation of the data model and the physical file format, supporting subsets of the total model, and the sharing of reference models among these subsets.

The STEP Committee defines the Application Protocols (APs) (the subsets) first, and the APs are later incorporated into the total model. An AP has two parts:

- The Application Reference Model (ARM) that represents the requirements of an application using the IDEF1X [35], NIAM (Nijssen's Information Analysis Method) [48], and EXPRESS-G models [54]

- The Application Interpreted Model (AIM) that uses EXPRESS [52] to specify the structure of the ARM data (the STEP Committee specifically commissioned the development of EXPRESS for this purpose)

Building-industry related APs include Part 225 (building elements), Part 228 (HVAC), and Part 230 (steelwork). Interpreted (shared) Resources include Part 41, the application context, and Part 42, geometric representation.

Eastman divides current building product modeling efforts into two categories [11]: aspect models that address a specific domain in the building industry, and framework models that address the whole structure of a building. He describes in detail three building aspect models that make use of several STEP technologies:

- The European CIMsteel effort for structural steel [8]
- COMBINE (Computer Models for the Building Industry in Europe), the European Union (EU) energy modeling effort [2,3]
- Part 225, the STEP AP that describes the building elements using explicit shape representation [53]

Eastman also describes the following building framework models:

- The Finish RATAS national building model project [6]
- Part 106, the STEP Building Core Construction Model (BCCM) [51].

BCCM utilizes the General Architecture, Engineering, and Construction (AEC) Reference Model (GARM) [21]. The GARM methodology views the product model as functional units associated with a functional requirement, and a matching set of one or more technical solutions. Furthermore, a technical solution can be decomposed into a set of lower order functional units, and the decomposition can be repeated for a technical solution associated with a functional unit as necessary. DeWard also uses the GARM in his residential building code-checking research [9].

1.2.2.2 Industry Foundation Classes

Currently, there is an effort by the International Alliance of Interoperability (IAI), a consortium of CAD vendors and other Architecture-Engineering-Construction/Facilities Management (AEC/FM) industry partners, to develop standards for a three-dimensional project model that enables interoperability between applications by different software vendors. The IAI's effort includes defining a set of objects called *Industry Foundation Classes* (IFC) that conform to current object-oriented philosophy [31][32][33].

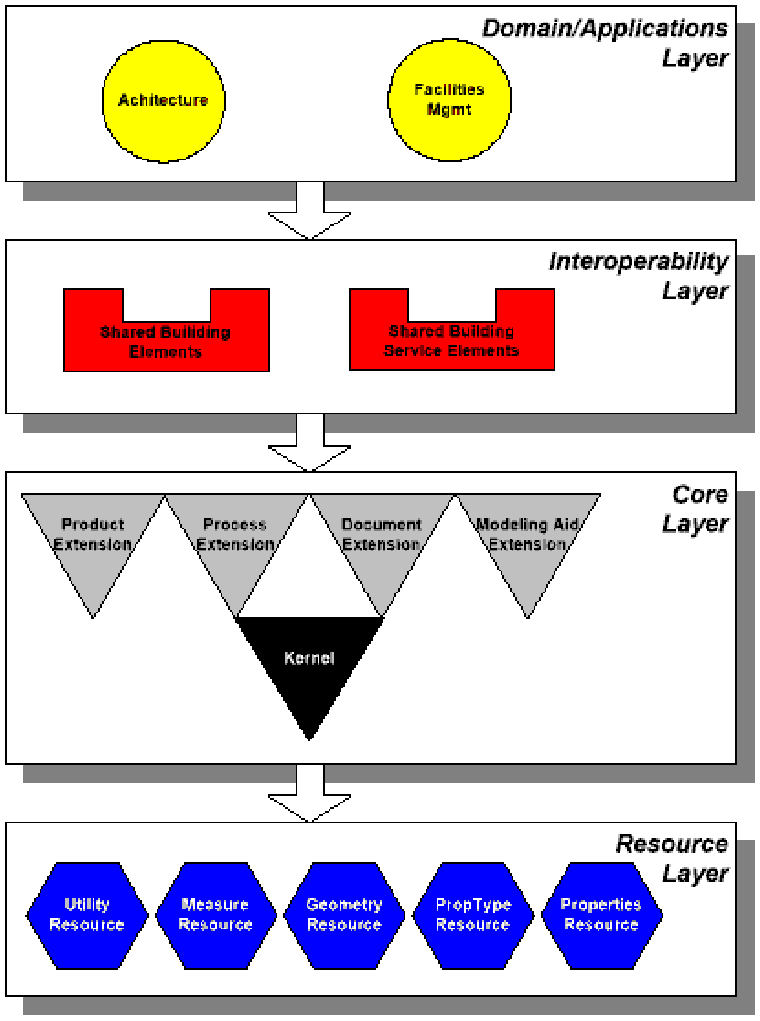


Figure 1.5: IFC Release 1.5 model architecture (from [32]).

The IFC model adopts terminology from the British Standard Institute (BSI) [22] and external sources from STEP. Figure 1.5 illustrates the decomposition of the IFC model architecture. The IFC model decomposes into four layers or modules. The lowest layer, the Resource Layer, defines resources such as units of measure, geometric representation, and other fundamental types. The next layer, the Core Layer, defines the Kernel and Core Extensions. The Kernel contains objects that are not AEC/FM-specific such as the `IfcProduct`, `IfcProcess`, `IfcModelingAid`, and `IfcDocument` objects. The

Core Extensions include AEC/FM-specific extensions to the Kernel objects. The Interoperability Layer contains Shared Building Elements and Shared Building Service Elements, and the final layer or module provides further detail in specific domains such as Architecture and Facilities Management.

Currently, the IFC model is expressed in the EXPRESS language format [54]. Efforts have been initiated to establish the IFC model in terms of *aecXML*, the Extensible Markup Language (XML) "schema for project and business-to-business communication for architecture, engineering, construction, and facility management (AEC+FM) transactions."[4]

This research develops a product model using the concepts and semantics of the IFC model. However, the product model utilizes a much simpler object hierarchy than the IFC hierarchy.

1.2.2.3 The Primitive-composite Approach

Howard et al. develop a primitive-composite (P-C) approach to address some of the object-oriented data modeling problems such as overuse of aggregation hierarchies and non-homogeneous characterization hierarchies (referring to the complex single-inheritance subclassing deficiencies) [28]. The P-C approach attempts to address these problems by developing rules to restrict the creation of complex objects. Rules include how to create primitive classes with strict subclassing restrictions, the preference for the reification of relationships over subclassing, and not allowing new attributes in the construction of composite classes from primitive classes. Primitive classes depend on descriptions of *form* (the description of an object's physical characteristics), *function* (the role or purpose of the object), and *behavior* (the way the object responds to environmental stimuli) which has also been proposed by Luth to organize engineering knowledge and concepts in structural design [38].

[4] See online documentation at http://www.aecxml.org/iaiadopt.htm for the aecxml.org press release.

This research uses an object similar in concept to the primitive object (the `GenericComponent`) as the main component in the product model and makes use of relationships to define views of `GenericComponent` instances. `FORM`, `FUNCTION`, and `BEHAVIOR` are attributes of the `GenericComponent` object as opposed to being primitive constructs.

1.2.2.4 A Design Rationale Model to Capture the Design Process

Garcia explored a design rationale model that is similar in philosophy to the design intent model developed in this research [17]. Her work investigated the use of "Active Design Documents" (ADD) to assist in the documentation of Heating, Ventilation, and Air Conditioning Systems (HVAC) preliminary design. The ADD assists both the designer and the design document user by focusing on design rationale. As opposed to static design documents, the ADD shows the designer or user the design rationale factors considered in arriving at a design decision. The model also allows the designer to either modify existing or add new parameters that affect the design decision-making process.

Garcia used field studies to establish the baseline HVAC design rationale parameters and dependencies. She notes that typical HVAC design is comprised of about 150 interdependent parameters. The ADD is a frame-based knowledge-engineering application in which the nodes (decision, alternatives, evaluation, criteria, constraints, topics, fixes, impacts previous cases, goal, design context, and design agents) are represented as frames and the relationships (generates, constrains, evaluates, selects) are represented as procedures.

1.3 The Design-intent Model for Disabled Access

As noted earlier, performance-based design code is emerging as an alternative to the traditional prescriptive-based codes. To properly define and measure the performance, the "intent" of the design code should be explicitly modeled.

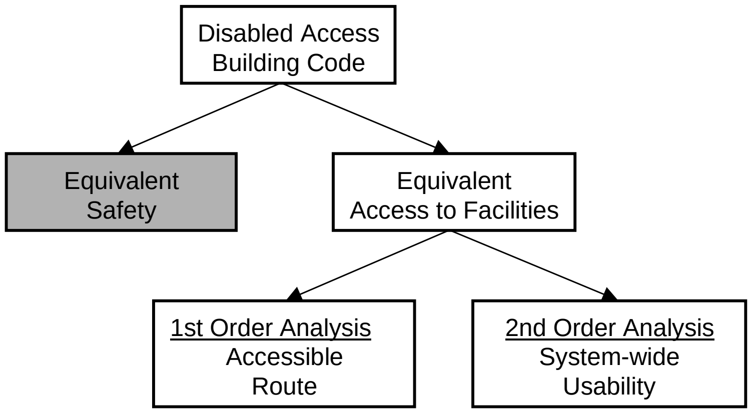

Disabled Access Building Code
Equivalent Safety
Equivalent Access to Facilities
1st Order Analysis Accessible Route
2nd Order Analysis System-wide Usability

Figure 1.6: The design-intent disabled access code model used in this research.

One of the key objectives of this research is to develop a design-intent code model for disabled access using the *Americans with Disabilities Act Accessibility Guidelines* (ADAAG) [1]. The following statement from this document states the main purpose of the ADAAG:

> **1. PURPOSE.** This document sets guidelines for accessibility to places of public accommodation and commercial facilities by individuals with disabilities…

The intent of this document can be decomposed into two sub-intents: equivalent safety and equivalent access to facilities for disabled persons. This research focuses on the issues related to equivalent access to facilities. The equivalent access to facilities can be further decomposed into two parts: the existence of an accessible route to the building components within the facility and the usability of these building components by disabled persons.

Figure 1.6 depicts a high-level view of the design-intent approach modeling the disabled access code. The accessible route analysis represents a “first-order” analysis of the disabled access code’s equivalent access to facilities. Figure 1.7 illustrates an example of an accessible route in a bathroom facility between the entrance and a water closet. To determine if the whole system of a design is usable, the whole system of accessible routes

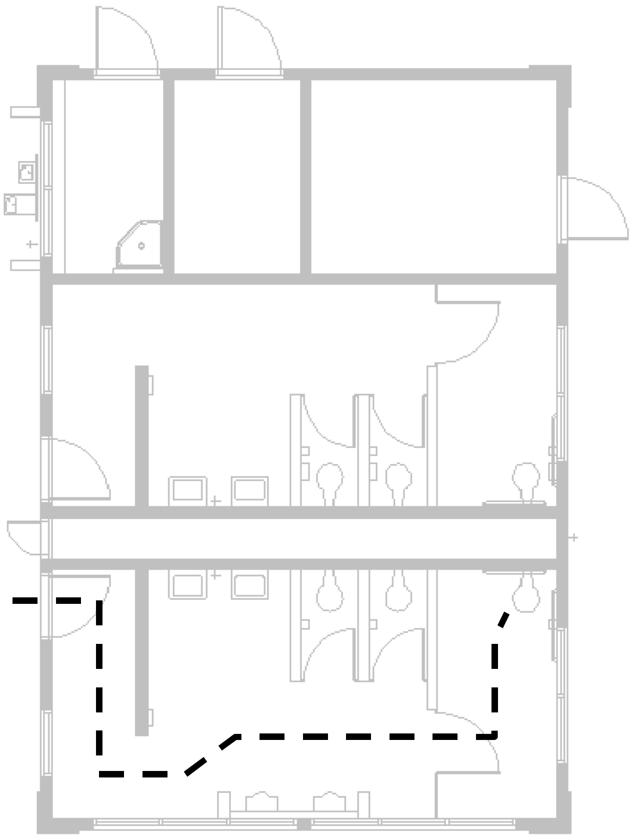


Figure 1.7: An example of an accessible route.

must be examined. An individual route or lack of an accessible route to an individual building component must be considered in the complete context of the designed facility. For example, in the bathroom facility shown in Figure 1.7, the ADAAG accepts the lack of accessible routes to the toilets in the smaller stalls as long as there is one toilet with an accessible route. This system-wide analysis represents a "second-order" analysis of usability.

This research develops two control models (the design-intent model of a disabled access building code and the product model) for the analysis of the equivalent access to facilities. The accessible route analysis module uses a combination of motion-planning

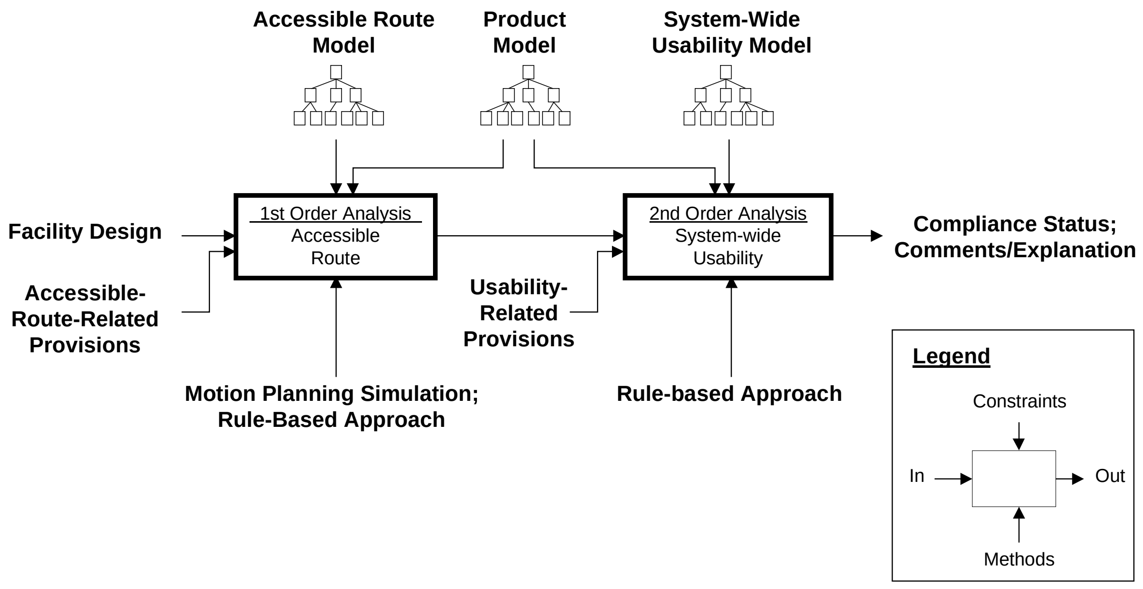

Accessible Route Model
Product Model
System-Wide Usability Model
Facility Design
Accessible-Route-Related Provisions
1st Order Analysis
Accessible Route
2nd Order Analysis
System-wide Usability
Usability-Related Provisions
Compliance Status; Comments/Explanation
Motion Planning Simulation; Rule-Based Approach
Rule-based Approach
Legend
Constraints
In
Out
Methods

Figure 1.8: The design-intent disabled access code analysis process.

simulations and "rules" to determine the existence of an accessible route in the facility. The system-wide usability analysis also uses rules operating on the information garnered from the accessible route analysis module.

Both the accessible route analysis and the system-wide usability analysis modules must understand the semantics or the description of the facility design in question, and the developed product model defines this description. From an architectural point of view, the functions of a facility can be decomposed hierarchically into buildings, stories, and spaces, and the product model captures this architectural decomposition. Both modules include the facility design (an instance of the product model) and the disabled access code as input, and both modules use the product model to store the analyses results. Figure 1.8 illustrates the interaction of the two analysis modules and the process for the disabled access code analysis.

Using motion-planning techniques, this research develops two accessible route models: the prescriptive-based model based on the ADAAG accessible route provisions and the performance-based model that attempts to directly capture wheelchair motion behavior. The research provides specific examples illustrating the deficiencies in the prescriptive-based formulation of the wheelchair usability constraints. Examples include designs that are code-compliant but not usable and designs that are non-compliant but usable.

1.4 Organization of the Thesis

The objective of this research is to develop a computational framework for the design-intent approach of ADAAG compliance checking utilizing both performance- and prescriptive-based approaches. This thesis first develops the design-aid framework for disabled access design assistance. Next, the thesis describes the implementation of prescriptive- and performance-based approaches of accessible route analysis, how these analyses are incorporated into the design-aid framework, and provides a contrast between the two approaches. The thesis completes the design-aid framework by describing the user-interactive environment.

This thesis is organized as follows:

- Chapter 2 presents an overview of the design-aid framework developed in this thesis. The overview describes the design-intent model, the product model, and the document model and introduces the disabled access analysis elements that will be incorporated into the framework.

- Chapter 3 first steps through the manual disabled access of a bathroom facility. Detailed examination of the manual process provides insight into the disabled access code intent, the prescriptive-based disabled access checking process, and the development of the automated analysis methods. The chapter then develops the product model and formalizes the hierarchical structure of the design-intent model utilized by the automated disabled access analysis.

- A major goal of this thesis is to validate a performance-based analysis as a complementary component to the prescriptive-based code-compliance. Toward this goal, Chapter 4 first develops a prescriptive-based accessible route analysis using motion-planning techniques. These analysis methods test for a code-compliant accessible route within a facility. Next, the chapter develops a performance-based accessible route analysis. The motion-planning simulations developed for the performance-based approach directly model wheelchair behavior. The chapter compares the results of the prescriptive-based and performance-based computer analysis methods to illustrate the advantages of the performance-based approach. In addition, the chapter discusses the power and the flexibility of the performance-based computer methods and how they can be applied to situations above and beyond the wheelchair behavior assumed by the code. Finally, the chapter describes the developed interactive and visualization techniques that give the designer additional disabled access analysis design tools and provide more qualitative information than the performance-based methods.

- While the main goal of this thesis is to develop the methods and framework for disabled access usability analysis, the thesis also investigates the actual delivery of the framework and analysis to the designer. Chapter 5 describes a modular integrated Internet-based design-aid framework. The chapter develops and describes the implementation of the interactive environment that allows a designer to transfer the design data from a commercial CAD package, run and visualize the analyses, and manipulate a wheelchair through the facility design.

- Chapter 6 describes the performance-based analysis of a facility on the Stanford University campus to validate the developed framework and methods. The facility violates the usability tests, and is subsequently modified to pass the performance-based analysis. The test case represents a retrospective study and, in addition, since recommendations to the University emerged from this study that will be taken into consideration, the test case also represents an intervention study.

- Chapter 7, the final chapter, contains a summary and discussion of the material presented in this thesis. The chapter summarizes the research contributions and includes specific provision recommendations to the ADAAG. In addition, the chapter discusses possible future extensions as well as some of the limitations of the presented research.

Chapter 2

The Design-aid Framework

One of the research goals is to provide the designer with a set of disabled access analysis tools to test the usability of a designed facility. Toward this goal, the research develops a modular framework that provides the interface between the designer and three interacting components that provide the analysis: the description of the designed facility, the organization of the analysis tools, and the generated analysis report.

This chapter develops a design-aid framework to provide the designer with a flexible array of design tools including code-checking and usability analysis. In this chapter, a brief overview of the design-aid framework is provided. In addition, the chapter introduces the applications that provide the interface between the user and the framework. The design-aid framework consists of three main component models: a product model, a design-intent model, and a document model. Figure 2.1 depicts the overall design-aid framework and the three components.

The objective of this chapter is to provide an overview of the framework, its component modules, and their interactions. Details of each component module are described in subsequent chapters. The three components form the foundation to develop the different types of design-aid analysis, and the chapter describes these components as follows:

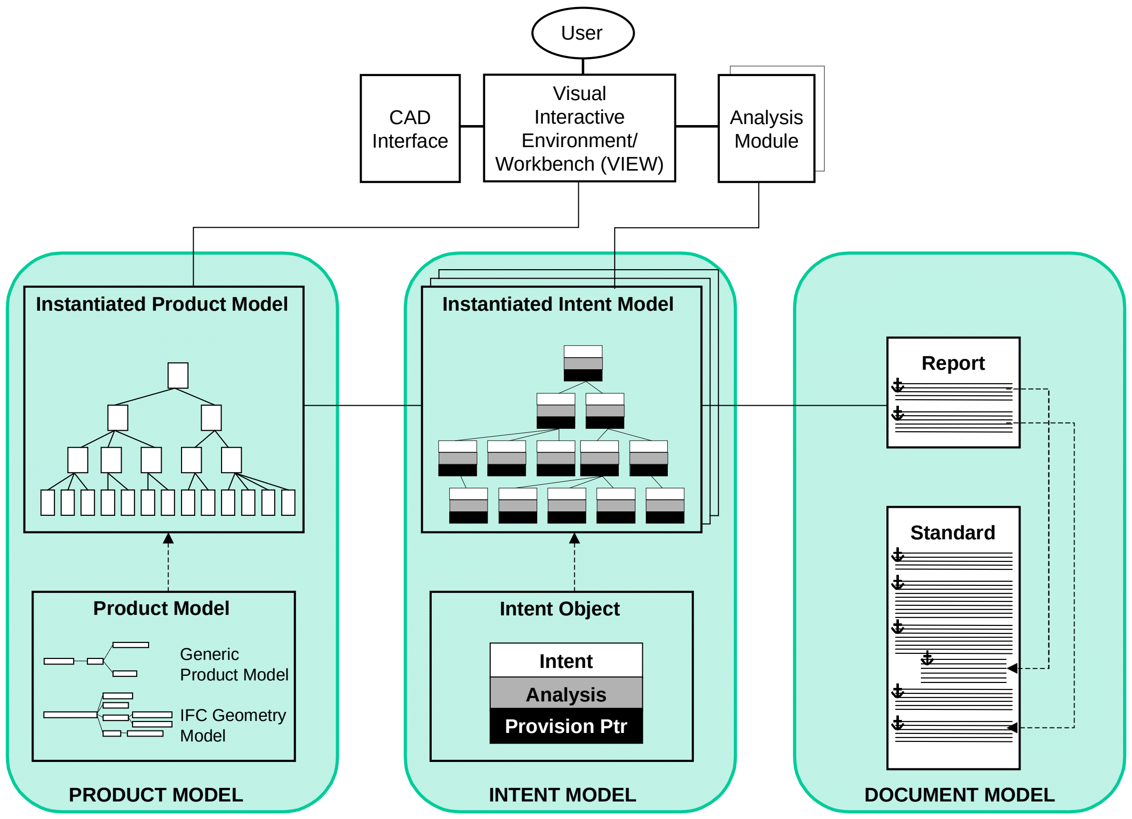


Figure 2.1: The design-aid framework.

- Section 2.1 introduces the product model. The product model describes the facility design, and the design-intent model analysis methods extract, modify, and insert information into the product model.

- Section 2.2 introduces the design-intent model, an object-oriented model that captures the intent of a building code that organizes the analysis structure used to execute the prescriptive- and performance-based analyses approaches described later in the thesis. A more detailed description of the design-intent model will be presented in Chapter 3.

- Section 2.3 describes the document model that contains the standard or code document relevant to the target analysis of the instantiated design-intent model. The document model generates the analysis report.

- Section 2.4 gives an overview of the prototype implementation. The description of the prototype gives the reader an understanding of the actual implementation and delivery of the analysis to the designer using current technology. Detailed implementation of the system is given in Chapter 5.

2.1 The Product Model

This research develops a product model that is flexible enough to describe the design of a facility and that also acts as a repository for the data derived from the analysis methods of the design-intent model. Toward this goal, the product model reifies three main objects:

- A `Table` object that associates a "key" object (such as a string) in a table with a "value" object in the same table.

- A `GenericComponent` object, a subclass of the `Table` object.

- A `Relationship` object, also a subclass of the `Table` object.

The `Table` object is a subclass of a root object, `ExpressEntity`, and Figure 2.1 shows this developed product model hierarchy in the "Product Model Hierarchy" box (the developed product model hierarchy references the generic IFC geometry hierarchy shown in this box to describe the form of a building component). The `GenericComponent` object is flexible enough to describe either a building component or an analysis concept. A doorway is an example of a building component, and "maneuvering clearance" is an example of an analysis concept (a wheelchair user maneuvers through this clearance to gain access to a building component). The `Relationship` object explicitly creates views or relationships among

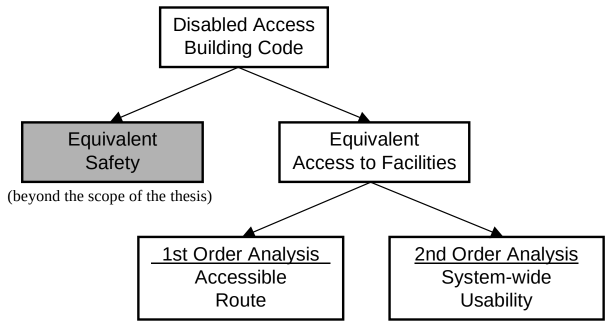

Disabled Access Building Code
Equivalent Safety
(beyond the scope of the thesis)
Equivalent Access to Facilities
1st Order Analysis
Accessible Route
2nd Order Analysis
System-wide Usability

Figure 2.2: The design-intent disabled access code model.

`GenericComponent` instances of both the building components of the facility and the concepts generated by the analysis methods.

This research develops the `GenericComponent` and the `Relationship` object extending the capabilities of the `IfcBuildingElement` object and the `IfcRelationship` object in the IFC product model. These main objects along with a set of supporting objects (for example, the descriptions of geometry taken from the IFC model) sufficiently describe both the design of the facility and the analysis information. While the product model follows the object-oriented paradigm, objects in the product model do not have associated methods. All methods are exclusively associated with the design-intent model.

2.2 The Design-intent Model

The design-intent model organizes the intent of a standard or code to enable automated usability or code-compliance analysis of a facility. The intent of a standard or code can be refined and decomposed into sub-intents, and the design-intent model uses this hierarchical structure with `Intent` objects, the reification of the intent, in this structure. Figure 2.2 illustrates the hierarchical structure of the disabled access code organized with the design-intent model. The intent of the disabled access code decomposes into the intents of equivalent safety and equivalent access to facilities. The equivalent access to facilities sub-intent further decomposes into the intents of providing accessible routes and system-wide usability throughout the facility. This research focuses on the intent of the disabled access code regarding equivalent access to facilities.

The key object of the implementation of the design-intent model is the `Intent` object. The attributes in the `Intent` object include:

- A state variable (`state`) that keeps the state of the `Intent` instance. The possible values are `FULFILLED`, `UNFULFILLED`, and `UNKNOWN`.
- A set of child sub-intents to the `Intent` instance that represents a hierarchical design-intent structure.
- A set of pointers to relevant provisions in the implemented standard or code document.

The `Intent` instance determines its state by running the `Analyze`() method associated with the instance. This method examines the facility design described by the product model, and, if necessary, inserts analysis-related information into the product model that may be critical for its own, a parent's, or a sibling's Intent instance `Analyze()` process. In addition, the `Analyze()` method traverses a hierarchical structure by recursively executing the `Analyze()` methods of the child sub-intents. After traversing the child sub-intents, the child-intents' `Analyze()` methods may have inserted necessary analysis information for the parent `Analyze()` to process the parent `Intent` instance.

Even after recursively traversing the child sub-intents, the `Analyze()` method may not have sufficient information stored in the product model to make either the `FULFILLED` or `UNFULFILLED` determination of the `Intent` instance's state, and the `Intent` instance remains in the `UNKNOWN` state. Lack of information occurs when the analysis of a specific building component needs to be made in the broader context of the facility system. For instance, the state of this particular `Intent` instance will be determined higher up in the hierarchical `Intent` structure. The results (`FULLFILLED`, `UNFULFILLED`, or `UNKNOWN`) are directly stored in the product model.

2.2.1 Accessible Route Analysis

Figure 2.2 shows two sub-intents of the equivalent access to facilities intent. This section introduces the first sub-intent, the accessible route. In developing the proposed design-aid framework, this research formalizes the first-order analysis required for equivalent access to facilities analysis, the *accessible route*. The ADAAG defines the accessible route as:

3.5 Definitions. Accessible Route. A continuous unobstructed path connecting all accessible elements and spaces of a building or facility. Interior accessible routes may include corridors, floors, ramps, elevators, lifts, and clear floor space at fixtures. Exterior accessible routes may include parking access aisles, curb ramps, crosswalks at vehicular ways, walks, ramps, and lifts.

The existence of an accessible route ensures the usability of a facility for wheelchair-bound users, and in most cases, the wheelchair user should be able to negotiate the accessible route using only forward motion.

Determining the existence of accessible routes in a facility design requires the design-intent code model to further decompose the accessible route analysis. The components generated from this decomposition are mapped to the prescribed provisions that the accessible route analysis module uses to analyze the design. The accessible route analysis uses geometric interference rules to determine the accessibility of some of the accessible route components and motion-planning simulations to generate code-compliant paths.

Table 2.1: Accessible route analysis models and methods.

Approach	Models	Methods
Prescriptive-based	Code-compliant Accessible Route Model Supporting Product Model	Code-compliant Motion Planning Simulations Code-compliant Rule-based Approach
Performance-based	Usable Accessible Route Model Supporting Product Model	Usability Motion Planning Simulations Usability Rule-based Approach

For the performance-based approach, this research utilizes the same accessible route model (and the supporting product model). However, as opposed to using motion-planning techniques to show compliance with the prescribed provisions, the performance-based motion-planning techniques directly model wheelchair behavior to generate the paths.

Table 2.1 summarizes the models and methods for the development of the design-aid framework using the prescriptive-based or performance-based formulation of the usability constraints along with the interactive environment.

2.2.1.1 Analyzing an Accessible Route for Prescriptive-based Code-Compliance

One advantage of prescriptive-based codes is the straightforward evaluation of compliance with established requirements explicitly stated in the code provisions. Indeed, automated evaluation of accessible routes is straightforward if the designer is required to indicate the routes through a facility. If the designer explicitly provides the accessible routes to the automated analysis, the routes could be evaluated by a computer application with a set of geometric interference tests. However, analysis based on a manually-defined accessible route has two deficiencies:

1. In practice, designers are not required to delineate the accessible routes in a facility since delineating the connections of all accessible building components to accessible routes is not practical. Rather, a building official will typically inspect a plan and determine the existence of accessible routes to all the accessible building components.

2. A design with mislabeled or missing accessible routes does not necessarily imply there is no accessible route since the designer may have failed to label or recognize a code-compliant or usable route.

Computer-based motion-planning techniques naturally translate to searching for an accessible path between building components representing the initial and goal points and the obstacles in the space. The geometric requirements listed in the prescriptive provisions map to the robot's geometric and motion parameters. Since the geometric requirements prescribed in the provisions are abstractions of the design intent of the accessible route, the robot captures these abstract behaviors as opposed to directly capturing wheelchair geometric and motion behavior.

2.2.1.2 Analyzing an Accessible Route for Performance-based Usability

As noted, the design-intent modeling approach allows the substitution of prescriptive methods with performance-based methods. This research determines the accessible route using a performance-based method. To analyze a facility for the existence of the required accessible routes, the research directly models wheelchair behavior using motion-planning techniques. As opposed to the prescriptive-based analysis in which the prescribed geometric requirements are modeled, the performance-based analysis models the actual wheelchair geometry and wheelchair movement parameters.

The performance-based motion planning simulation uses the wheelchair geometry prescribed by the disabled access code and restricts the motion parameters of the wheelchair to reflect the compliance/non-compliance threshold for configurations prescribed by the disabled access code. This simulation reveals both the deficiency in a prescriptive-based approach and the advantage of a performance-based approach of wheelchair use:

- A prescriptive-based approach represents an approximation and summary of behavior for a model wheelchair.
- The performance-based approach can vary the wheelchair parameters to analyze the performance of specific wheelchair models.

2.2.2 System-wide Usability Analysis

The accessible route analysis passes the generated route information to the system-wide usability analysis, the second sub-intent of the equivalent access to facilities intent shown in Figure 2.2. This module determines the overall accessibility for a collection of building components based on the accessible route information.

Note that not all of the building components within the collection need to have accessible routes in order for the collection to be accessible. For example, for the bathroom facility illustrated in Figure 2.3, only one of the water closets has a valid accessible route (as shown by the dashed line), but the bathroom still complies with disabled access requirements. From the ADAAG:

4.23.4 Water Closets. If toilet stalls are provided, then *at least one* shall be a standard toilet stall complying with 4.17…

The system-wide usability analysis module must examine components in the entire system to determine whether the system is usable even if one or more of the individual components within the system are not usable. In this example, two of the water closets do not have valid accessible routes (there are also non-accessible-route-related issues that make these water closets unusable for disabled persons), yet in compliance with the intent of the code, a disabled person can still effectively use the facility. Thus, the system-wide

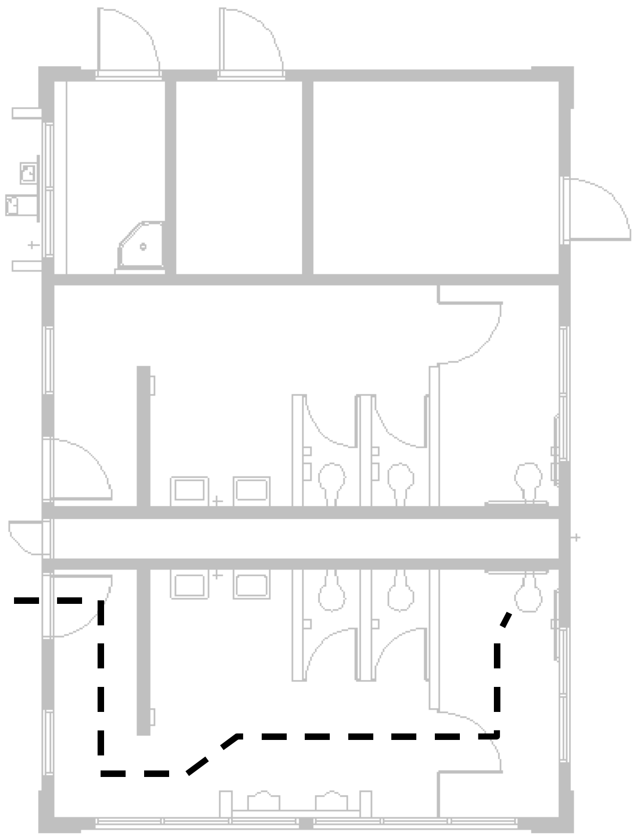


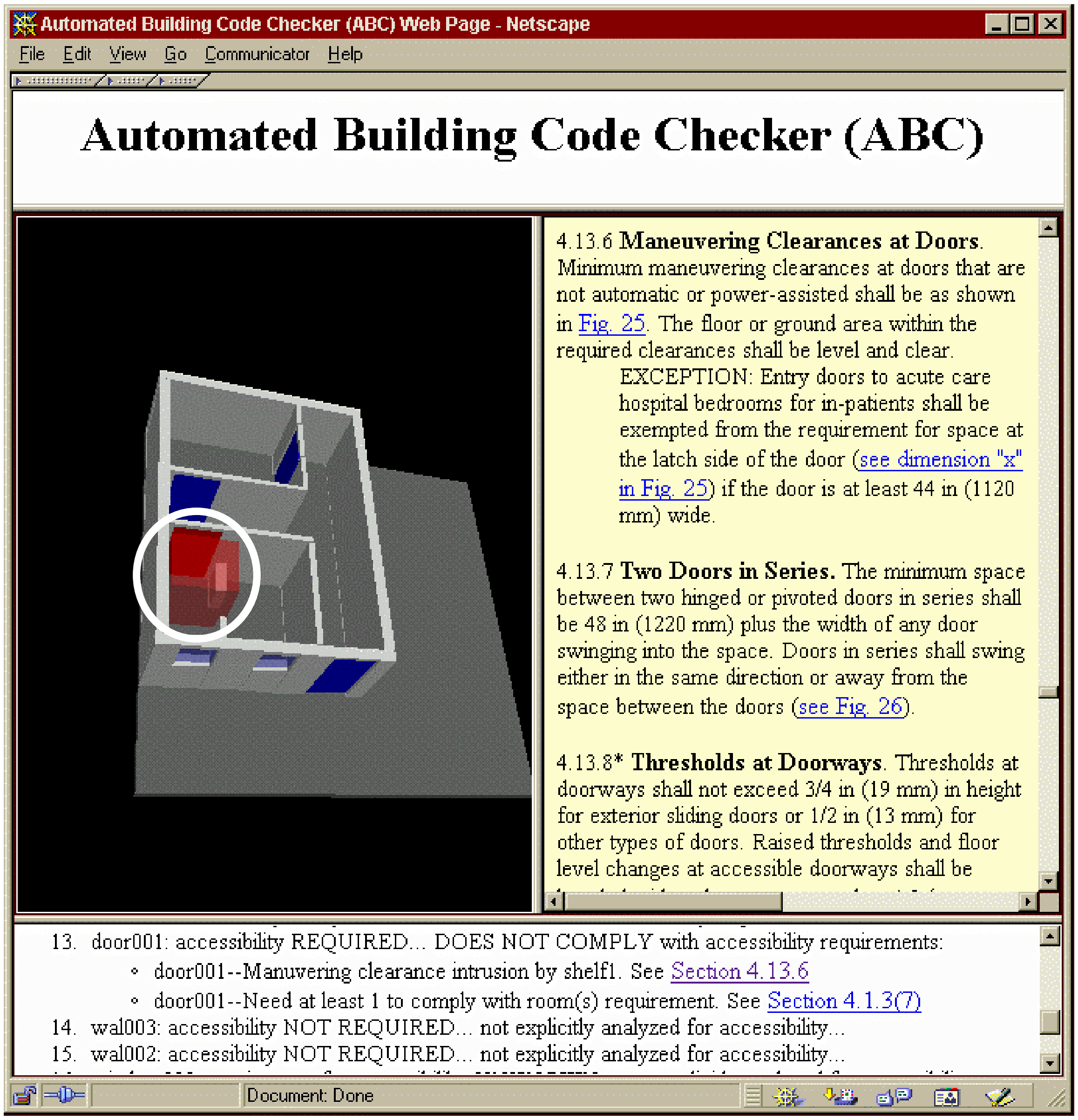
Figure 2.3: The bathroom facility example with one accessible water closet.

usability module is an integral part of the design-intent formulation of the disabled access code.

2.3 The Document Model

Figure 2.4 shows the user interface generated from the document model. The document model consists of two components:

- The standard or code document stored in HTML format.


Automated Building Code Checker (ABC) Web Page - Netscape
File Edit View Go Communicator Help
Document: Done

Automated Building Code Checker (ABC)

4.13.6 **Maneuvering Clearances at Doors**. Minimum maneuvering clearances at doors that are not automatic or power-assisted shall be as shown in Fig. 25. The floor or ground area within the required clearances shall be level and clear.

EXCEPTION: Entry doors to acute care hospital bedrooms for in-patients shall be exempted from the requirement for space at the latch side of the door (see dimension "x" in Fig. 25) if the door is at least 44 in (1120 mm) wide.

4.13.7 **Two Doors in Series.** The minimum space between two hinged or pivoted doors in series shall be 48 in (1220 mm) plus the width of any door swinging into the space. Doors in series shall swing either in the same direction or away from the space between the doors (see Fig. 26).

4.13.8* **Thresholds at Doorways**. Thresholds at doorways shall not exceed 3/4 in (19 mm) in height for exterior sliding doors or 1/2 in (13 mm) for other types of doors. Raised thresholds and floor level changes at accessible doorways shall be

13. door001: accessibility REQUIRED... DOES NOT COMPLY with accessibility requirements:
    - door001--Manuvering clearance intrusion by shelf1. See Section 4.13.6
    - door001--Need at least 1 to comply with room(s) requirement. See Section 4.1.3(7)
14. wal003: accessibility NOT REQUIRED... not explicitly analyzed for accessibility...
15. wal002: accessibility NOT REQUIRED... not explicitly analyzed for accessibility...

Figure 2.4: The report generated from the document model (from [24]).

- A report stored in HTML format generated by the design-intent model analysis that contains analysis comments and hypertext links to the standard or code document.

As illustrated in Figure 2.1, the `Analyze()` associated with the "Analysis" methods of individual `Intent` objects within the instantiated design-intent model contain pointers to the relevant provisions in the standard or code document. In addition to generating the report, the `Analyze()` methods populate the product model with links to the report, and the user activates these links through the graphical interface.

Han et al. first describe this document model in [24]. As shown in Figure 2.4, the left frame in the browser window contains the graphical model of the facility design. In this example, the dark transparent box (circled) represents a clearance violation, and it contains a hyperlink to one of the generated report that appears in the bottom frame in the browser window. The report, in turn, contains a hyperlink to the referenced provision from the standard that is displayed in the right frame in the browser window. In the figure, the user has clicked on a referenced provision in the comment window, and the relevant provision appears in the standard document window.

2.3.1 The Standard or Code Document

Currently, the World Wide Web uses HTML as the standard for publishing text-based information. Users view HTML documents using an HTML-compliant web browser (the browser window's right frame in Figure 2.4 shows the ADAAG). Among its features, HTML provides an *anchor* feature that allows the navigation to a specific position in a document. Figure 2.5 illustrates an ADAAG provision in HTML format, and the line in bold text illustrates the anchor syntax.

The anchor feature along with the ubiquity of web browsers and HTML-based documents makes web-based delivery the logical choice for the standard or code document. The online version of the ADAAG is published using the HTML standard.[5]

[5] See http://www.access-board.gov/bfdg/adaag.htm.

```
<dt>
         <a name="4.13.6">4.13.6</a>
         <b>Maneuvering Clearances at Doors</b>.
         Minimum maneuvering clearances at doors that are not
         automatic or power-assisted shall be as shown in <a
         href="fig25.html">Fig.  25</a>.  The  floor  or  ground
         area within the required clearances shall be level and
         clear.
</dt>
<dd>
         EXCEPTION: Entry doors to acute care hospital bedrooms
         for in-patients shall be exempted from the requirement
         for  space  at  the  latch  side  of  the  door  (<a
         href="fig25.html">see dimension &quot;x&quot; in Fig.
         25</a>) if the door is at least 44 in (1120 mm) wide.
</dd>
```

Figure 2.5: An example provision in HTML format.

2.3.2 The Report

The design-intent model generates the report component of the document model shown in the browser window's bottom frame in Figure 2.4. If an `Intent` instance cannot be fulfilled, the analysis method associated with the `Intent` instance generates a comment in the report that references the required provisions that have not been fulfilled. Otherwise, if the `Intent` instance can be fulfilled, the analysis method associated with the `Intent` instance generates a comment that references the provision that has been fulfilled.

As noted in Section 2.2, the `Intent` instance has a set of pointers to the relevant provisions. Since the document model uses HTML as its document format, these pointers take the form of HTML-compliant references to anchors within the standard or code document. Part of the comment generated for the report will contain these hypertext references so that the user can click on a reference as illustrated in Figure 2.4 and see the associated provision that has been violated.

The comment lines, in turn, contain HTTP-compliant anchors so the user can see the associated comment when clicking the relevant feature in the graphical interface. In addition to generating the report, the design-intent model analysis methods insert this hyperlink information into the product model, and the graphical interface makes use of this hyperlink information as illustrated in Figure 2.4.

2.4 The Prototype

This section gives a brief overview of the prototype implementation. Chapter 5 describes the implementation in detail. The choice of platform and programming languages for the prototype system reflects the following considerations, all of which are inter-related:

- The prototype takes advantage of the ubiquity of the World Wide Web.
- The prototype utilizes object-oriented programming languages in developing the various hierarchical model structures.
- The prototype uses a state-of-the-art distributed object platform that can be integrated with the World Wide Web environment as well as the choice of programming languages.

Toward the first goal, the prototype adopts the web browser as the user interface and uses Java applet technology allowing the user to interact with the design information and the analysis programs. In addition, the *Visual Interactive Environment/Workbench* (VIEW) uses the Virtual Reality Modeling Language (VRML) [34] to provide the graphical interface to the facility model, and a Java-VRML interface, the External Authoring Interface (EAI) allows for the flexible user interaction with the graphical model [45].

This research implements the prototype using the Java and C++ programming languages. The web-based interactive environment between the analysis programs and the graphical user interface is implemented in Java. The prototype also utilizes Java to implement the product model and the design-intent model, both for modeling the objects and for implementing the analysis methods associated with the `Intent` instances. Computationally intensive algorithms such as the motion planning simulations are implemented in C++, and the appropriate Java interfaces provide the link between Java and C++.

Use of the World Wide Web and several design processes naturally lend themselves to leveraging the power of distributed object environments [25]. Furthermore, the hierarchical analysis structure further suggests the use of distributed objects. The prototype uses the Common Object Request Broker Architecture (CORBA) to connect the analysis modules and the instantiated design-intent model's hierarchical structure as well as the analysis of the `Intent` instances within an instantiated design intent structure. In addition CORBA provides interfaces to the Java-developed models [59].

Finally, the document model components conform to HTML format and can be viewed using the web-browser environment. Figure 2.6 illustrates the implemented interfaces, programming languages, and environments used by the prototype.

Figure 2.7 illustrates the VIEW. It acts as the user interface between the designer and the design-aid framework. The VIEW provides the designer with a graphical view of the facility as described by the productmodel. A designer can generate the design using an external CAD package and upload the design to the product model module of the design-aid framework or upload a design from a model repository directly to the product model module.

From the VIEW, the designer can modify the design, interact with the design, or execute an analysis program associated with an instantiated design-intent model. The execution of an analysis program generates the links between the graphical model of the design (viewed in the VIEW) and the generated analysis report and the standard document in the document model.

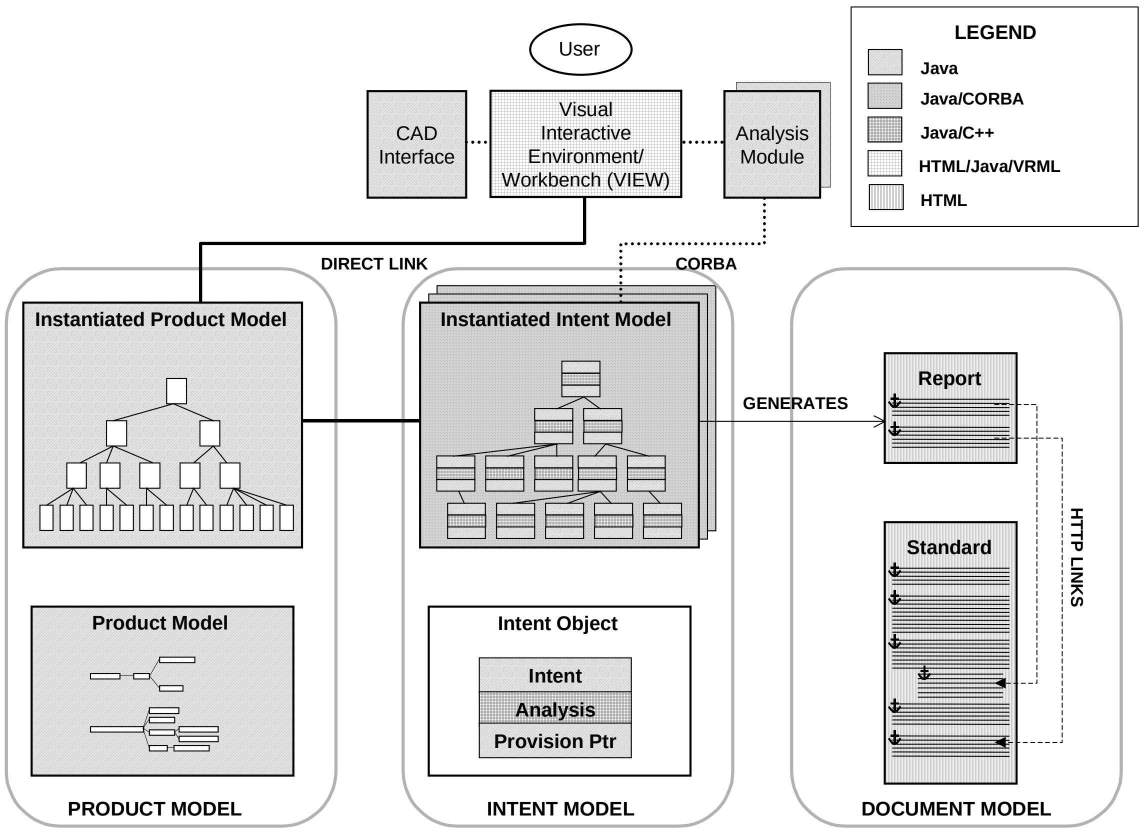
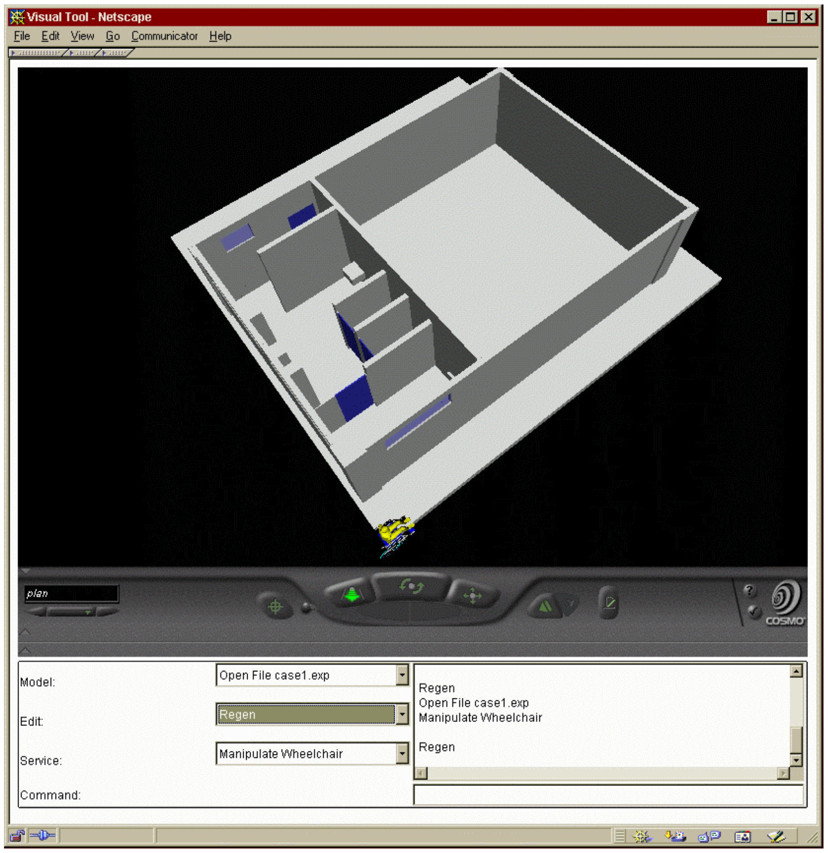


Figure 2.6: The prototype implementation.

Disabled access codes provide guidelines for the equivalent access to facilities for disabled persons, and the accessible route is a main component of this concept [1]. However, the concept of equivalent access cannot be completely mapped to either the prescriptive- or performance-based approach. Thus, the research also provides an interactive environment in which the designer can manipulate a virtual wheelchair through a facility design. The manipulation techniques that utilize the behavior captured by performance-based methods can help the designer further determine the equivalent facilitation of a particular design or configuration. The interactive environment provides the insight that is difficult to quantify such as maneuvering along the accessible route from the wheelchair user's point of view or relaxing the forward-motion constraint by allowing multiple wheelchair backups.



Model: Open File case1.exp

Edit: Regen

Service: Manipulate Wheelchair

Command:

```
Regen
Open File case1.exp
Manipulate Wheelchair

Regen
```

Figure 2.7: The Visual Interactive Environment/Workbench (VIEW).

2.5 Summary

The design-aid framework presented in this chapter consists of three components: a product model, a design-intent model, and a document model. This research uses this design-aid framework to automate disabled access analysis and to provide an interactive wheelchair-manipulation environment.

This chapter introduced the disabled access design-intent model that supports the automated disabled access process. The automated disabled access analysis decomposes into two components: accessible route analysis and system-wide analysis of the facility design. The chapter introduced the two approaches to accessible route analysis. The prescriptive-based approach attempts to capture the prescriptive-based provisions of the ADAAG, and the performance-based approach attempts to directly model wheelchair behavior.

The Visual Interactive Environment/Workbench (VIEW) displays the facility design that it extracts from the product model. From the VIEW, a designer can manipulate the facility design or interact with the design with a simulated wheelchair user. In addition, the designer can send the facility design to the design-intent model to analyze the facility design. The design-intent model extracts the facility design information from the product model, and the design-intent model processes this information in the automated analysis process. The design-intent model then reports the results of the analysis using the document model.

The design-aid framework is strongly influenced by the Hyper-Object-Logic model [60]. In the Hyper-Object-Logic model, the object-logic model aggregates the standards base and the CAD object database. The design-aid framework conceptually separates the design-intent model (that is analogous to the standards base) and the product model (that is analogous to the CAD object database). In addition, an implementation of the Hyper-Object-Logic model explicitly creates subclasses of the standards base member hierarchy and the CAD object database object model. In contrast, the design-aid framework in this work does not create subclasses of the main components of the design-intent model (the `Intent` object) and the product model (a generic component, `GenericComponent`). Instead, populating the various attributes of each respective object differentiates them from one another. This approach provides a more global template to the approach of creating subclasses of the main components.

While the Hyper-Object-Logic model uses object-logic sentences to define methods associated with the standards base method objects, no such restriction exists for the analysis methods associated with the `Intent` object instances. Removing this restriction allows integrating more powerful and flexible methods such as the motion-planning simulations directly into the design-intent model. These analysis methods contribute to the determination of an `Intent` instance's final state (fulfilled, unfulfilled, or unknown), and the results of these analyses are stored directly in the product model.

The design-aid framework uses a much simpler document model than Yabuki's Hyper-Object-Logic HyperDocument model. The design-intent model structure consisting of `Intent` object instances contain pointers to the relevant standard document. The standard document that is fully-contained by the document model adheres to the hypertext markup language (HTTP) standard, the document standard for perusing documents using the hypertext transfer protocol over the World Wide Web.

The design-intent model developed in this research is similar to Garcia's Active Design Documents Model (ADD) [17]. Garcia acquires HVAC design rationale to drive the HVAC design decision-making process, and this research extracts intent from the disabled access building code to produce a hierarchical structure that enables automated analysis of a facility design. Garcia used the ADD to dynamically acquire and organize the design rationale and the associated decisions, and the design-intent model organizes an immutable hierarchy of the design-intent of a specific standards or code document.

To use the ADD, the designer explicitly declares the critical design parameters (for example, the number of stories in the building, the duct ceiling space provided), and the ADD analyzes these parameters using the ADD design rationale network. In contrast, the design-aid framework uses analysis methods that populate the design intent model to analyze an instance of a symbolic product model that represents a facility design.

Finally, a case-specific ADD represents an evolving ad hoc network of inter-dependent design rationales whereas the design-intent model attempts to organize the intent of a standards document in a structure that can be traversed in a sequential manner. The ad hoc nature of the design rational structure allows the designer to modify existing or insert new parameters based on personal experience or case-specific information. In contrast, the hierarchical structure of the design intent model is immutable, but as will be shown in

Chapter 4, the methods that test the intents can be interchanged. The research uses this flexibility to populate the same design-intent model with either the prescriptive-based and performance-based analysis methods.

Chapter 3

A Disabled Access Analysis Design-aid Framework

This chapter examines the manual disabled access analysis of a bathroom facility, and, from this analysis, develops an automated analysis process. Detailed examination of the manual process provides insight into the disabled access code intent and the prescriptive-based disabled access checking process. Examining the manual checking process provides the understanding and the decomposition of the accessible route and helps develop the automated accessible route analysis. As a result, the accessible route analysis can be cast into the context of the system wide facility analysis.

This chapter is organized as follows:

- Section 3.1 describes a typical manual process that is commonly employed in analyzing a facility. The bathroom facility example will be employed to illustrate the process.
- Section 3.2 develops the product model used to describe the facility and store the analysis information.

- Section 3.3 develops a design-intent code model using the concepts taken from the manual analysis.

- Section 3.4 decomposes the accessible route analysis according to the intent for equivalent access to facilities. The accessible route analysis tries to generate an accessible route to all building components. The accessible route analysis represents a first-pass for the disabled access analysis problem and is hereby called first-order analysis.

- The accessible route analysis generates the routes, and these routes must then be examined by the usability analysis. Section 3.5 describes the system-wide usability analysis incorporating the design-intent code model. The system-wide analysis of accessibility and usability is hereby called second-order analysis.

- Finally, Section 3.6 illustrates the automated analysis process for a bathroom facility.

3.1 The Manual Disabled Access Analysis Process

This section describes a typical manual compliance checking for disabled access using the bathroom facility illustrated in Figure 3.1. A building official or designer examines the building components labeled A through K to determine disabled access code-compliance by asking questions about the facility design in a roughly hierarchical manner. Provisions from the ADAAG illustrate specific arguments supporting the analysis with the pertinent points from these provisions italicized.

3.1.1 Manual Analysis of the Bathroom Facility

A. Top Level:

Question: Is this facility subject to the ADAAG disabled access code?

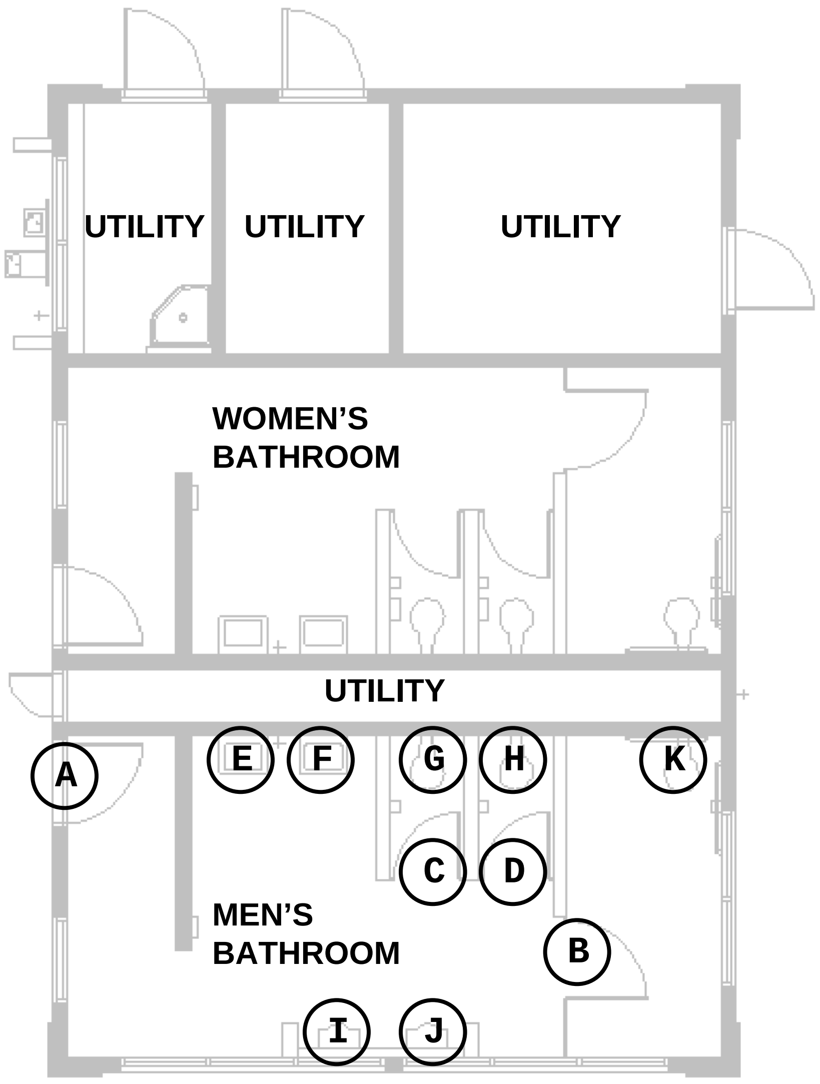


Figure 3.1: The example bathroom facility.

Analysis: The facility is a newly-constructed bathroom building of a middle school.

Relevant ADAAG provisions:

1. PURPOSE. This document sets guidelines for accessibility *to places of public accommodation and commercial facilities* by individuals with disabilities.

4.1.1 Application. (1) General. All areas of *newly designed or newly constructed buildings and facilities* required to be accessible by 4.1.2 and 4.1.3 and altered portions of existing buildings and facilities required to be accessible by 4.1.6 shall comply with these guidelines, 4.1 through 4.35, unless otherwise provided in this section or as modified in a special application section.

Answer: Yes.

A.1 <u>Building/Story Level</u> (there is only one story):

Question: What spaces in this building must be accessible?

Analysis: Bathrooms need to be accessible. Utility spaces do not.

Relevant ADAAG provisions:

4.1.1 Application. (1) General. *All areas* of newly designed or newly constructed buildings and facilities required to be accessible by 4.1.2 and 4.1.3…

4.1.2 Accessible Sites and Exterior Facilities: New Construction. An accessible site shall meet the following minimum requirements:…(6) If toilet facilities are provided on a site, then *each such public or common use toilet facility shall comply* with 4.22. If bathing facilities are provided on a site, then each such public or common use bathing facility shall comply with 4.23…

4.23.1 Minimum Number. *Bathrooms*, bathing facilities, or shower rooms required to be accessible by 4.1 shall comply with 4.23 and shall be on an accessible route.

4.1.1 Application. (5) General Exceptions. (b) Accessibility is not required…(ii) in non-occupiable spaces…*frequented only by service personnel* for repair purposes…

Answer: Men's Bathroom, Women's Bathroom.

A.1.1 <u>Space Level</u>: Men's Bathroom (only the Men's Bathroom is analyzed here):

Question: Does the bathroom comply with clear floor space requirements?

Analysis: A 60-inch turning circle fits into the space.

Relevant ADAAG provisions:

4.23.3 Clear Floor Space. … *An unobstructed turning space* complying with 4.2.3 shall be provided within an accessible bathroom…

4.2.3 Wheelchair Turning Space. The space required for a wheelchair to make a 180-degree turn is a clear space of 60 in (1525 mm) diameter (see Fig. 3(a)) or a T-shaped space (see Fig. 3(b)).

Answer: Yes.

Question: What building components need to be accessible?

Analysis: See cited provisions below.

Relevant ADAAG provisions:

4.1.3 Accessible Buildings: New Construction. Accessible buildings and facilities shall meet the following minimum requirements:…(7) Doors:…(b) Within a building or facility, at least one door at each accessible space shall comply with 4.13.

4.23.2 Doors. *Doors* to accessible bathrooms shall comply with 4.13. Doors shall not swing into the floor space required for any fixture.

4.23.4 Water Closets. If toilet stalls are provided, then *at least one* shall be a standard toilet stall complying with 4.17…

4.23.5 Urinals. If urinals are provided, then *at least one* shall comply with 4.18.

4.23.6 Lavatories and Mirrors. If lavatories and mirrors are provided, then *at least one* of each shall comply with 4.19.

Answer: Doors (including the entry door), at least one water closet, at least one urinal, and at least one lavatory.

A.1.1.1 Building Component Level: Door A

Question: Is Door A accessible (if this door is not accessible, the bathroom is not accessible)?

Analysis: It has sufficient clear width, any approach maneuvering clearance outside the bathroom, and side-approach maneuvering clearance inside the bathroom.

Relevant ADAAG provisions:

4.13.5 Clear Width. Doorways shall have a minimum *clear opening of 32 in* (815 mm) with the door open 90 degrees, measured between the face of the door and the opposite stop (see Fig. 24(a), (b), (c), and (d)). Openings more than 24 in (610 mm) in depth shall comply with 4.2.1 and 4.3.3 (see Fig. 24(e)).

4.13.6 Maneuvering Clearances at Doors. Minimum maneuvering clearances at doors that are not automatic or power-assisted shall be as shown in *Fig. 25*. The floor or ground area within the required clearances shall be level and clear.

Answer: Yes.

A.1.1.2 Building Component Level: water closets.

Question: Is there at least one accessible water closet?

Analysis: Grab Bars: All water closets are in stalls, and Water Closets G and H do not have grab bars, so they are not accessible. Water Closet K has grab bars that comply with requirements.

Relevant ADAAG provision:

4.17.6 Grab Bars. Grab bars complying with the length and positioning shown in Fig. 30(a), (b), (c), and (d) *shall be provided*. Grab bars may be mounted with any desired method as long as they have a gripping surface at the locations shown and do not obstruct the required clear floor area. Grab bars shall comply with 4.26.

Analysis: Clear Floor Space: Water Closet K has sufficient diagonal approach clearance.

Relevant ADAAG provision:

4.16.2 Clear Floor Space. Clear floor space for water closets not in stalls shall comply with *Fig. 28*. Clear floor space may be arranged to allow either a left-handed or right-handed approach.

Analysis: Accessible Route: There is an accessible route between Door A and Water Closet K through Door B (see A.1.1.3).

Relevant ADAAG provisions:

3.5 Definitions. Accessible Route. A continuous unobstructed path connecting all accessible elements and spaces of a building or facility. Interior accessible routes may include corridors, floors, ramps, elevators, lifts, and clear floor space at fixtures. Exterior accessible routes may include parking access aisles, curb ramps, crosswalks at vehicular ways, walks, ramps, and lifts.

4.3.2 Location. (3) *At least one* accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

4.3.3 Width. The minimum clear width of *an accessible route shall be 36 in* (915 mm) except at doors (see 4.13.5 and 4.13.6). If a person in a wheelchair must make a turn around an obstruction, the minimum clear width of the accessible route shall be as shown in Fig. 7(a) and (b).

Answer: Yes, Water Closet K is accessible.

A.1.1.3 Building Component Level: Door B

Question: Is Door B accessible?

Analysis: It has sufficient clear width, front approach maneuvering clearance outside the toilet stall, and side-approach maneuvering clearance inside the toilet stall.

Answer: Yes.

A.1.1.4 <u>Building Component Level</u>: urinals.

Question: Is there at least one accessible urinal?

Analysis: Clear Space: Both urinals have sufficient clear space.

Relevant ADAAG provision:

4.18.3 Clear Floor Space. *A clear floor space 30 in by 48 in* (760 mm by 1220 mm) shall be provided in front of urinals to allow forward approach. This clear space shall adjoin or overlap an accessible route and shall comply with 4.2.4…

Analysis: Accessible Route: There is an accessible route between Door A and both urinals.

Answer: Yes, both are accessible.

A.1.1.5 <u>Building Component Level</u>: lavatories.

Question: Is there at least one accessible lavatory?

Analysis: Clear Space: Both lavatories have sufficient clear space.

Relevant ADAAG provision:

4.19.3 Clear Floor Space. *A clear floor space 30 in by 48 in* (760 mm by 1220 mm) complying with 4.2.4 shall be provided in front of a lavatory to allow forward approach. Such clear floor space shall adjoin or overlap an accessible route and shall extend a maximum of 19 in (485 mm) underneath the lavatory (see Fig. 32).

Analysis: Accessible Route: There is an accessible route between Door `A` and both lavatories.

Answer: Yes, both are accessible.

A.1.1.6 <u>Building Component Level</u>: Door `C` and Door `D`

Analysis: These doors do not have sufficient clear space in the toilet stalls. However, since Water Closets `G` and `H` do not have to be accessible (see A.1.1.2), these doors do not have to be accessible.

3.1.2 Discussion

The above design analysis follows roughly a hierarchical procedure. The strict hierarchical structure breaks down when certain decisions that had to be deferred until the analysis of other components needed to be completed. For example, the accessibility of Water Closet `K` depends on an accessible route, but the accessible route to `K` cannot be determined until the designer or building official determines the accessibility of Door `B`.

A designer or building official could follow the strict hierarchical decomposition by first examining all components that possibly participate in the accessible routes, in this case, examining all the doors first. Indeed, the automated analysis approach taken in this chapter follows this brute-force procedure of examining all of the building components. The disadvantage of this procedure manifests itself in the unnecessary analysis of certain building components such as Doors `C` and `D`.

3.2 A Product Model to Support Disabled Access Analysis

In order to support disabled access analysis, a product model needs to be flexible enough to describe the design of a facility as well as serve as a repository for the data derived from the analysis methods of the design-intent model module. Product modeling requirements depend on the type of analysis. This research adopts the definitions of *form*, *function*, and *behavior* as described by Howard et al. [28].

The geometric description or the *form* of the building components is a critical attribute for accessible route and usability determination since the generation of an accessible route depends on geometric constraints. Equally important is the description of the *function* and *behavior* of a building component or a set of building components. Though some functions can be easily derived from the form of an element (for example, an element's form determines its function as an obstacle), making an element's function explicit alleviates the need for the analysis program to derive more complicated functions. For example, it is simpler to declare an object as a *water closet* as opposed to deriving this function from an element or group of elements. Attaching function to an element has liabilities, such as incorrectly labeling the functionality. Given a declaration of an element's function, an analysis program should first verify that an element functions correctly. For example, a program should verify that a grab bar is attached to a wall and is not floating in space. Finally, explicitly labeling the architectural space functions (Men's Bathroom, Women's Bathroom, and Utility Rooms) is necessary for this accessibility analysis.

Relationships among building components also define the functionality and behavior of the components. Reifying these relationships simplifies the manipulation of features of a building. For example, if the concept of an opening is reified, and a relationship between the opening and the wall is established, the size or placement of the opening in the wall

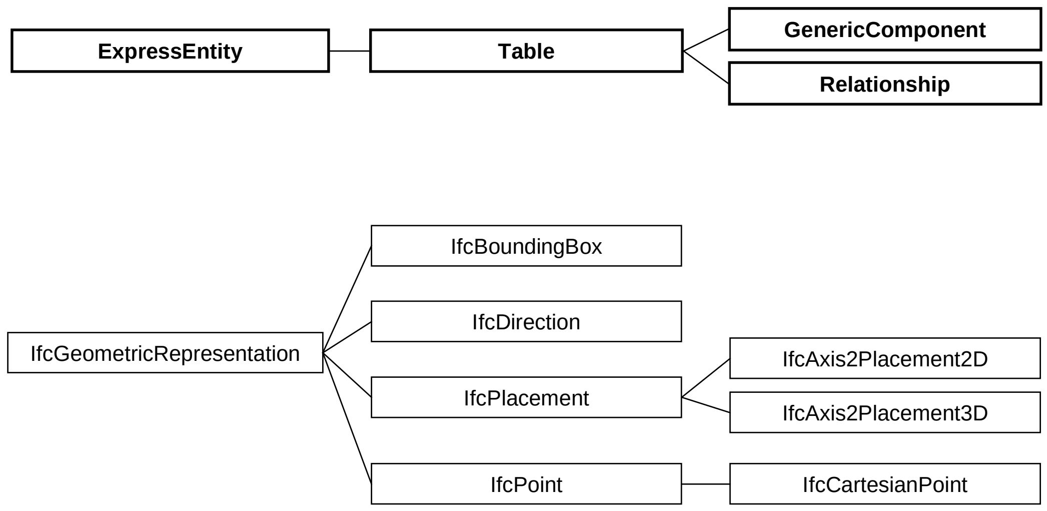


Figure 3.2: The product model hierarchy of objects.

can easily be modified to change the functionality or behavior of the wall. Similarly, reification of certain relationships simplifies the analysis of elements in a design.

This research uses the conceptual structure of the IFC product model described in [33]. Figure 3.2 illustrates the product model hierarchy used that supports the disabled access analysis as well as the geometrical representation hierarchy taken from the IFC geometry. The root object in the hierarchy, `ExpressEntity`, contains an index field taken directly from the EXPRESS methodology of relating EXPRESS schema objects to one another in a generated static file [52]. In addition, there are additional flags that relate to revision information. The `Table` subclass is a table structure that holds a table of keys and their associated values. Figure 3.3 illustrates the EXPRESS schema for these two objects, and Figure 3.4 shows an example of an instantiated `Table` object, a table that represents the form of an `OPENING` object.

The `id` of the `Table` instance shown in Figure 3.4 is `FORM/OPENE-N02`, keys are `SOLID`, `LOCALPLACEMENT`, and `PRODUCTSHAPE`, and the associated values are `FALSE`, `#640`, and `#635` (the numbers `#640` and `#635` refer to other instantiated

```
ENTITY ExpressEntity
      ABSTRACT SUPERTYPE OF (Table);
            index       : INTEGER;
            insert      : BOOLEAN;
            delete      : BOOLEAN;
            replace     : BOOLEAN;
            proxy       : BOOLEAN;
END_ENTITY;


ENTITY Table
      SUPERTYPE OF (ONEOF (
            GenericComponent,
            Relationship
      ))
      SUBTYPE OF (ExpressEntity);
            id          : STRING;
            keys        : LIST [0:N] OF ANY
            values      : LIST [0:M] OF ANY
      WHERE
            WR1: N=M;
END_ENTITY;
```

Figure 3.3: EXPRESS schema for the `ExpressEntity` and `Table` objects.

```
#641 = TABLE ('FORM/OPENE-N02', ('SOLID', 'LOCALPLACEMENT'
'PRODUCTSHAPE'), (.F., #640, #635));
```

Figure 3.4: An example `Table` instance.

objects). `LOCALPLACEMENT` and `PRODUCTSHAPE` refer to IFC geometry objects. The next two subsections describe the concepts and the extensions to the IFC product model employed in this research.

```
ENTITY GenericComponent
      SUBTYPE OF (Table);
      WHERE
            WR1:  keys[0] = "FORM",
                  keys[1] = "FUNCTION",
                  keys[2] = "BEHAVIOR";
END_ENTITY;
```

Figure 3.5: EXPRESS schema for the `GenericComponent` object.

3.2.1 Reification of Components

The product model has a generalized object structure as illustrated in the top figure of Figure 3.2. A building element is an instantiation of the object `GenericComponent`, a subclass of the `Table` object, which restricts the keys of the table to `FORM` (a `GenericComponent` uses the IFC geometric representation objects to capture the form), `FUNCTION`, and `BEHAVIOR`. Figure 3.5 illustrates the EXPRESS schema for the `GenericComponent` object.

The IFC model explicitly reifies certain objects. For example, an opening in the IFC model is represented by an `IfcOpeningElement`. As opposed to explicitly defining certain building component objects, a `GenericComponent` instance represents a specific building component as a key/value associated with the `FUNCTION` and `BEHAVIOR` keys. For example, a `GenericComponent` that represents an opening has an `OPENING` key/value in `Table` instances associated with the component's `FUNCTION` and `BEHAVIOR` keys. By using this model, any IFC building element and any new building components such as ramps (which are not explicitly reified in the IFC Release 1.5 model[6]) can easily be added to the product model. Conversely, since the IFC model explicitly reifies certain objects and their attributes, an analysis program can be

[6] A ramp object (`IfcRamp`) is reified in the IFC Release 2.0 model, but the issue of adding reified objects versus providing an object structure to easily incorporate new objects is still valid.

```
#641 = TABLE ('FORM/OPENE-N02', ('SOLID', 'LOCALPLACEMENT'
'PRODUCTSHAPE'), (.F., #640, #635));
#642 = TABLE ('FUNCTION/OPENE-N02', (OPENING), (OPENING));
#643 = TABLE ('BEHAVIOR/OPENE-N02', (OPENING), (OPENING));
#644 = GENERICCOMPONENT ('OPENE-N02', #641, #642, #643);
```

Figure 3.6: A `GenericComponent` and supporting `Table` instances for an opening.

written to efficiently parse IFC building elements *a priori* by making the function and behavior implicit in the code.

In this research, all the subclasses of the IFC Release 1.5 `IfcBuildingElement` object are represented by equivalent `GenericComponent` objects [33]. For accessibility analysis, the product model needs to support the description of building components such as openings, doors, water closets, lavatories, and urinals. For example, Figure 3.6 describes an instance of an opening object along with the associated `Table` instances. The string `OPENE-N02` denotes the component's `id`, and the three numbers indicate pointers to `Table` objects representing the `FORM`, `FUNCTION`, and `BEHAVIOR` of the component (note that `#641` is the `Table` object described in Figure 3.4).

An analysis method extracts the information from a `GenericComponent` and the supporting `Table` instances. For example, when an analysis method examines the `GenericComponent` shown in Figure 3.6, the method discovers that the `GenericComponent` is an opening from the supporting `FUNCTION` and `BEHAVIOR` `Table` instances. In this example, the supporting `FUNCTION` and `BEHAVIOR` `Table` instances contain the information that this building component functions and behaves like an opening.

Container objects describe specific `GenericComponent` objects that group sets of `GenericComponent` instances. The development of these container objects in the

```
#641 = TABLE ('FORM/OPENE-N02', ('SOLID', 'LOCALPLACEMENT'
'PRODUCTSHAPE'), (.F., #640, #635));
#642 = TABLE ('FUNCTION/OPENE-N02', (OPENING), (OPENING));
#643 = TABLE ('BEHAVIOR/OPENE-N02', (OPENING,
ACCESSIBILITY), (OPENING, #10000));
#644 = GENERICCOMPONENT ('OPENE-N02', #641, #642, #643);
#10000 = TABLE ('ACCESSIBILITY/BEHAVIOR/OPENE-N02',
(ACCESSIBLE), (NULL));
```

Figure 3.7: The modified `GenericComponent` and `Table` instances.

context of the `Relationship` object is discussed in the next section. Container objects support the functional view assumed by different architectural design analyses such as disabled access. As noted in the manual analysis of the bathroom facility in Chapter 3, the disabled access code assumes the description of buildings, stories, and spaces (for example, the Men's Bathroom) in the facility documentation.

Finally, the product model has been designed to be flexible enough for the analysis methods of the design-intent model module to deposit analysis information into the product model by dynamically adding additional keys and associated `Table` instances to an existing `Table` instance. For example, an analysis method can modify the behavior of the relevant opening component by creating an "accessibility" `Table` instance that becomes part of the opening's behavior. Initially, the values for the "accessibility" `Table` cannot be determined, and hence, these fields are undefined. Figure 3.7 illustrates the same opening component shown in Figure 3.6 incorporating a simplified "accessibility" `Table` instance, `#10000`. When a subsequent analysis method examines this modified `GenericComponent`, this building component contains the additional accessibility analysis information.

```
ENTITY Relationship
      SUBTYPE OF (Table);
      WHERE
            WR1:  keys[i]     : GenericComponent,
                  values[i]   : LIST [0:N] OF
                                    GenericComponent;
END_ENTITY;
```

Figure 3.8: EXPRESS schema for the `Relationship` object.

3.2.2 Reification of Component Relationships

As illustrated in Figure 3.2, the `Relationship` object is a subclass of the `Table` object. The `Relationship` object corresponds in functionality to the IFC `IfcRelationship` object [33]. The `Relationship` object explicitly associates one `GenericComponent` with one or more other `GenericComponents` and creates a specific way of viewing of the model alleviating the need for an analysis program to derive this information.

Figure 3.8 illustrates the EXPRESS schema for the `Relationship` object. Note that the `Relationship` object does not add any new attributes, but the keys are now restricted to `GenericComponents` and values are restricted to lists of `GenericComponents`. Specifically, the `Relationship` object associates a specific `GenericComponent` (`keys[i]`) with a list (`LIST [0:N]`) of N `GenericComponents` (`values[i]`).

The IFC model specifies subclasses of the `IfcRelationship` object to refine a specific type of relationship. For example, the `IfcRelVoids` object (a subclass of `IfcRelationship`) explicitly describes the relationship that an opening (`IfcOpening`) has with a solid object such as a wall (`IfcWall`). Similarly, in this research, a `Relationship` object defines the relationship that one or more

```
#1 = RELATIONSHIP ('VOIDS', (#599), ((#614, #644)));
#599 = GENERICCOMPONENT ('WALL-1  , ...);
#614 = GENERICCOMPONENT ('OPENING-1', ...);
#644 = GENERICCOMPONENT ('OPENING-2', ...);
```

Figure 3.9: A `Relationship` instance and the associated `GenericComponents` defining the `VOIDS` relationship.

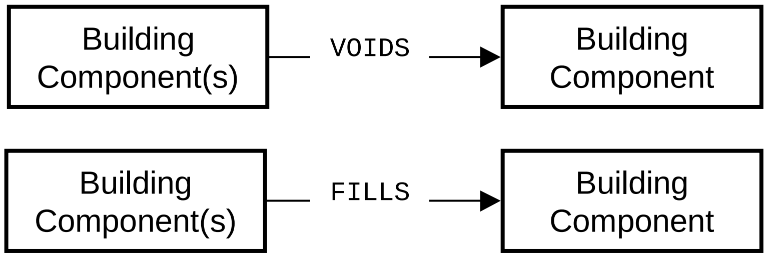


Figure 3.10: The `VOIDS` and `FILLS` relationships.

`GenericComponent` objects have with another `GenericComponent` object—the name of the `Relationship` instance defines its functionality. For example, the `Relationship` object named `VOIDS` corresponds to the `IfcRelVoids` object in the IFC model. A new type of relationship can easily be added to the product model. Conversely, since the IFC model explicitly reifies certain relationships and their attributes, an analysis program can be written to efficiently parse predetermined IFC relationships. Figure 3.9 illustrates an instance of the `VOIDS` relationship. Note that `#599` corresponds to a wall, `#614` corresponds to an opening, and `#644` corresponds to the opening in Figure 3.6.

First-order accessible route analysis takes advantage of the `VOIDS/FILLS` relationships shown in Figure 3.10 to differentiate between open spaces and spaces that constitute openings in walls and which building components fill the openings. The analysis

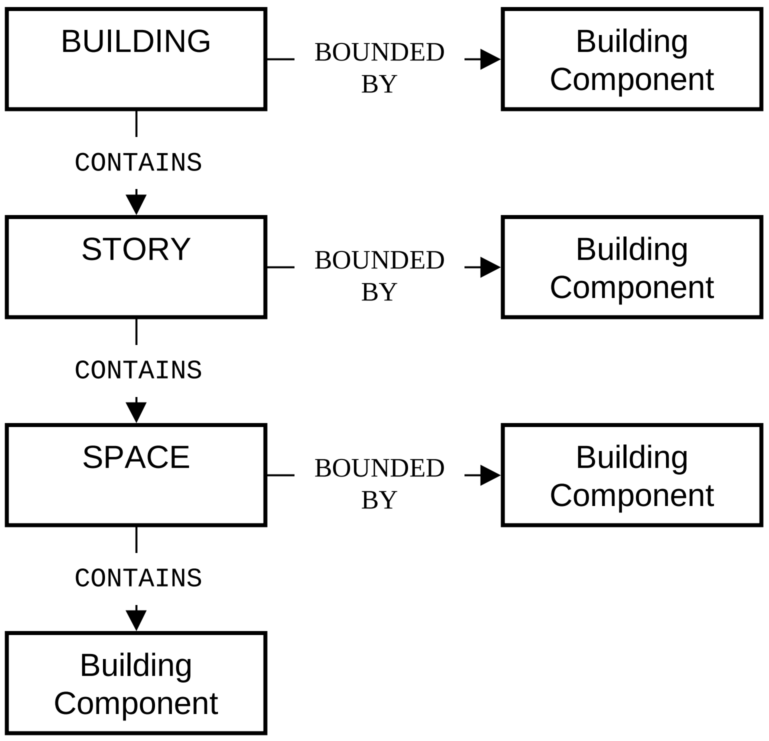


Figure 3.11: Relationship instances defining an architectural view of a facility.

decomposes the accessible route treating open spaces and wall openings differently so explicitly reifying an opening and the relationship between an opening and a wall relieves the analysis from deriving openings in a facility. The different treatments reflect the disabled access code's definition of an accessible route: in an open space, the width of the route is 36 inches, and at an opening, a 32-inch width complies with the code [1].

The `BUILDING`, `STORY`, and `SPACE` container elements (`GenericComponents` which have the functions and behaviors labeled `BUILDING`, `STORY`, and `SPACE`) shown in Figure 3.11 correspond to the `IfcBuilding`, `IfcStorey`, and `IfcSpace` container elements respectively [33]. While first-order accessible route analysis does not need the relationship information shown in Figure 3.11 to construct the accessible routes to the building components, the analysis can take advantage of these relationships by connecting accessible route segments between various similar container objects (for example, between `SPACES`). However, second-order disabled access usability analysis is dependent on the architectural view or interpretation of a design that decomposes a facility into buildings, stories, and spaces. An analysis program could derive this view information, but even for the manual accessibility analysis process, designers make the container relationships in Figure 3.11 explicit by communicating them in design documents. For example, the manual analysis process utilized the explicit labeling of the architectural space functions.

In utilizing the `CONTAINS` and `BOUNDED BY` relationships, this research makes the distinction between a container object and the architectural definition of the container object's name. For example, a `BUILDING` container object is `BOUNDED BY` walls or other building components (walls are not contained within the `BUILDING` container object). In contrast, architecturally, a *building* is composed of the `BUILDING` container object and the building components that define the boundaries of the `BUILDING` container object. A bounding building component can define the boundary of multiple container objects simultaneously. For example, a common wall can define the boundary of two different `SPACE` containers. This distinction allows the straightforward decomposition of a facility based on the architectural function since the common wall will exist in decomposed `SPACE` models.

Finally, analogous to the `GenericComponent` object described in Section 3.2.1, analysis methods in the design-intent model module can instantiate new relationships relevant to the type of analysis. For example, if an analysis method instantiates a maneuvering clearance object, it must create or modify an existing `Relationship` object that associates the clearance object with an opening element.

3.3 The Design-intent Model and Disabled Access

The design-intent model presented in this research organizes the disabled access code's intent to enable the automated disabled access analysis. The model must be populated with the equivalent access to facilities sub-intent of the disabled access code to analyze the equivalent access of a facility design. An instance of an `Intent` object, the main data structure in the model, has methods and attributes (for example, sub-intent `Intent` instances) associated with it. The hierarchical structure of the design-intent model allows an `Intent` object to be populated with sub-intents that represent the decomposition of the `Intent` object.

Considering the decomposition and processing of these member `Intent` objects, the `Analyze()` method has four subroutines:

1. The `PreProcess()` subroutine performs product model analysis that needs to be addressed before the decomposition of the examined `Intent` instance into the sub-intents.

2. The `subIntentPreProcess()`handles any pre-processing of the sub-intents.

3. The `subIntentPostProcess()`handles any post-processing of the sub-intents.

4. The `PostProcess()` subroutine gathers the analysis information from the decomposition of the sub-intents and processes the information.

The analysis of a sub-intent is executed between the `subIntentPreProcess()` and the `subIntentPostProcess()`. If there is more than one sub-intent, the `Analyze()` routine executes a loop of the sub-intent subroutines. The sub-intents are processed recursively using their own `Analyze()` methods.

Referring to the manual disabled access exercise discussed earlier in this chapter, the above subroutines can be mapped to the following actions:

- The `PreProcess()` subroutine maps to analyzing all the doors before decomposing the designed facility.

- The decomposition of the facility (for example, decomposing a story of the facility into spaces) occurs in a `subIntentPreProcess()` subroutine.

- The decomposed facility design analysis occurs in the `subIntents.Analyze()` subroutine.

- The `subIntentsPostProcess()` subroutine recomposes the decomposed facility and the results from the `subIntents.Analyze()` subroutine.

- The `PostProcess()` subroutine produces the final analysis of the facility.

Finally, if the `Intent` object can be associated with one or more code provisions, the `Intent` object provides pointers to the appropriate provisions that reside in the document model. Figure 3.12 shows the `Intent` object and its methods in Java-like syntax. Using the `Intent` class structure shown in Figure 3.12, the methods associated with the `Intent` object operate on and modify a design of the facility or an instance of a `ProductModel` object. Since the model of the facility is shared across all `Intent` object instances, the `Intent` object instances must be able to resolve disparate modifications on common components, especially when several of the analysis methods are run in parallel. For the disabled access code, the decomposition of the facility model and the decomposition of the analysis analyze and modify separate portions of the model. For example, the analysis decomposes a building story into the defined spaces within the story, and the analysis of one space does not affect the analysis of another space.

As illustrated in the disabled access manual checking process, a facility's compliance with the equivalent access to facilities depends on two main levels of analysis:

1. Is there an accessible route to the building components that is usable by disabled persons (accessible route generation/determination)?

```
class Intent
{
     public  Analyze() {
          this.PreProcess();
          for (int i=0; i < subIntents.length; i++) {
               this.subIntentsPreProcess(i);
               subIntents[i].Analyze();
               this.subIntentsPostProcess(i);
          }
          this.PostProcess();
     }

     private PreProcess() {
          ...
     }

     private PostProcess() {
          ...
     }

     private subIntentsPreProcess(int i){
          ...
     }

     private subIntentsPostProcess(int i){
          ...
     }

     public  ProductModel    model;

     private ProductModel    subModels[];
     private Intent          subIntents[]
     private Provision       provisions[];
}
```

Figure 3.12: The `Intent` object.

2. Are the building elements or some fraction of the building components usable by disabled persons (system-wide usability analysis)?

The two analyses are inter-dependent. The accessible route to a building component can depend on the accessibility of the other building components in the facility, and the usability of a building component depends on the existence of an accessible route to the component. This inter-dependency dictates that these sub-intents cannot be processed concurrently. These two analyses populate the equivalent access to facilities `Intent`

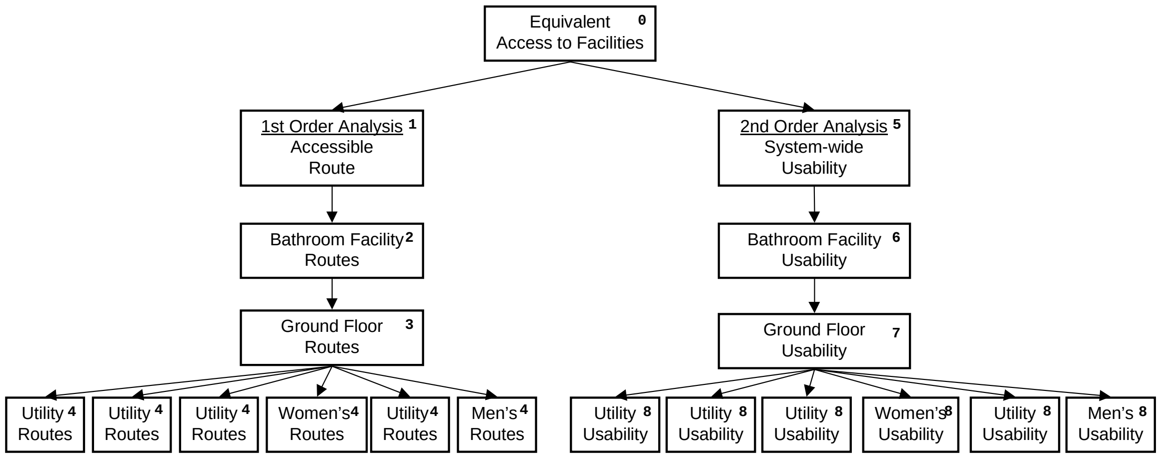


Figure 3.13: The design-intent disabled access model for the bathroom facility.

instance, and the accessible route analysis can be thought of as first-order analysis and the system-wide usability analysis can be thought of as second-order analysis in this process.

The accessible route generation/determination intent and the system-wide usability analysis intent further decompose the bathroom facility by architectural function. Figure 3.13 illustrates the hierarchical structure of the equivalent access to facilities decomposition for the bathroom facility. It is worth noting here that the decomposition takes advantage of the product model container relationships shown in Figure 3.11. Figure 3.13 shows the basic decomposition of the intents as well as the sequence in which the intents are to be executed.

The sequential ordering scheme addresses the higher-order logical structure inherent in the code. In this example, second-order analysis is labeled with a “5,” and cannot be executed until all of the processes labeled “1” through “4” have been completed. As shown in the figure, the *Building Routes* `Intent` instance in Figure 3.13 has only one sub-intent, the *Story Routes* `Intent` instance. However, the *Building Routes* `Intent` instance `Analyze()` method can spawn as many parallel *Story Routes* `Intent` instance `Analyze()` processes as necessary, and each of the process can be executed in parallel. Figure 3.13 shows the decomposition of the facility with one story and the six spaces: the Men's bathroom, the Women's bathroom, and the four utility rooms.

3.4 The Accessible Route Analysis Process

3.4.1 Decomposition of the Accessible Route

In the hierarchically-structured design-intent structure of the disabled access code model, a disabled accessible route is formulated as follows:

Proposition 1 *Let* R *represent an accessible route. Then the route* R *is a composition of accessible components:*

$$\mathsf{R} = \mathsf{R}_{\mathtt{init}} + \Sigma\ \mathsf{R}_{\mathtt{sos}} + \mathsf{R}_{\mathtt{goal}} \qquad (1)$$

where:

$\mathsf{R}_{\mathtt{sos}}$ = $\mathsf{R}_{\mathtt{seg}} <+ \mathsf{R}_{\mathtt{open}} + \mathsf{R}_{\mathtt{seg}}>$,

$\mathsf{R}_{\mathtt{init}}$ = *the initial point* (the starting point of an accessible route),

$\mathsf{R}_{\mathtt{goal}}$ = *the goal point* (the ending point of an accessible route),

$\mathsf{R}_{\mathtt{seg}}$ = *a segment of the accessible route within a space,*

$\mathsf{R}_{\mathtt{open}}$ = *the clearance area at an opening*, and

<> = *optional arguments*

$\mathsf{R}_{\mathtt{init}}$ and $\mathsf{R}_{\mathtt{goal}}$ nodes may also be instances of $\mathsf{R}_{\mathtt{open}}$ nodes.

The following are some examples of accessible routes using this notation:

1. The expression

$$\mathsf{R} = \mathsf{R}_{\mathtt{init}} + \mathsf{R}_{\mathtt{seg}} + \mathsf{R}_{\mathtt{goal}} \tag{2}$$

represents the shortest accessible route. Note that $\mathsf{R}_{\mathtt{seg}}$ can represent either a horizontal path or a vertical path as in the case of an elevator shaft.

2. The expression

$$\mathsf{R} = \mathsf{R}_{\mathtt{init}} + \mathsf{R}_{\mathtt{seg1}} + \mathsf{R}_{\mathtt{seg2}} + \mathsf{R}_{\mathtt{seg3}} + \mathsf{R}_{\mathtt{goal}} \tag{3}$$

can be used to describe a configuration in which a disabled access ramp ($\mathsf{R}_{\mathtt{seg2}}$) connects two segments in the accessible route.

3. Finally, the expression

$$\mathsf{R} = \mathsf{R}_{\mathtt{open}} + \mathsf{R}_{\mathtt{seg}} + \mathsf{R}_{\mathtt{open}} \tag{4}$$

describes the required accessible route of a turning circle within a space (in some spaces, the disabled access code requires that a wheelchair must be able to get in and out of a space without having to back up).

3.4.2 The Brute-force Approach

This research develops a first-order brute-force approach to identify accessible routes composed of the accessible route components defined above. Once the first-order analysis identifies the accessible routes in a facility design, the second-order usability analysis examines the routes. This research uses the following accessible route determining procedure:

1. Establish R_{init}.

2. Analyze all openings to look for potential R_{open} accessible route components.

3. Analyze all other building components to look for potential R_{goal} accessible route components.

4. Construct all possible R_{seg} between pairs of accessible building components generated from steps 1 and 2.

5. Construct all possible R from accessible route components generated in steps 1, 2, 3, and 4.

Following the `Analyze()` features illustrated in Figure 3.12:

- The `PreProcess()` subroutine performs the first two steps.

- The child process `Analyze()` method that decomposes the facility into its architectural spaces and performs the accessibility analysis of these spaces concurrently in separate sessions executes Steps 3 and 4.

- Once all the child `Analyze()` sessions complete and return the $\mathsf{R}_{\mathtt{goal}}$ and $\mathsf{R}_{\mathtt{seg}}$ analyses, the the `PostProcess()` subroutine performs Step 5 and composes the results.

Figure 3.14 shows selected accessible routes for the Men's bathroom in the bathroom facility generated by the analysis procedure. The nodes of the generated graph consist of potential $\mathsf{R}_{\mathtt{open}}$, $\mathsf{R}_{\mathtt{init}}$, and $\mathsf{R}_{\mathtt{goal}}$ accessible route components, and $\mathsf{R}_{\mathtt{seg}}$ arcs connect these nodes. An $\mathsf{R}_{\mathtt{init}}$ node (Door A) represents a "root" node since it is a starting point in the accessible route graph (note that Door A is also an $\mathsf{R}_{\mathtt{open}}$ node). $\mathsf{R}_{\mathtt{goal}}$ nodes

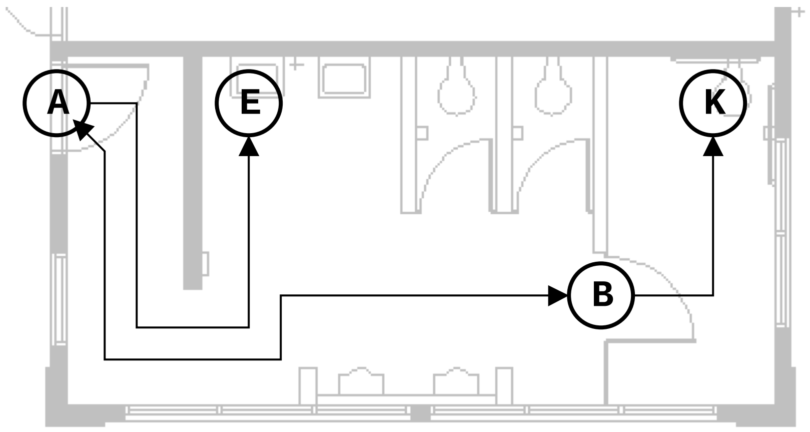


Figure 3.14: Selected accessible routes in the Men's bathroom.

(Sink E and Toilet K) represent "leaf" nodes since they are ending points in the accessible route graph. The generated graph is partially directed. The R_{seg} arcs are one-way toward the R_{goal} nodes and two-way when connecting two R_{open} nodes (the arc between Door A and Door B).

Table 3.1 summarizes the characteristics of this graph. Once the analysis generates the graph, the procedure determines if the goal points are reachable, and the results are forwarded to the second-order usability analysis procedure.

3.4.2.1 Establishing R_{init}

The accessible route analysis procedure establishes the starting point of the route R_{init} (or multiple R_{init} nodes). The type of building component associated with R_{init} depends on the level of analysis of the design. In most cases, an opening in a building provides the connection between the exterior and the interior of the largest container being examined serves as the R_{init} node. If the largest container in the design is a SITE

Table 3.1: Characteristics of the accessible route graph.

R Component	Graph Component	Comments
R_{seg}	arc	uni- or bi-directional depending on connecting nodes
R_{open}	node	
R_{init}	"root" node	R_{seg} arcs directed away from the node
R_{goal}	"leaf" node	R_{seg} arcs directed toward the node

Table 3.2: Rules for establishing the R_{init} building component.

Level of Design	Type of Building Component for Initial Point
SPACE	OPENING(S) on the boundary of a SPACE
STORY	OPENING(S) on the boundary of a STORY
BUILDING	OPENING(S) on the boundary of a BUILDING
SITE	ENTRY(IES) on the boundary of a SITE

object, the ENTRY to this SITE that lies on the boundary of the SITE object serves as the R_{init} node. Table 3.2 summarizes the relationship between the developed product model container level and the building component associated with the R_{init} node.

This step in the accessible route analysis procedure establishes the potential R_{init} accessible route components, but any number of potential R_{init} components may be discarded in the subsequent steps of the accessible route analysis procedure. If all potential $\mathsf{R}_{\mathtt{init}}$ components are discarded, then, by default, there is no accessible route for the given design.

3.4.2.2 Analyzing Potential $\mathsf{R}_{\mathtt{open}}$ Accessible Route Components

After labeling the potential $\mathsf{R}_{\mathtt{init}}$ accessible route components, the accessible route analysis procedure examines all `OPENING` objects in the facility for potential $\mathsf{R}_{\mathtt{open}}$ accessible route components. As already noted, this accessible route analysis does local usability analysis on the accessible route components, and the usability of the `OPENING` objects depends on the implemented usability analysis.

In general, there may be several usability options for an `OPENING` object, and which options are applicable may not be determinable *a priori*. For example, a user of the facility can approach an opening from several directions. The accessible route analysis procedure determines which approaches are valid in the step that generates the $\mathsf{R}_{\mathtt{seg}}$ route segments.

3.4.2.3 Analyzing Potential $\mathsf{R}_{\mathtt{goal}}$ Accessible Route Components

The accessible route analysis procedure determines which building components qualify as potential $\mathsf{R}_{\mathtt{goal}}$ accessible route components. This step in the accessible route generating procedure exploits the architectural `SPACE` and its associated relationships illustrated in Figure 3.11 in the product model development in Section 3.2.2, the advantage being the possible concurrent execution of multiple space analyses. As with the analysis of potential $\mathsf{R}_{\mathtt{open}}$ components, determining the usability of a building component depends on analysis implementation. For example, in this research, the prescriptive-based analysis and the performance-based analysis can yield different results as code-compliance and usability are not necessarily the same. There may be several context-dependent criteria that determine a component's usability. As with the $\mathsf{R}_{\texttt{open}}$ analysis, some of the criteria may be resolved in this first-order accessible route analysis procedure. For example, in the prescriptive-based analysis, if the potential $\mathsf{R}_{\texttt{goal}}$ component has no maneuvering clearance, it no longer qualifies as a potential $\mathsf{R}_{\texttt{goal}}$ component.

Some criteria may need to be resolved in the second-order usability analysis. For example, if the usability of a water closet is determined following the maneuvering clearance rules of the ADAAG, the geometry of this clearance area depends on the disabled access usability of other water closets in the same bathroom space. This example shows the advantage of using a product model that explicitly defines the SPACE object that represents the functionality of a bathroom space. Otherwise, the analysis program would need to derive the bathroom function within a facility.

3.4.2.4 Generating $\mathsf{R}_{\texttt{seg}}$ Route Segments

The final step in the accessible route analysis procedure connects the $\mathsf{R}_{\texttt{open}}$ and the $\mathsf{R}_{\texttt{goal}}$ nodes with $\mathsf{R}_{\texttt{seg}}$ arcs as shown in Figure 3.14. As with the $\mathsf{R}_{\texttt{goal}}$ analysis, this step in the accessible route generating procedure utilizes the architectural space concept and its associated relationships in the product model. Indeed, this final step in the procedure can generate the $\mathsf{R}_{\texttt{seg}}$ arcs without the decomposition of the facility design into SPACE object by searching for $\mathsf{R}_{\texttt{seg}}$ arcs from any $\mathsf{R}_{\texttt{open}}$ to any $\mathsf{R}_{\texttt{goal}}$ or any $\mathsf{R}_{\texttt{open}}$ to any other $\mathsf{R}_{\texttt{open}}$. However, with the facility elements grouped by architectural SPACE objects, the $\mathsf{R}_{\texttt{seg}}$ arc generation step in the procedure can decompose the problem by examining each SPACE object. This step generates the $\mathsf{R}_{\texttt{seg}}$ arcs from any $\mathsf{R}_{\texttt{open}}$ to any $\mathsf{R}_{\texttt{goal}}$ or any $\mathsf{R}_{\texttt{open}}$ to any other $\mathsf{R}_{\texttt{open}}$ associated with the space. After examining all the SPACE objects, the procedure completes generating the accessible route graph since, by definition, an opening and the associated $\mathsf{R}_{\texttt{open}}$ connects adjoining SPACE objects or one SPACE object with the area outside the facility design.

Since a single building component may be associated with several R components (openings are associated with $\mathsf{R}_{\texttt{open}}$ and possibly $\mathsf{R}_{\texttt{goal}}$ nodes, and other building components are associated with $\mathsf{R}_{\texttt{goal}}$ nodes), the procedure attempts to generate $\mathsf{R}_{\texttt{seg}}$ arcs between all these nodes within the SPACE object. Some or all of the R components associated with a building component may be pruned since the $\mathsf{R}_{\texttt{open}}$ and $\mathsf{R}_{\texttt{goal}}$ nodes establish local usability of the building component and the generation of the $\mathsf{R}_{\texttt{seg}}$ arc determines whether the building component is reachable and thus accessible.

3.4.2.5 Traversing the Accessible Route Graph

The $\mathsf{R}_{\texttt{seg}}$ arc generation procedure may produce several disconnected graphs instead of one continuous graph since the procedure starts the accessible route construction in each SPACE object. Therefore, after the procedure generates all valid $\mathsf{R}_{\texttt{seg}}$ arcs, the procedure traverses the graph structure without allowing cycles (visiting a node more than once) starting at all $\mathsf{R}_{\texttt{init}}$ nodes and attempting to reach all $\mathsf{R}_{\texttt{goal}}$ nodes.

Successfully reaching a desired building component does not guarantee the accessibility of the component, but any building component that is not reachable with this traversal is guaranteed to be inaccessible. Using the generated graph structure and the traversal information, the second-order system-wide usability analysis then determines the accessibility of the facility.

3.5 System-wide Usability Analysis

The manual analysis process of the bathroom facility example does not address the decomposition of the facility into stories since it is a one-story building. However, the analysis of a multistory building can be decomposed and is subject to the following ADAAG provisions:

4.1.3 Accessible Buildings: New Construction. Accessible buildings and facilities shall meet the following minimum requirements:…(5) One passenger elevator complying with 4.10 shall serve each level, including mezzanines, in all multi-story buildings and facilities unless exempted below. If more than one elevator is provided, each full passenger elevator shall comply with 4.10.

EXCEPTION 1: Elevators are not required in facilities that are less than three stories or that have less than 3000 square feet per story unless the building is a shopping center, a shopping mall, or the professional office of a health care provider, or another type of facility as determined by the Attorney General. The elevator exemption set forth in this paragraph does not obviate or limit in any way the obligation to comply with the other accessibility requirements established in section 4.1.3. For example, floors above or below the accessible ground floor must meet the requirements of this section except for elevator service. If toilet or bathing facilities are provided on a level not served by an elevator, then toilet or bathing facilities must be provided on the accessible ground floor…

The building-level analysis executes story-level sub-intent analysis for the stories in a building. The story-level analysis returns its results to the building-level analysis that maps the rules from the above provisions. For the example bathroom facility, since there is only one story, the above rules are not relevant.

Continuing down the hierarchy of the system-wide usability analysis and as noted in the previous section, the analysis procedure examines each story of a building concurrently. The story-level sub-intent provides equal access to facilities for disabled persons on each story of both building components and spaces. For example, the following provisions from [1] fulfill this story-level equivalent-access-to-facilities sub-intent for drinking fountains (building components) and for dressing rooms (spaces):

4.1.3 Accessible Buildings: New Construction. Accessible buildings and facilities shall meet the following minimum requirements:…

(10) Drinking Fountains:…(b) Where more than one drinking fountain or water cooler is provided on a floor, 50% of those provided shall comply with 4.15 and shall be on an accessible route…

(21) Where dressing and fitting rooms are provided for use by the general public, patients, customers or employees, 5 percent, but never less than one, of dressing rooms for each type of use in each cluster of dressing rooms shall be accessible and shall comply with 4.35…

Analogous to the building-level decomposition, the story-level analysis executes space-level sub-intent analysis for each space of the story in question concurrently utilizing the child `Analyze()` subroutine. Once these subroutines return their analysis results, the story-level analysis maps the rules such as the provisions shown above in the `PostProcess()` subroutine.

The space-level analyses constitute the leaves at the lowest levels of the hierarchically structured system-wide usability sub-intent. A story-level sub-intent invokes the space-level sub-intents, and these space-level sub-intents can be processed concurrently. A space-level sub-intent examines the building components contained by the space. As in the manual analysis example, similar functioning building components are grouped together to determine the space's accessibility. For example, when determining the accessibility of the bathroom facility, a designer or inspector poses the question: What building components need to be accessible? The automated space-level analysis references the same prescribed parameters in the building-component-level provisions. For example the provision for water closets from the ADAAG is given as follows:

4.23.4 Water Closets. If toilet stalls are provided, then *at least one* shall be a standard toilet stall complying with 4.17…

The previous accessible route analysis determined which water closets in the space have accessible routes. From this information, the `Analyze()` method for the space-level sub-intent examines the water closets with accessible routes for other usability criteria (if there are any) and make the determination about the accessibility of the space based on the accessibility of the building components. The system-wide usability analysis does not further decompose the space-level sub-intent and maps the relevant space-level provisions to rules to be executed in the `PostProcess()` subroutine in the space-level sub-intent.

3.6 Case Example

3.6.1 First-order Accessible Route Analysis

The following analysis refers to the Men's bathroom space of the bathroom facility shown in Figure 3.15. Doorway A maps to $\mathbf{R}_{init}$; Doorways B, C and D map to $\mathbf{R}_{open}$ nodes; building components E through K map to $\mathbf{R}_{goal}$ nodes; the connections between the nodes map to $\mathbf{R}_{seg}$ arcs.

A wheelchair user can transfer from the chair to the toilet using two different methods: from the side of the toilet (side transfer) or approaching the toilet diagonally (diagonal transfer). Thus, the accessible route analysis established two goal nodes for Water Closet K, K_1 and K_2. The analysis eliminates Doorways C and D as potential $\mathbf{R}_{open}$ components since they do not have the sufficient clearance requirements inside the toilet stalls.

As shown by the highlighted accessible routes in Figure 3.16, the path from Doorway A to Water Closet K is represented by two accessible routes:

$$\mathbf{R}_1 = A + \mathbf{R}_{seg1} + B + \mathbf{R}_{seg2} + K_1 \quad (5)$$

$$\mathbf{R}_2 = A + \mathbf{R}_{seg1} + B + \mathbf{R}_{seg3} + K_2 \quad (6)$$

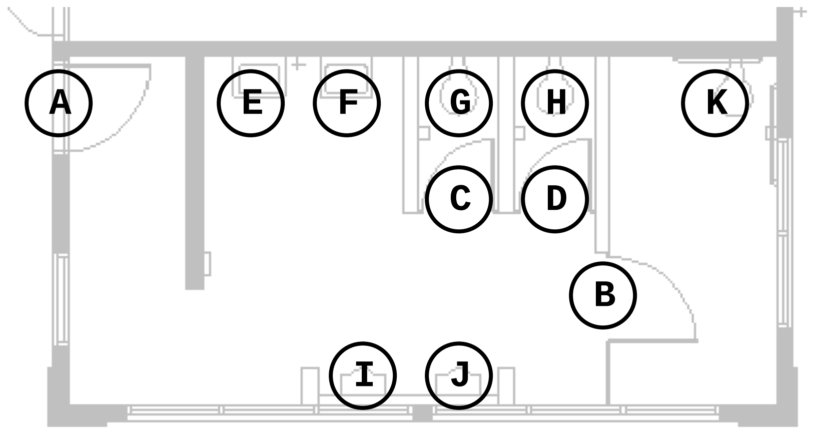


Figure 3.15: The labeled potential accessible route components in the bathroom facility.

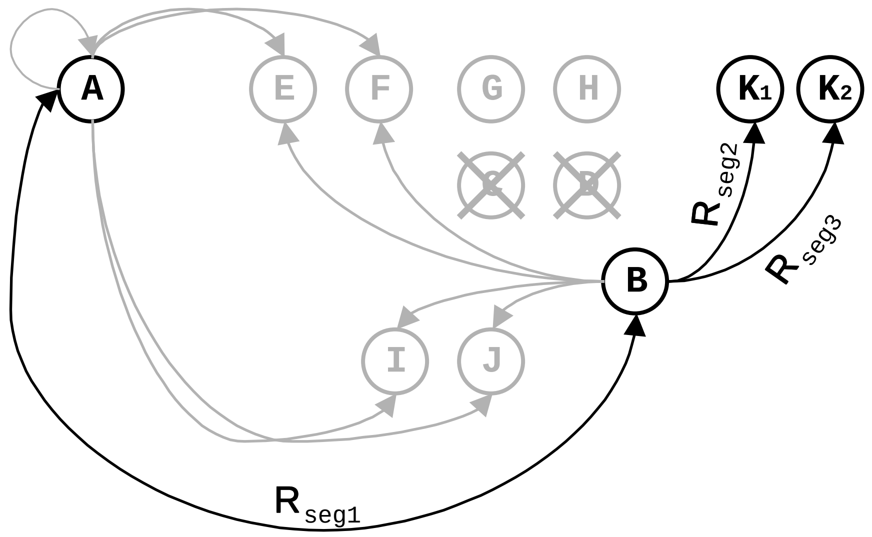


Figure 3.16: The accessible route graph for the bathroom facility.

The requirement for a turning circle within a space can be represented as

$$\mathbf{R} \quad = A + \mathbf{R}_{seg} + A \tag{7}$$

Figure 3.16 shows all the accessible routes generated for the bathroom facility including the turning circle requirement (shown by the arrow from A back to A).

3.6.2 Second-order System-wide Usability Analysis

Once the first-order accessible route analysis has generated accessible route information, the second-order system-wide analysis can make the final determination of the bathroom facility's accessibility. Following the second-order analysis in the design-intent model shown in Figure 3.13, the analysis decomposes the bathroom facility into one story and the one story into five spaces (the Men's bathroom, the Women's bathroom, and the three utility rooms). The analysis starts at the space-level sub-intent (in this exercise, the analysis only examines the Men's bathroom) and progress back up the hierarchical structure to the story-level sub-intent and the building-level sub-intent.

At the space-level sub-intent, the analysis checks the usability of the groups of building components (doorways, water closets, urinals, and lavatories). For example, for the water closets, only Water Closet K has an accessible route, so the space-level sub-intent analysis checks other usability parameters for this water closet such as the existence of code-compliant grab bars. As in the manual analysis case, the space-level analysis determines that the Men's bathroom contains the sufficient number of required building components and determines that the bathroom space complies with disabled access requirements.

The story-level sub-intent analysis determines that the story complies with the story-level disabled access requirements assuming the Women's bathroom also complies (the utility rooms do not have to comply with disabled access requirements). This analysis passes the results up the hierarchy to the building-level sub-intent analysis.

Finally, the building-level sub-intent analysis determines the bathroom facility complies with disabled access requirements since the story in the building complies with the requirements and, in this particular case, building contains only one story. If the facility had contained more than one story then the building-level analysis would need to determine the accessibility of all the stories.

3.7 Summary

This chapter started by stepping through the manual process of disabled access analysis using a bathroom facility as the test case. From this process, the chapter developed the necessary models, a product model and a design-intent code model, to support an automated disabled access process. The design-intent code model for disabled access requires two main reasoning components:

1. A determination and analysis of the accessible routes in a facility

2. Using these accessible routes, a determination of the system-wide usability of the facility.

The chapter describes and develops the methods to automate these two analyses.

Finally, the chapter steps through the automated analysis process revisiting the bathroom example. The manual and automated analyses are similar. The automated analysis is a brute-force procedure sequentially processes the two main components described above. In contrast, in the manual procedure, the person performing the analysis can shift between the determination of accessible routes and the requirement of these routes in the context of the whole system of the facility.

Chapter 4

Accessible Route Analysis: A Prescriptive-based, a Performance-based, and an Interactive Model

The previous chapters have developed the design-aid framework for disabled access analysis and a decomposition of the accessible route that is dependent on the building components in a facility design. This chapter describes the automated analysis implementation for the determination of the accessible route to enable the overall automated disabled usability analysis. The chapter first develops methods for the ADAAG prescriptive-based accessible route analysis and then uses some of the prescriptive methods to develop the performance-based accessible route analysis methods. The prescriptive methods are the basis for some of the performance-based usability parameters.

In addition, the chapter describes wheelchair manipulation tools and visualization methods that provide the designer with an interactive environment in which the designer can manipulate a virtual wheelchair through the facility design. The manipulation and visualization tools provide the designer further insight into the disabled access problem supplementing the developed prescriptive-based and performance-based methods.

The chapter is organized as follows:

- Section 4.1 describes the prescriptive-based analysis that captures the ADAAG accessible route provisions.

- Section 4.2 describes the performance-based accessible route analysis. Both the prescriptive-based and performance-based analyses use motion-planning techniques to generate the accessible routes, and the differences in the developed techniques is described in the respective sections.

- Section 4.3 provides a comparison between the prescriptive-based and performance-based analyses with several examples. A major goal of this thesis is to validate a performance-based analysis as a complementary component to the prescriptive-based code-compliance. These examples demonstrate the deficiencies of the prescriptive-based accessible route analysis and the power of the performance-based analysis.

- Finally, Section 4.4 describes the various wheelchair manipulation and animation algorithms developed to support this research.

4.1 Automated Prescriptive-Based Analysis: The Code-Compliant Accessible Route

The implementation of the prescriptive-based accessible route analysis utilizes the parameters prescribed in the ADAAG as described in the manual analysis of the bathroom facility in the previous chapter. The analysis uses motion-planning techniques to generate the code-compliant accessible routes within `SPACE` container objects (the `GenericComponent` object in the product model that describes a space) in a facility design. The motion planner generates a path between an *initial* point and a *goal* point. Building components along the accessible route graph map to the required `R` nodes:

$\mathsf{R}_{\mathtt{open}}$ nodes map to initial and goal points, $\mathsf{R}_{\mathtt{init}}$ nodes map to initial points, and $\mathsf{R}_{\mathtt{goal}}$ nodes map to goal points. The arcs of the graph (the $\mathsf{R}_{\mathtt{seg}}$ components) map to the generated path between the R nodes.

4.1.1 Motion Planning Basics

In the basic motion-planning problem, a robot A moves through a Euclidean space W (the *workspace*) represented as $\mathbf{R}^N$ where $\mathbf{R}$ is the set of real numbers, and $N = 2$ or 3. This research assumes a two-dimensional space motion planner, and $N = 2$. Obstacles represented as B_1, $\mathsf{B}_2 \ldots \mathsf{B}_q$, populate W, and the motion planner accurately knows the positions and geometric parameters (shape, position, and orientation) of A, the B_is, and W. The motion planner tries to generate a continuous path τ through the workspace W for the robot A avoiding the obstacles B_is given an initial position and orientation and a goal position and orientation. If no path exists, the motion planner reports failure.

The motion planner generates a *configuration space* C from the geometric properties of A, the B_is, and W and attempts to construct the path in this configuration space. In the new space C, the motion planner transforms robot A to a point object, and the motion-planning problem becomes one of generating the path τ in C. If the dimension of W is 2 ($\mathsf{W} = \mathbf{R}^N = \mathbf{R}^2$), then the dimension m of C is 3. For example, a robot A restricted to move in the xy-plane ($\mathsf{W} = \mathbf{R}^2$) has three degrees of freedom: x, y, and the orientation θ. Of course, if A is a disk or is not free to rotate, $m = N$ ($\mathsf{C} = \mathbf{R}^N = \mathbf{R}^2$). Working in the configuration space C instead of the workspace W, the constraints become more explicit. If the motion planner tries to develop the plan τ directly in the workspace W, it would

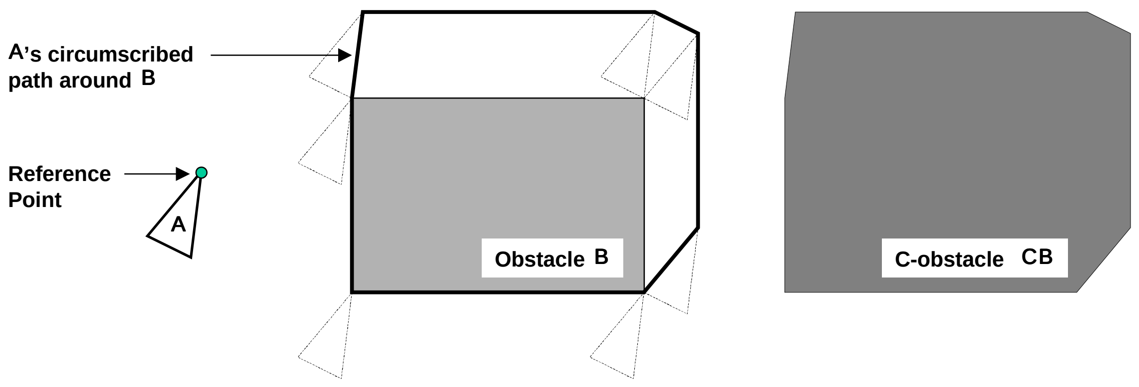


Figure 4.1: Mapping an obstacle to a C-obstacle.

have to perform operations such as collision-checking at each proposed path position whereas in C, collision-checking has already been addressed for all possible robot positions.

As A maps or "shrinks" to a point object, an obstacle B_i maps to the C-obstacle CB_i by "growing" dependent on the geometric parameters of A and B_i. The basic algorithm consists of establishing a reference point with respect to the robot A and tracing A around the obstacle B_i. The path circumscribed by A describes the C-obstacle CB_i. If A can freely rotate, the shape of CB_i depends on A's orientation, so again, if W = $\mathbf{R}^2$, then C = $\mathbf{R}^{2+1}$ = $\mathbf{R}^3$. Figure 4.1 illustrates the transformation of an obstacle to a C-obstacle.

With the generation of the configuration space C, the motion planner has transformed the path-planning problem into a point robot moving within C. Now, the motion planner must guide the robot from the initial point to the goal point through C. Latombe notes that using some type of potential field is the most successful method for guiding the robot

A [37]. The generated potential field guides A by forcing it down the gradient from the initial point to the goal point. The motion planner discretizes C by throwing a grid over the space and generates the potential field values for each grid cell.

Since the motion planner knows the geometric parameters of A, the B_is, and W *a priori*, it can generate potential fields free of local minima. The accessible route analysis in this thesis uses two potential-field-generating algorithms known as NF1 and NF2 that are free of local minima [37]. NF1 creates a potential field that guides the robot A from the initial point to the goal point on a path τ that grazes the C-obstacles. NF2 guides A on a path τ that maximizes its distance from the C-obstacles. See [37] for complete descriptions of the NF1 and NF2 algorithms.

4.1.2 Determining the $\mathsf{R}_{\mathsf{open}}$ Components

The $\mathsf{R}_{\mathsf{open}}$ node of an accessible route graph consists of three clearance components: the clearance of the opening and clearances on either side of the opening. For the opening, the $\mathsf{R}_{\mathsf{open}}$ analysis applies a geometric test with the parameters of the required clearance box taken directly from the following provision of [1]:

4.13.5 Clear Width. Doorways shall have a minimum clear opening of 32 in (815 mm) with the door open 90 degrees, measured between the face of the door and the opposite stop (see Fig. 24(a), (b), (c), and (d)). Openings more than 24 in (610 mm) in depth shall comply with 4.2.1 and 4.3.3 (see Fig. 24(e)).

EXCEPTION: Doors not requiring full user passage, such as shallow closets, may have the clear opening reduced to 20 in (510 mm) minimum.

If the opening passes the clearance box geometry interference test, the accessible route analysis continues with the $\mathsf{R}_{\mathsf{open}}$ analysis. If the opening fails this test, the opening does not qualify as a potential $\mathsf{R}_{\mathsf{open}}$ component.

The clearance box geometries on opposite sides of an opening depend on the characteristic of the door filling the opening. Figure 4.2 summarizes the dimensions of the approach clearance boxes. For a single swinging door, the ADAAG defines the side from which the user pulls the door to open it as the *pull side* and the side from which the user pushes the door to open it as the *push side*. From each side, the user can approach the opening from the *front*, *hinge side*, or *latch side* of the door:

- For the front pull side approach, the clearance box extends 60 inches from the wall that contains the opening and the door and covers the width of the opening plus 18 inches on the latch side of the door (Figure 4.2 (a), left side).

- For the front push side approach, the clearance box extends 48 inches from the wall and covers the width of the opening plus 12 inches on the latch side if the door has a closer and a latch (Figure 4.2 (a), right side).

- For the hinge pull side approach, the clearance box extends 60 inches from the wall and covers the width of the opening plus 36 inches on the latch side. Or the clearance box extends at least 54 inches from the wall and covers the width of the opening plus 42 inches on the latch side (Figure 4.2 (b), left side).

- For the hinge push side approach, the clearance box extends 42 inches from the wall (48 inches if the door has a latch and closer) and covers the width of 54 inches from the latch side extending toward the hinge side (Figure 4.2 (b), right side).

- For the latch pull side approach, the clearance box extends 48 inches from the wall (54 inches if the door has a latch and closer) and covers the width of the opening plus 24 inches on the latch side (Figure 4.2 (c), left side).

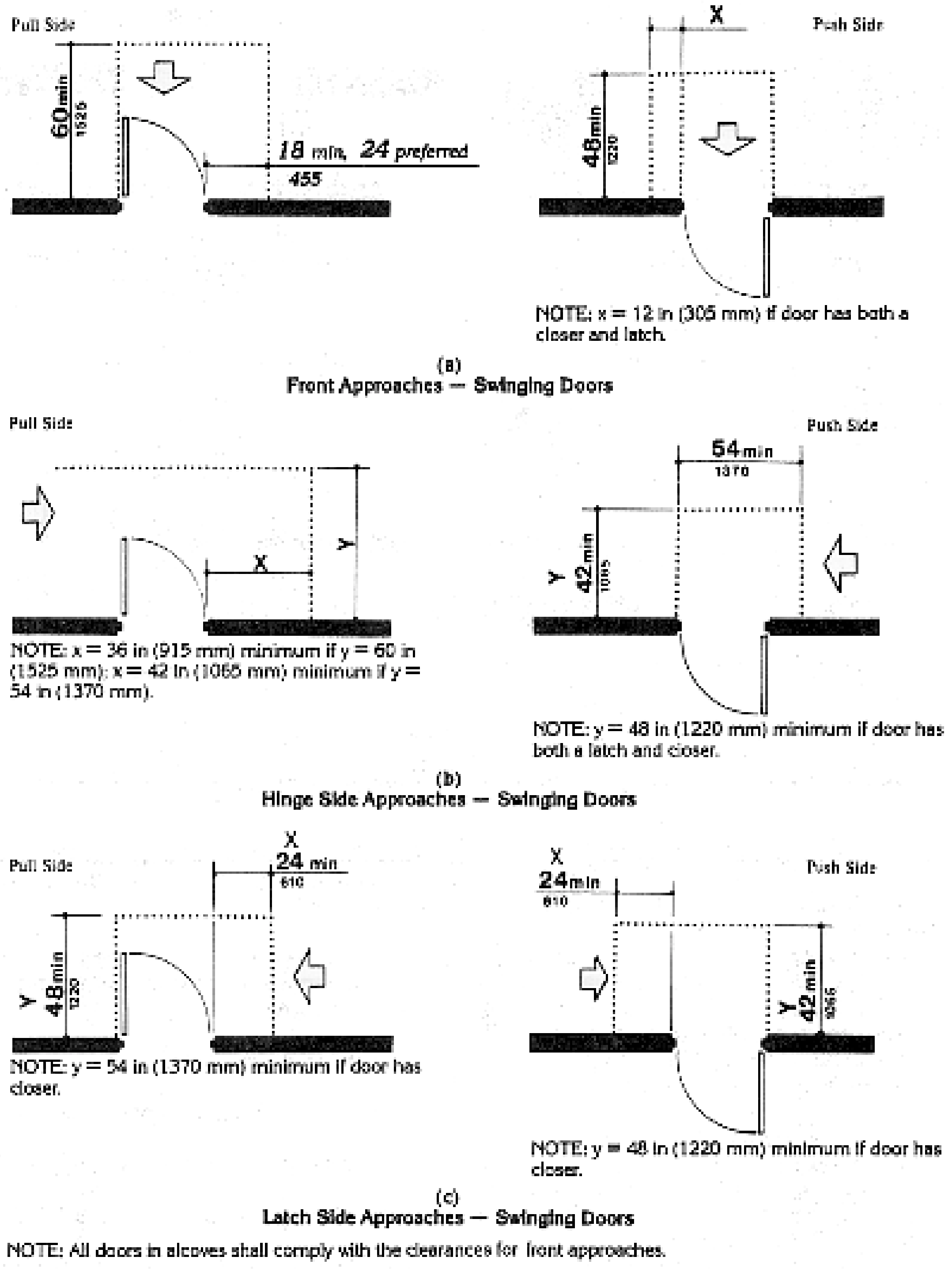

Pull Side
60 min
1525
18 min, 24 preferred
455
X
Push Side
48 min
1220
NOTE: x = 12 in (305 mm) if door has both a closer and latch.
(a)
Front Approaches — Swinging Doors
Pull Side
X
Y
NOTE: x = 36 in (915 mm) minimum if y = 60 in (1525 mm); x = 42 in (1065 mm) minimum if y = 54 in (1370 mm).
Push Side
54 min
1370
Y 42 min
1065
NOTE: y = 48 in (1220 mm) minimum if door has both a latch and closer.
(b)
Hinge Side Approaches — Swinging Doors
Pull Side
X
24 min
610
Y 48 min
1220
NOTE: y = 54 in (1370 mm) minimum if door has closer.
X
24 min
610
Push Side
Y 42 min
1065
NOTE: y = 48 in (1220 mm) minimum if door has closer.
(c)
Latch Side Approaches — Swinging Doors
NOTE: All doors in alcoves shall comply with the clearances for front approaches.

Figure 4.2: Door approaches and clearances, from the ADAAG [1].

- For the latch push side approach, the clearance box extends 42 inches from the wall (48 inches if the door has a latch and closer) and covers the width of the opening plus 24 inches on the latch side (Figure 4.2 (c), right side).

The accessible route analysis examines all possible approaches by performing geometry interference tests on the associated clearance boxes. Failure of all interference tests for either the pull or push side disqualifies the potential $\mathsf{R}_{\mathsf{open}}$ component. Conversely, if at least one clearance box on either side passes the interference test, the potential $\mathsf{R}_{\mathsf{open}}$ component qualifies as a node in the accessible route graph.

Finally, since the potential $\mathsf{R}_{\mathsf{open}}$ component participates in the accessible route graph as a node (both as an initial and goal point), the accessible route analysis establishes the initial/goal point for each valid approach clearance box associated with the potential $\mathsf{R}_{\mathsf{open}}$ component. Each approach clearance box has an initial/goal line segment generalized from the basic motion-planning initial/goal point formulation. The motion planner allows the wheelchair to start anywhere along the initial line. Similarly, the motion planner has successfully found a path to an $\mathsf{R}_{\mathsf{open}}$ component if the robot can get to any position along the goal line segment:

- For the front approach, the accessible route analysis defines the initial/goal line segment as the front edge of the clearance box.

- For either side approach, the accessible route analysis defines the initial/goal line segment as the applicable side edge (the edge that is penetrated by the user during the approach) of the clearance box.

Figure 4.3 illustrates the initial/goal line segments for the clearance boxes shown in Figure 4.2.

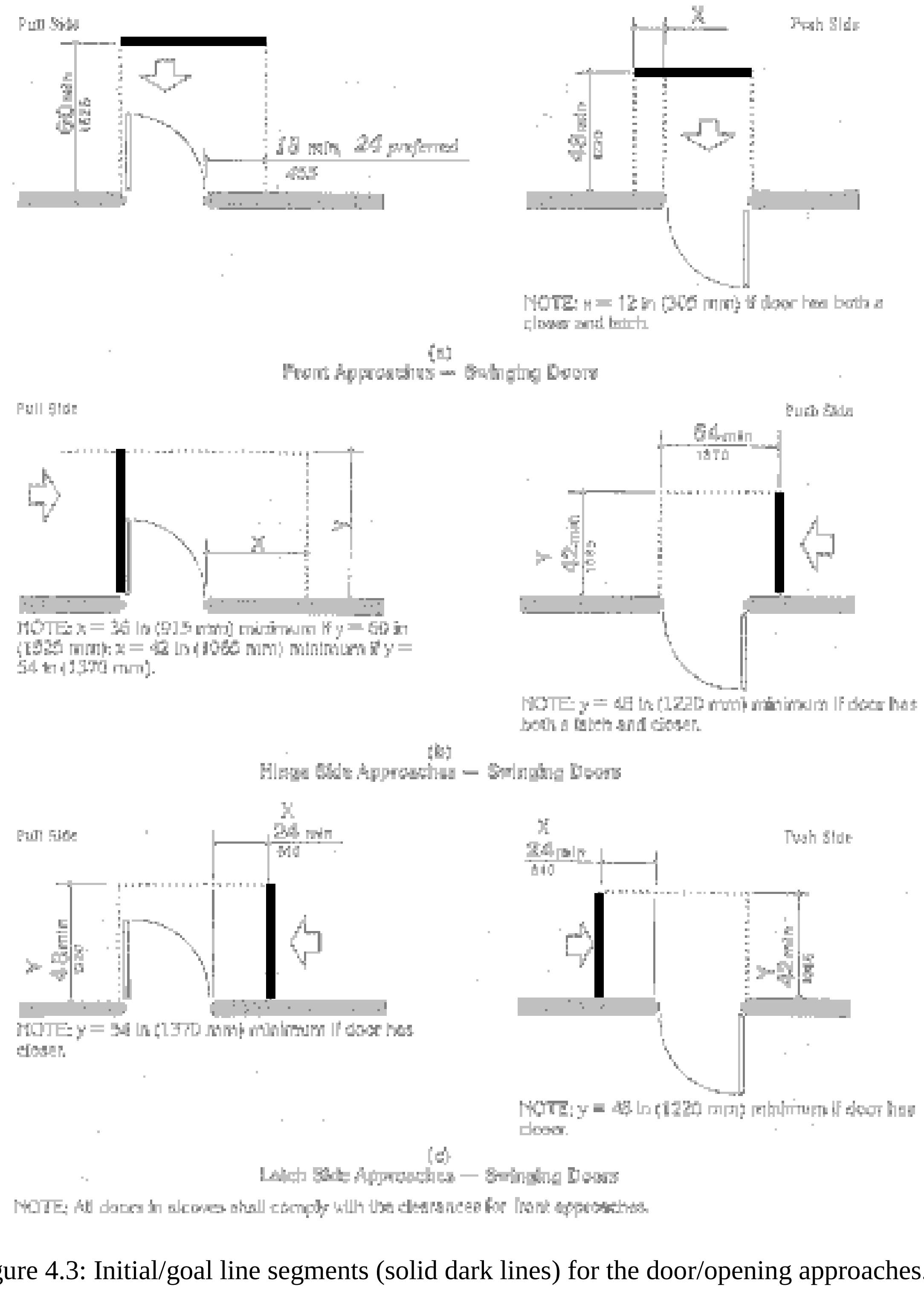

Pull Side
60 min
1525
18 min, 24 preferred
455
x
Push Side
48 min
1220
NOTE: x = 12 in (305 mm) if door has both a closer and latch.
(a)
Front Approaches — Swinging Doors
Pull Side
x
y
NOTE: x = 36 in (915 mm) minimum if y = 60 in (1525 mm); x = 42 in (1065 mm) minimum if y = 54 in (1370 mm).
Push Side
54 min
1370
y
42 min
1065
NOTE: y = 48 in (1220 mm) minimum if door has both a latch and closer.
(b)
Hinge Side Approaches — Swinging Doors
Pull Side
x
24 min
610
y
48 min
1220
NOTE: y = 54 in (1370 mm) minimum if door has closer.
Push Side
x
24 min
610
y
42 min
1065
NOTE: y = 48 in (1220 mm) minimum if door has closer.
(c)
Latch Side Approaches — Swinging Doors
NOTE: All doors in alcoves shall comply with the clearances for front approaches.

Figure 4.3: Initial/goal line segments (solid dark lines) for the door/opening approaches.

4.1.3 Determining the $\mathtt{R}_{\mathtt{init}}$ and $\mathtt{R}_{\mathtt{goal}}$ Components

With the exception of site-level design analysis (as opposed to building-, story-, and space-level design analysis), the accessible route analysis assigns the $\mathtt{R}_{\mathtt{init}}$ node or nodes to a doorway or doorways in the facility design. Since the accessible route analysis has already tested all openings when looking for potential $\mathtt{R}_{\mathtt{open}}$ components, the analysis is not required to perform the geometric interference tests again. If none of the potential $\mathtt{R}_{\mathtt{open}}$ components are valid $\mathtt{R}_{\mathtt{init}}$ nodes, the facility design contains no accessible routes connecting the facility's building components to the entry points of the facility.

The accessible route analysis tests all other relevant building components for potential $\mathtt{R}_{\mathtt{goal}}$ components (recall that the first-order analysis uses a brute-force approach) since, initially, the analysis does not know which building components are relevant to the accessibility of the facility. All building components have at least one associated clearance box, and the number of clearance boxes associated with a building component depends on the number of user approaches it has. As with the $\mathtt{R}_{\mathtt{open}}$ testing, the analysis uses each clearance box (or boxes) in a geometric interference test. For water closet approach, the clearance box parameters are prescribed by [1] as follows:

Figure 28. For a front transfer to the water closet, the minimum clear floor space at the water closet is a minimum 48 inches (1220 mm) in width by a minimum of 66 inches (1675 mm) in length. For a diagonal transfer to the water closet, the minimum clear floor space is a minimum of 48 inches (1220 mm) in width by a minimum of 56 inches (1420 mm) in length. For a side transfer to the water closet, the minimum clear floor space is a minimum of 60 inches (1525 mm) in width by a minimum of 56 inches (1420 mm) in length. (4.16.2, A4.22.3)

Figure 4.4 illustrates the clearance box parameters for the three water closet transfer options.

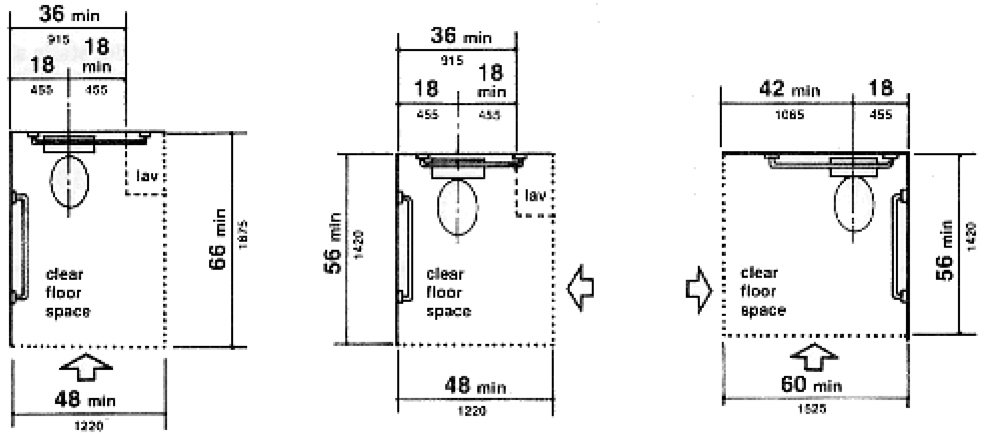


Figure 4.4: Clear floor space for front transfer, diagonal transfer, and side transfer [1].

Finally, since the potential $\mathsf{R}_{\mathtt{goal}}$ component is an end node in the accessible route, the accessible route analysis establishes the goal point for each valid approach clearance box associated with the potential $\mathsf{R}_{\mathtt{goal}}$ component. Similar to the $\mathsf{R}_{\mathtt{open}}$ component each approach clearance box has a goal line segment as opposed to a point. For example, for water closets, the following approach definitions apply:

- For the front approach, the accessible route analysis defines the goal line segment as the front edge of the clearance box.
- For either side approach, the accessible route analysis defines the goal line segment as the applicable side edge (the edge that is penetrated by the user during the approach) of the clearance box.
- For the diagonal approach, the accessible route analysis defines two goal line segments combining the front approach with the side approach formulation.

Figure 4.5 illustrates the initial/goal line segments for the clearance boxes shown in Figure 4.4.

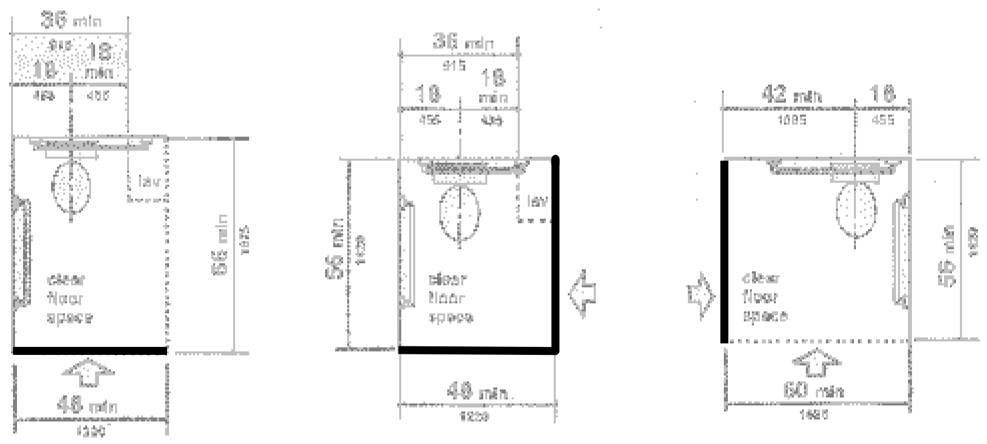


Figure 4.5: Goal line segments (solid dark lines) for the water closet approaches.

4.1.4 Determining R_{seg} Accessible Route Components

The accessible route analysis can now determine if an accessible path exists between all pairs of nodes within a SPACE container. The following provision from the ADAAG describes the critical path-determining parameters:

4.3.3 Width. The minimum clear width of an accessible route shall be 36 in (915 mm) except at doors (see 4.13.5 and 4.13.6). If a person in a wheelchair must make a turn around an obstruction, the minimum clear width of the accessible route shall be as shown in Fig. 7(a) and (b).

Note that the second sentence in the provision represents two exceptions to the prescribed rule in the first sentence (Figure 4.6 illustrate Figure 7(a) and Figure 7(b) from [1]). Furthermore, the two exceptions only address two of possible turn-around-an-obstacle examples. This limitation illustrates a deficiency in the prescriptive-based accessible route as there are many other turn-around-an-obstacle scenarios. The diagrams in Figure 4.6 illustrate only orthogonal paths, and there are non-orthogonal 36-inch wide paths that

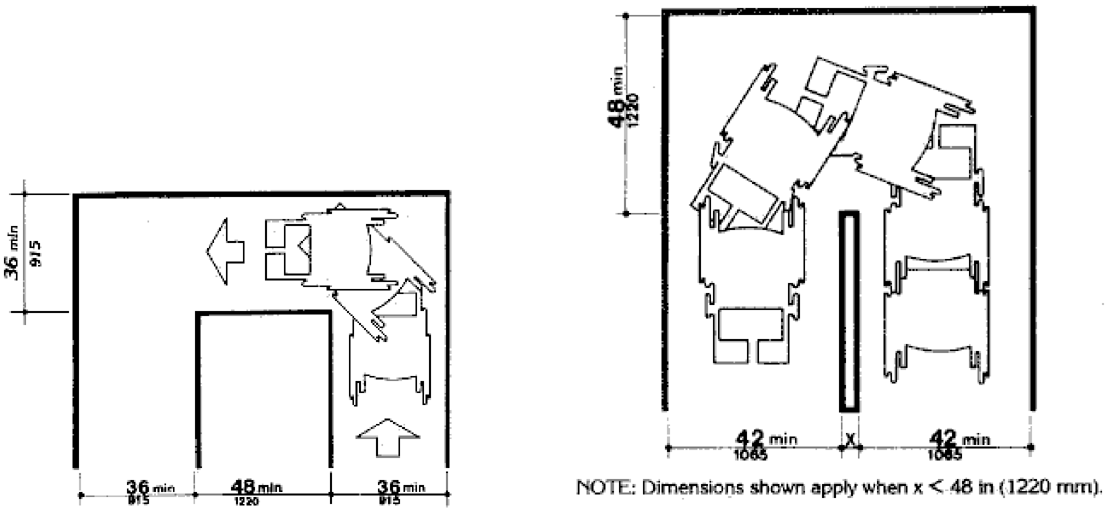


Figure 7(a)
Accessible Route
90 Degree Turn

A 90 degree turn can be made from a 36 inch (915 mm) wide passage into another 36 inch (915 mm) passage if the depth of each leg is a minimum of 48 inches (1220 mm) on the inside dimensions of the turn.

Figure 7(b)
Accessible Route
Turns around an Obstruction

A U-turn around an obstruction less than 48 inches (1220 mm) wide may be made if the passage width is a minimum of 42 inches (1065 mm) and the base of the U-turn space is a minimum of 48 inches (1220 mm) wide.

Figure 4.6: Minimum accessible route turning clearances defined in the ADAAG [1].

need some minimum distance for the legs of the path. For example, the ADAAG does not address a leg length requirement for a 95-degree turn.

4.1.4.1 The 36-inch-wide Path

From the first sentence of Provision 4.3.3 from the ADAAG, the motion planner uses a 36-inch disc to describe the geometry of the robot A_{36} (the implementation uses a regular dodecagon to approximate the circle geometry). The building component geometry determines the workspace W— the motion planner generates the configuration space C_{36} given A_{36} and W using the algorithm described in Section 4.1.1. Figure 4.7 illustrates

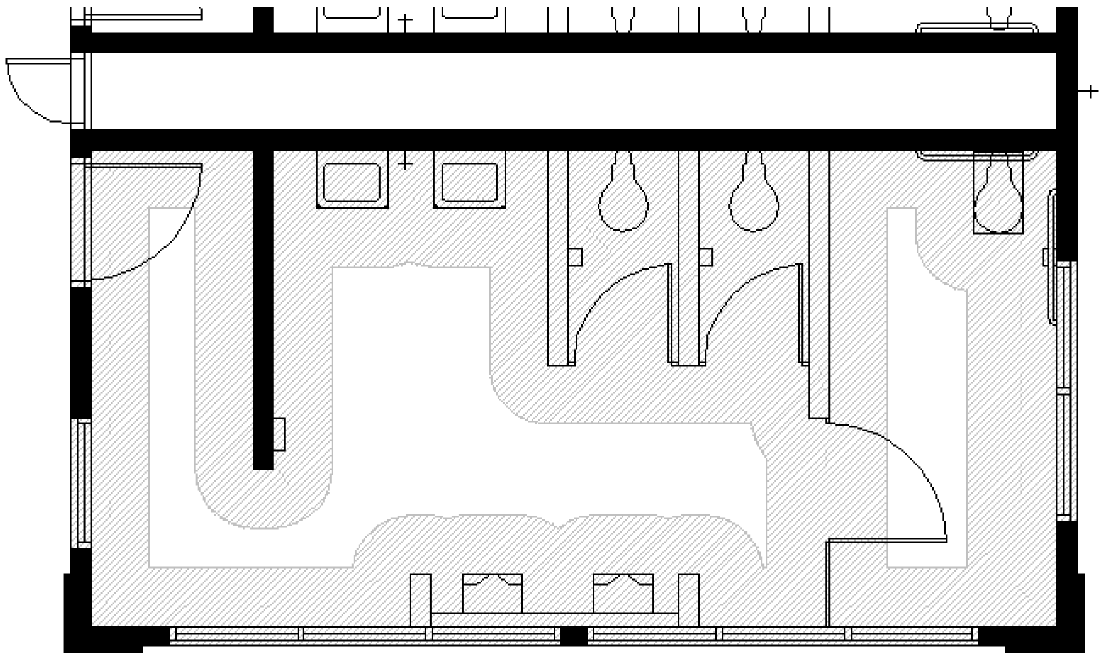


Figure 4.7: The 36-inch disc robot A_{36} configuration space C_{36}.

C_{36} for the Men's Bathroom space in the bathroom facility example. The white areas represent legal positions for centerpoint the A_{36} disc robot.

Note that the motion planner treats a doorway with the door in the closed position, and, hence, in Figure 4.7, the configuration space between the entry door and the accessible toilet is discontinuous. However, as shown in Figure 4.3, the wheelchair robot only needs to reach the goal line segment associated with the doorway and does not have to pass through the opening. As long as the opening complies with ADAAG clearance requirements, the accessible route continues on the other side of the doorway at that opening's complementary initial line segment. Figure 4.8 illustrates possible accessible route sequences through a doorway with the dashed lines indicating the path of travel

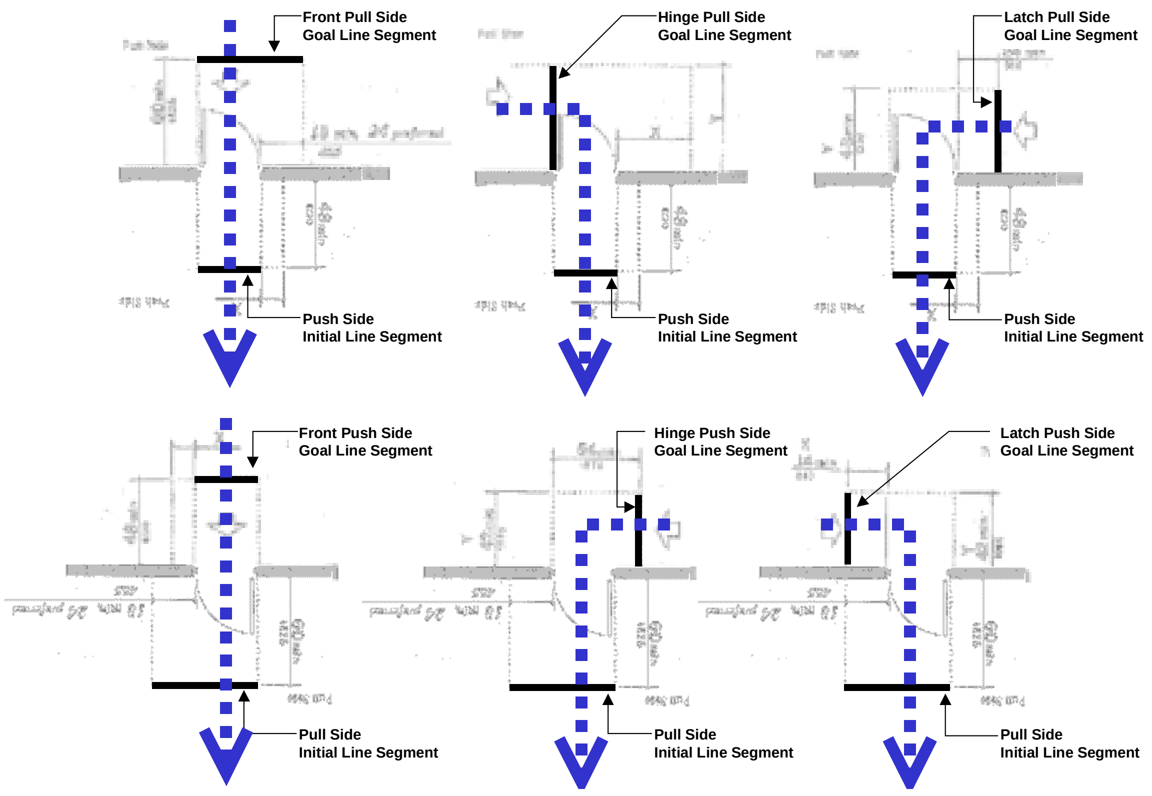


Figure 4.8: The possible accessible route sequences through a doorway.

from the goal line segment of an accessible route to the door to the initial line segment of an accessible route from the door.

The motion planner uses the NF1 potential-field-generating algorithm between pairs of nodes [37]. This algorithm generates a potential map “wavefront” from the goal line segment(s) to the initial line segment(s). The wavefront generation terminates either at some point on the initial segment(s) or when the motion planner has run out of configuration space to generate the wavefronts. If the wavefront has not reached an initial line segment, then no path exists between the initial and goal nodes. Conversely, if the wavefront has reached the initial point, then a 36-inch path exists between nodes.

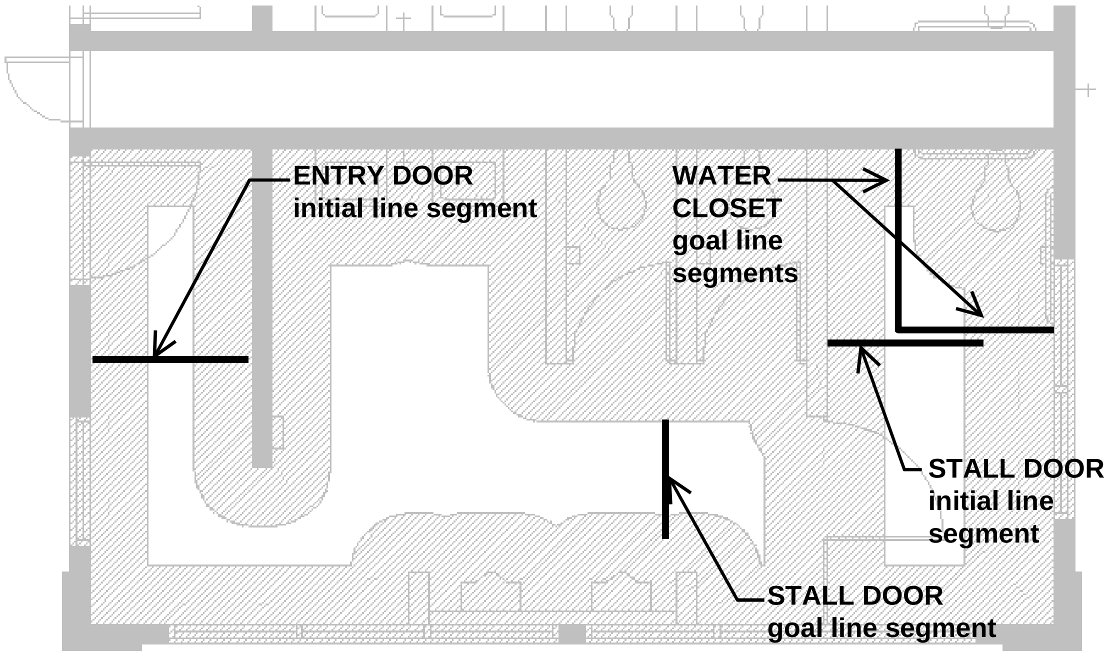

ENTRY DOOR
initial line segment
WATER CLOSET goal line segments
STALL DOOR initial line segment
STALL DOOR goal line segment

Figure 4.9: The initial and goal segments inserted into the $\mathbf{C}_{36}$ configuration space.

Figure 4.9 illustrates an initial line segment for the `ENTRY DOOR`, the initial and goal line segments for the disabled access `STALL DOOR`, and the goal line segments associated with the disabled access `WATER CLOSET`. The NF1 algorithm finds a path between the `ENTRY DOOR` initial line segment and the `STALL DOOR` goal line segment, but it fails to find a path between the `ENTRY DOOR` initial line segment and the `WATER CLOSET` goal line segments. (Upon visual inspection, there is a continuous white area between legal positions on the two `DOOR` line segments, but the white area is discontinuous between legal positions of `ENTRY DOOR` line segment and the `WATER CLOSET` line segments). However, the path between the `ENTRY DOOR` and disabled access `WATER CLOSET` actually consists of two path segments:

1. The path between the `ENTRY DOOR` initial line segment and the `STALL DOOR` goal line segment.

2. The path between the `STALL DOOR` initial line segment and one of the `WATER CLOSET` goal line segments.

The analysis connects these two paths using an accessible path sequence through a doorway similar to the one shown in Figure 4.8.

4.1.4.2 The Turn-around-an-obstruction Exceptions

This section describes a proposed method to handle the turn-around-an-obstruction exceptions that arise from the prescriptive-based accessible route in the ADAAG. Note that the complicated nature of this exception handling proposal can be traced to the nature of these prescriptive exceptions. The turn-around-an-obstruction exceptions are arbitrary configurations that the ADAAG addresses as being impassable using the general 36-inch wide path, and this section shows the difficulty in taking these arbitrary configurations into consideration. While developing the exception analysis methods, the investigation of these exceptions in this section shows that the exceptions contradict each other.

The analysis in Section 4.1.4.1 does not actually need to generate a path $\boldsymbol{\tau}$ between pairs of nodes; it simply determines if a 36-inch path exists. Now, however, the motion planner must examine the wheelchair path's signature. The motion planner generates an NF2 potential map in the configuration space to generate a path $\boldsymbol{\tau}$ that maximizes its distance from the obstacles [37]. Note that the proposed exception analyses indicate possible violations to the exceptions as opposed to making the actual determination that a configuration actually violates the exceptions.

The motion planner also constructs two additional configuration spaces to analyze the facility design for the first turn-around-an-obstruction exception shown in Figure 4.6:

- A configuration space $\mathsf{C}_{36+\varepsilon}$ using the same workspace W but a (36+ε)-inch disc robot $\mathsf{A}_{36+\varepsilon}$ to generate the C-obstacles in the configuration space.

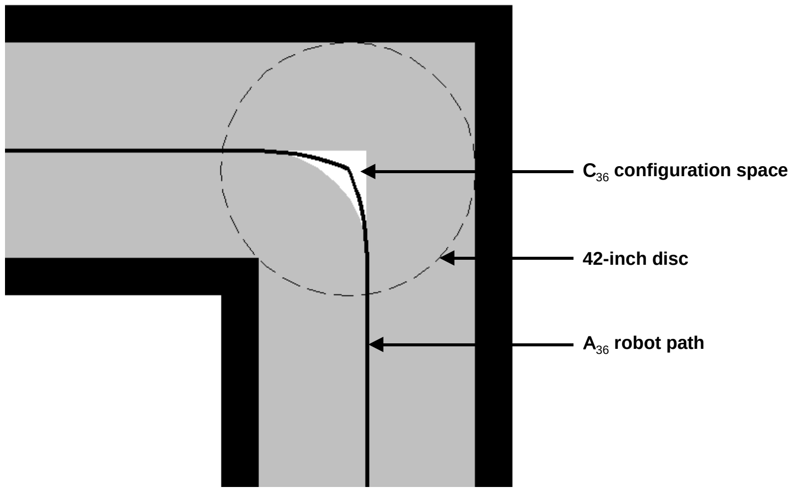


Figure 4.10: A 36-inch corridor with the path and legal $\mathbf{C}_{36}$ and positions.

- A configuration space $\mathbf{C}_{42+\varepsilon}$ using the same workspace $\mathbf{W}$ but a (42+ε)-inch disc robot $\mathbf{A}_{42+\varepsilon}$ to generate the C-obstacles in the configuration space.

The first exception applies only to a path in two perpendicular 36-inch corridors, so the motion planner must be alerted when the $\mathbf{A}_{36}$ robot is following a path through such a corridor. To determine if the $\mathbf{A}_{36}$ robot is in such a corridor, the motion planner uses the $\mathbf{C}_{36+\varepsilon}$ configuration space. As the motion planner steps through the path τ, it also checks to see if the robot's position is legal in $\mathbf{C}_{36+\varepsilon}$. If it is not, then the robot must be in an area that is only 36 inches wide since the NF2-generated path τ stays as far from obstacles as possible.

Once the motion planner has determined that the $\mathbf{A}_{36}$ robot is in a 36-inch-wide area, the motion planner needs to know if path τ resides in a perpendicular 36-inch corridor configuration. Figure 4.10 illustrates a 36-inch corridor with the $\mathbf{A}_{36}$ robot path τ, legal positions in C_{36} (the robot can travel in the white space and along τ). The largest legal area in the 36-inch perpendicular corridor configuration is a 42-inch wide disc as illustrated by the dashed circle in Figure 4.10. Therefore, if τ traverses a legal position in the $\mathsf{C}_{42+\varepsilon}$ configuration space, the motion planner knows the A_{36} robot is *not* in a perpendicular 36-inch wide corridor configuration.

When the robot enters a 36-inch wide area, the motion planner begins storing the path positions (including orientations) in a queue. Since A_{36} is a disc and has no orientation, the motion planner approximates the current orientation θ_C at the current path position $(x,y)_C$ by calculating the angle formed between $(x,y)_{C-1}$ and $(x,y)_{C+1}$. If at any point the path τ enters the $\mathsf{C}_{42+\varepsilon}$ configuration space (implying that the A_{36} robot is not in the perpendicular 36-inch corridor configuration), the exception analysis terminates, and the motion planner starts the exception analysis again when τ encounters the next 36-inch wide area.

The motion planner compares $(x,y,\theta)_C$, to each position $(x,y,\theta)_i$ already in the queue. If θ_C equals some θ_i, the motion planner inserts $(x,y,\theta)_C$ at the front of the queue and removes the $(x,y,\theta)_i$ and all the positions before$(x,y,\theta)_i$ in the queue. θ_C equaling some $(\theta_i, + \pi/2)$ indicates that the A_{36} robot has made a right-angle turn. At this point, the motion planner indicates a possible violation of the first exception and for the path segment between $(x,y,\theta)_C$, and $(x,y,\theta)_i$.

The second exception addresses a U-turn around an obstacle shown in Figure 4.6. From the beginning of the analysis of path τ, the motion planner inserts the positions of τ into a queue and compares the current position $(x,y,\theta)_C$, to each position $(x,y,\theta)_i$ already in the queue. θ_C equaling some $(\theta_i, + \pi)$ indicates that the A_{36} robot has made a U-turn, and the motion planner evaluates the perpendicular distance between $(x,y,\theta)_C$, and the $(x,y,\theta)_i$. If the perpendicular distance is less than 84 inches (from the left diagram in Figure 4.6, the

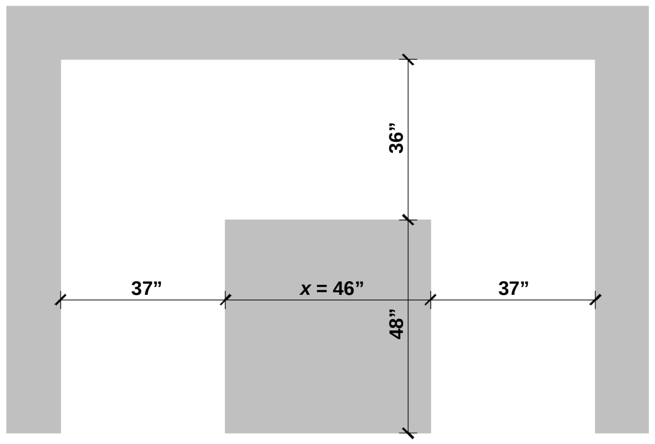


Figure 4.11: A contradictory configuration to the Provision 4.3.3 exceptions from the ADAAG.

second leg of the path, 48 inches, plus half the width of the first and third legs of the path, 18 inches and 18 inches), then the motion planner indicates a possible violation for the second exception for the path segment between $(x,y,\theta)_C$, and $(x,y,\theta)_i$.

Note that the above analysis does not exactly match the prescribed parameters of the second exception because there is a fundamental flaw between the relationship of the first exception and the second exception. A graphical example illustrates this flaw: Figure 4.11 illustrates a configuration that actually has *more* maneuvering space than a configuration that complies with the prescribed parameters of the first exception, yet this example configuration violates the prescribed parameters of the second exception. In Figure 4.11, the obstruction width x is less than 48 inches (x = 46 inches) implying that the second exception applies. However the first and third legs of the path are less than 42 inches wide and the second leg is less than 48 inches wide. Thus, the configuration violates the second exception. This contradiction illustrates the conflicts that can arise with the prescribed parameters of a building code’s intent.

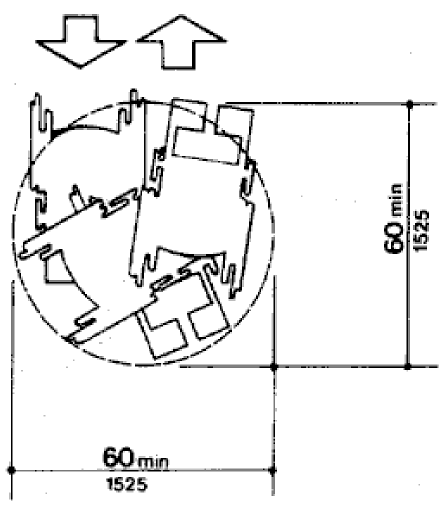


Figure 3a
Wheelchair Turning Space
60-in (1525 mm) Diameter Space

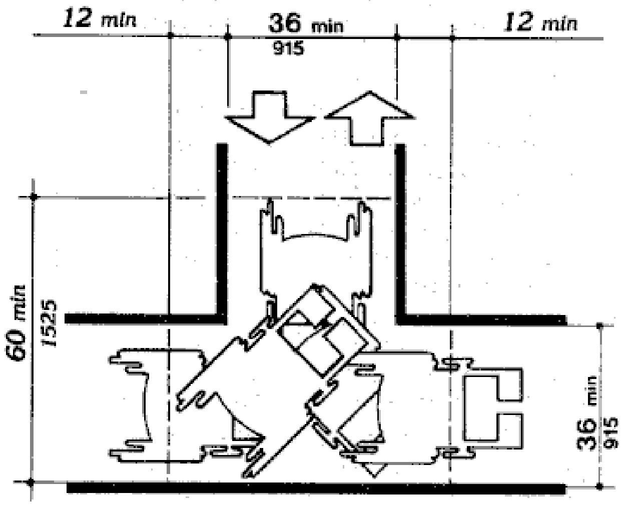


Figure 3b
Wheelchair Turning Space
T-Shaped Space for 180 Degree Turns

Figure 4.12: The prescribed turning circle and T-space from the ADAAG [1].

4.1.4.3 The Wheelchair Turning Circle

The ADAAG requires a wheelchair turning circle or T-space in toilet rooms, bathrooms, bathing facilities, and shower rooms, dressing and fitting rooms:

4.2.3 Wheelchair Turning Space. The space required for a wheelchair to make a 180-degree turn is a clear space of 60 in (1525 mm) diameter (see Fig. 3(a)) or a T-shaped space (see Fig. 3(b)).

Figure 4.12 illustrates the turning circle and T-space from the ADAAG.

The motion planner analyzes a relevant SPACE container for the wheelchair turning circle option by generating a configuration space $\mathbf{C}_{60}$ using a 60-inch disc robot $\mathbf{A}_{60}$ that represents the turning circle. If the SPACE does not contain any legal positions for robot $\mathbf{A}_{60}$, no turning circle exists in the SPACE, and the facility does not comply with [1]. If legal positions exist, the analysis continues. Figure 4.13 illustrates the $\mathbf{C}_{60}$ configuration

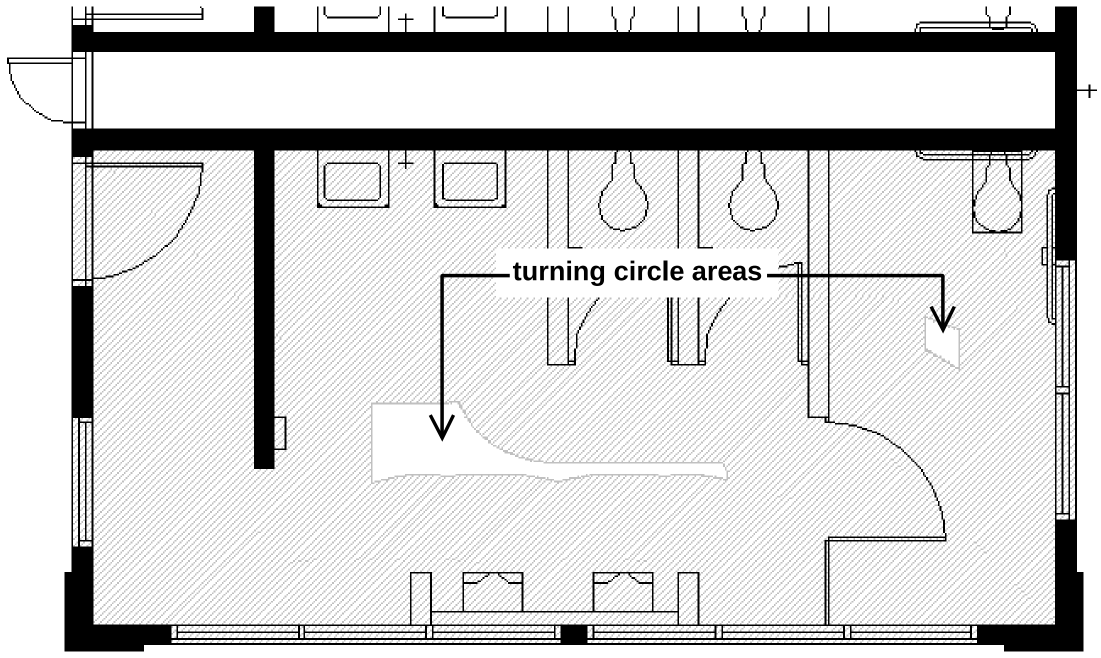


Figure 4.13: The 60-inch disc robot A_{60} configuration space C_{60}.

space for the bathroom facility, and the white spaces indicate legal positions for the 60-inch turning circle.

Now, the motion planner tries to generate a path from the $\mathsf{R}_{\mathsf{open}}$ nodes that exist on the boundary of the given `SPACE` container (in this bathroom facility case, the `ENTRY DOOR`). The motion planner evaluates each of these $\mathsf{R}_{\mathsf{open}}$ nodes separately. Each $\mathsf{R}_{\mathsf{open}}$ node becomes the initial point (initial line segment), and the turning circle areas become the goal points (goal areas), and the motion planner generates the NF1 wavefront from the turning circle areas. Figure 4.14 illustrates the turning circle areas from C_{60} configuration space that now become the goal points (goal areas) in the C_{36} configuration space, and the `ENTRY DOOR` line segment that becomes the initial point (initial line segment).

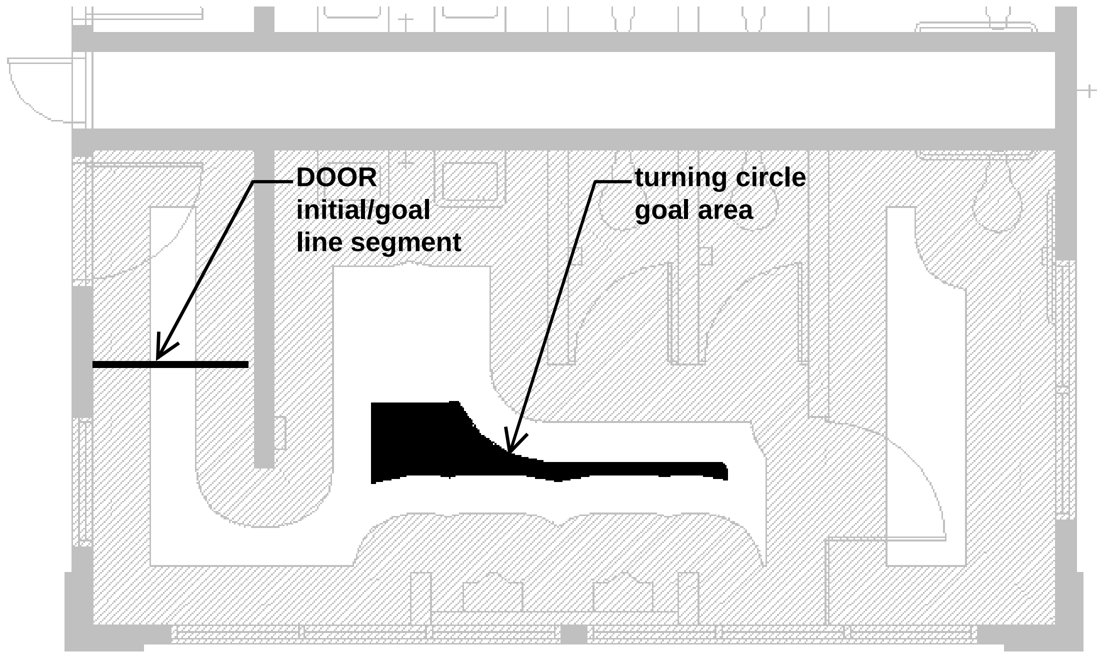


Figure 4.14: The entrance door initial/goal and the 36- and 60-inch configuration spaces.

The motion planner now generates the NF1 wavefront from the turning circle areas, and if the wavefront terminates at the given $\mathtt{R}_{\mathtt{open}}$ node (the initial line segment), the motion planner has proven that an accessible wheelchair turning circle exists in the given `SPACE` container. In the bathroom facility case, the motion planner successfully terminates the wavefront at a point on the `ENTRY DOOR` line segment. (Upon visual inspection, a continuous white space exists between the `ENTRY DOOR` line segment and the turning circle area).

4.1.5 Prescriptive-based Analysis Discussion

The developed prescriptive-based motion-planning techniques described in this chapter verify the 36-inch width and turning circle requirements as prescribed by the ADAAG. However, using motion-planning techniques to test for the turn-around-an-obstruction exceptions is a more complicated process since these exceptions are abstractions that are static geometric tests as opposed to defining acceptable wheelchair motion.

In addition, as has been shown, these abstractions which define the turn-around-an-obstruction exceptions only address two of the possible problem configurations given the prescribed 36-inch width. However, these two prescribed configurations contradict one another, and the contradiction adds to the complication of modeling the exceptions for automated analysis.

The motion-planning techniques guarantee the discovery of a 36-inch accessible route if one exists provided that the motion planner generates configuration space grid discretization at least equal to the precision of the facility design measurements. For example, if the facility is designed to the nearest inch, the grid discretization should be at least as fine as one inch.

4.2 Automated Performance-based Analysis: The Usable Accessible Route

The performance-based accessible route analysis presented in this section attempts to address the deficiencies of the prescribed accessible route parameters of the ADAAG. Difficulties in capturing intent of a standard and the behavior of the dependent components are major issues in developing performance-based methods. This section demonstrates that determining the accessible route belongs to a family of problems that can be successfully modeled using performance-based methods.

The performance-based accessible route analysis uses motion-planning simulation to generate the accessible route graph. In addition to a variation of the motion planner developed for the prescriptive-based analysis that addresses the wheelchair user’s comfort level, the motion-planning parameters developed in this section directly capture wheelchair behavior (the motion planner developed for the prescriptive-based approach captures the ADAAG's prescribed parameters). The interaction of the wheels of a car-like robot such as a wheelchair and the ground surface constrains robot motion in a manner dependent on its instantaneous orientation. This non-slipping or *non-holonomic* turning constraint restricts the motion of the robot in its attempt to reach the goal point. Dubins solved the shortest path in a plane for a robot-like car that cannot reverse and cusps are not allowed [10]. The wheelchair motion developed in this research is a variation on the car motion formulation, and next section describes the non-holonomic planner developed for wheelchair motion.

4.2.1 Overview of Performance-based Motion Planning: Developing a Non-holonomic Planner

The motion planner developed for the prescriptive-based analysis utilized various sizes of disc robots. The performance-based analysis also uses the A_{36} robot to generate the C_{36} configuration space and generates an NF2 potential field, and the motion planner uses this potential field as a guide to generate the path $\boldsymbol{\tau}$ for the actual wheelchair robot A_{wc}. Figure 4.15 shows the reference wheelchair dimensions from the ADAAG.

Figure 4.16 illustrates the geometry of the A_{wc} robot. Note that the robot is less than 36 inches wide, but the motion planner uses the C_{36} configuration space since this 36-inch width in the ADAAG represents a comfortable width for the wheelchair user to negotiate.

Since the A_{wc} is not a disc, the motion planner must keep track of the robot's orientation while generating a path, and the motion planner must check each wheelchair position and orientation against the obstacles in the space. Therefore, the motion planner discretizes the rotation space and creates configuration spaces $\mathsf{C}_{\mathsf{wc}0}\ldots\mathsf{C}_{\mathsf{wc}n}$ such that $2\pi/n$ equals the

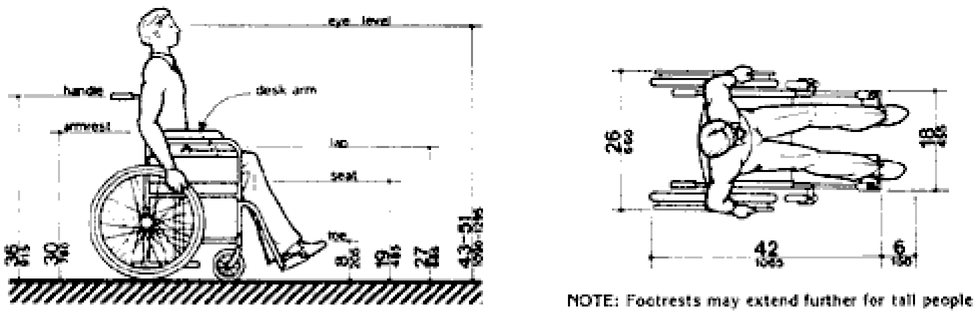


Figure 4.15: ADDAG wheelchair dimensions.

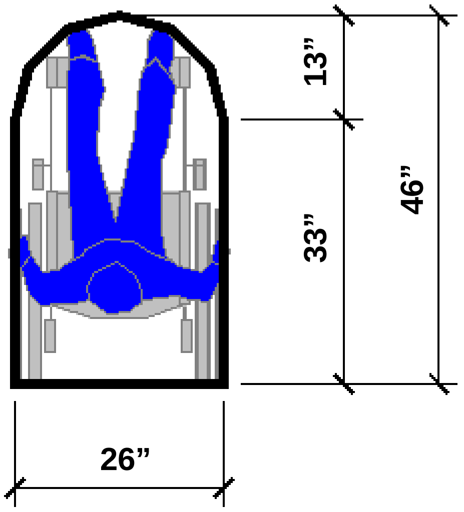


Figure 4.16: Dimensions of the robot A_{WC}.

number of degrees between sequential orientations of A_{WC}. Now, the motion planner can check the wheelchair position and orientation (x,y,θ) against the appropriate $\mathsf{C}_{\mathsf{WC}i}$

configuration space. Section 4.2.2 and Section 4.2.3 address the initial/goal point formulation and Section 4.2.4 addresses the non-holonomic path.

4.2.2 Determining the $\mathsf{R}_{\mathsf{open}}$ Components

The performance-based approach that determines the $\mathsf{R}_{\mathsf{open}}$ components uses the same $\mathsf{R}_{\mathsf{open}}$ formulation for the prescriptive-based approach. However, instead of specifying multiple clearance boxes and the associated initial and goal line segments, this $\mathsf{R}_{\mathsf{open}}$ determination uses one initial point and one goal point on the opening's pullside and a combined initial/goal point on the opening's pushside. While a clearance box is explicitly prescribed in the ADAAG to be tested as a static evaluation method, the performance-based accessible route analysis models the actual wheelchair path directly, and the wheelchair must pass through some clearance area when starting from or getting to the initial or goal point.

Figure 4.17 illustrates the positions of the initial and goal points associated with the opening. Since the motion planner uses the initial and goal points to generate the NF2 potential field in the C_{36} configuration space, the figure shows the A_{36} robot as well as the A_{wc} robot. The A_{wc} robot shown in the figure has a fixed orientation associated with the initial points. However, the motion planner accepts any orientation within a 90-degree range for the orientation of the A_{wc} robot at the goal position. Note that when passing though a door opening, the wheelchair goes from the goal point of a path segment on one side of the door opening to the initial point of a path segment on the opposite side of the door opening. The goal point-initial point sequence through a door opening is either (b)-(c) or (d)-(a) from the figures shown in Figure 4.17. This research has developed the door opening goal point and initial point parameters guaranteeing that a path exists from the goal point-initial point pair.

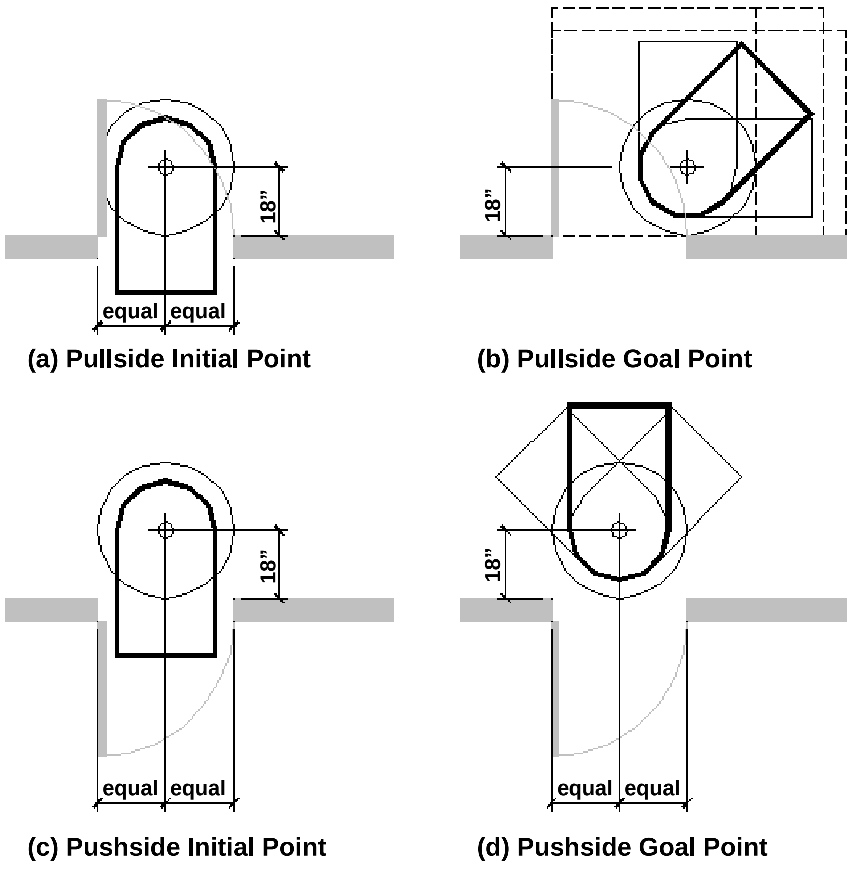


Figure 4.17: Initial and goal points for the $\mathsf{R}_{\texttt{open}}$ node.

4.2.3 Determining the $\mathsf{R}_{\texttt{init}}$ and $\mathsf{R}_{\texttt{goal}}$ Components

As with the $\mathsf{R}_{\texttt{open}}$ component, the determination of $\mathsf{R}_{\texttt{init}}$ and $\mathsf{R}_{\texttt{goal}}$ utilizes the basic motion-planning initial and goal point as opposed to an initial and goal line segment associated with the relevant clearance box as prescribed by the ADAAG. To reiterate the motivation, the ADAAG prescribes a clearance box as the only testable static method to ensure component usability, but a dynamic method should evaluate whether the preconditions and the goals can be directly satisfied.

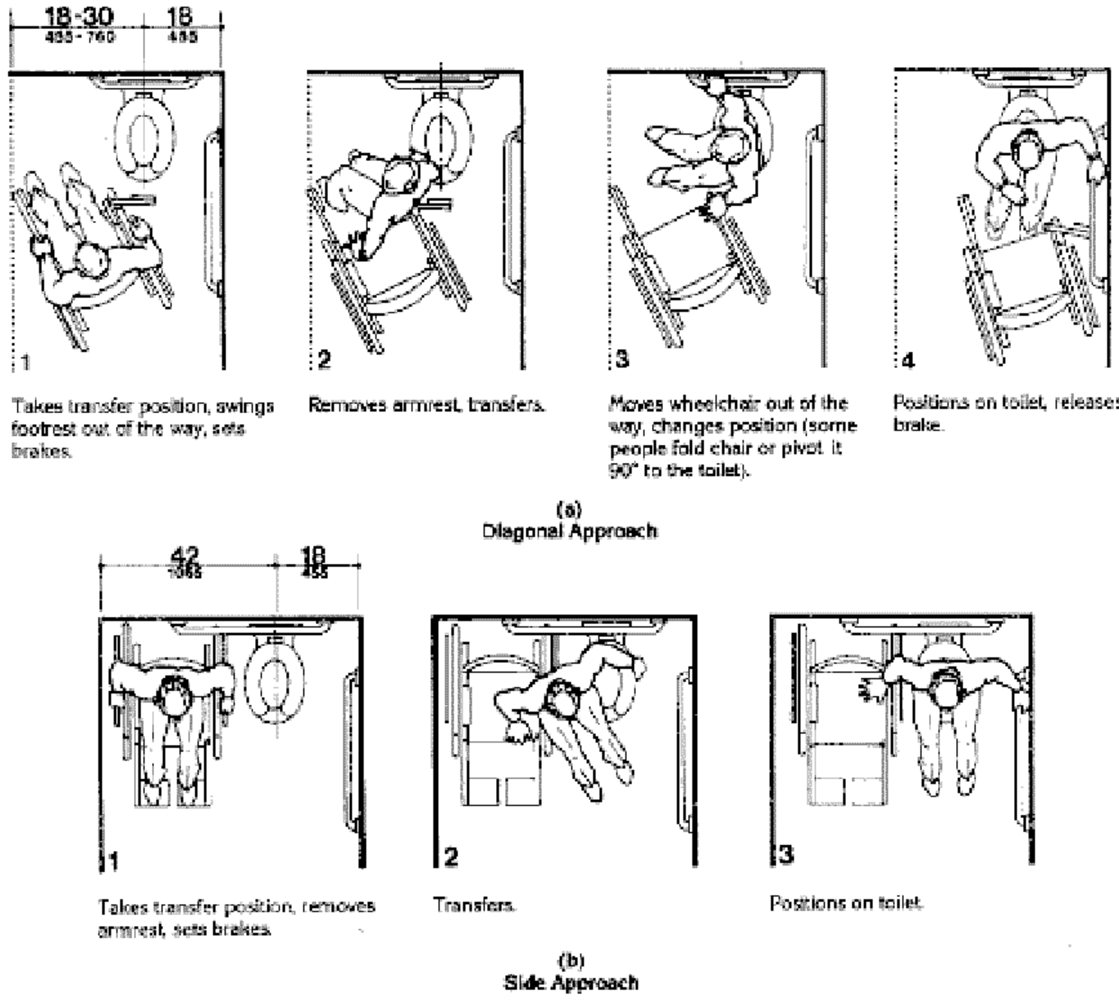

18-30
455-760
18
455
1
Takes transfer position, swings footrest out of the way, sets brakes.
2
Removes armrest, transfers.
3
Moves wheelchair out of the way, changes position (some people fold chair or pivot it 90° to the toilet).
4
Positions on toilet, releases brake.
(a)
Diagonal Approach
42
1065
18
455
1
Takes transfer position, removes armrest, sets brakes.
2
Transfers.
3
Positions on toilet.
(b)
Side Approach

Figure 4.18: ADAAG wheelchair transfer diagrams for water closets [1].

In general, an $\mathsf{R}_{\mathtt{goal}}$ node maps to one goal point. However, in certain cases, the motion planner needs more than one goal point to decide a component's accessibility. Figure 4.18 illustrates toilet usage by a wheelchair user, an action known as wheelchair transfer. As shown in the figure, the wheelchair user can transfer from the wheelchair to the toilet via two fundamentally different methods: diagonal transfer and side transfer. Thus, the motion planner specifies two different goal points and orientations to reflect the different methods.

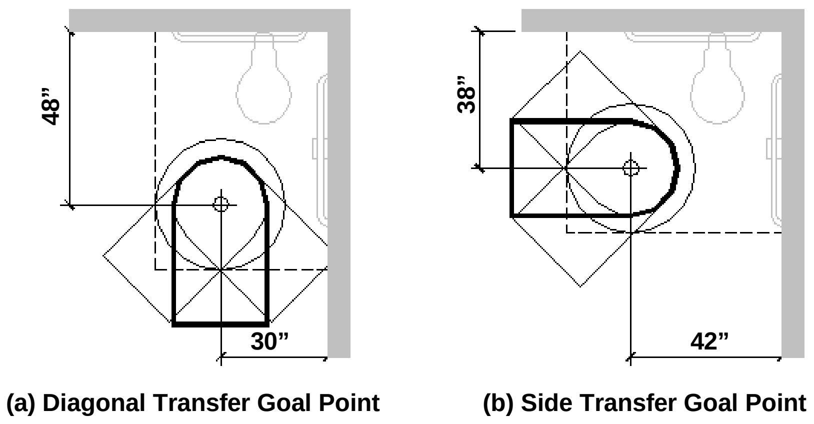

48”
30”
(a) Diagonal Transfer Goal Point
38”
42”
(b) Side Transfer Goal Point

Figure 4.19: The goal points for water closet diagonal and side transfer respectively.

Figure 4.19 illustrates the two goal points and orientations associated with the two transfer options. The side transfer goal point and orientation of the A_{wc} robot illustrated in Figure 4.19(b) does not directly correspond to the side transfer position illustrated in Figure 4.18(b) for the following reason. The motion planner restricts the wheelchair to only forward motion, and the ADAAG assumes backing up to the final side transfer position. Therefore, the motion planner positions the A_{wc} robot in a position to make the backup move to the final side transfer position.

4.2.4 Determining the R_{seg} Components

The previous sections in this chapter established the C_{36} configuration space based on the A_{36} robot, the initial point, and the goal point for the motion planner to generate NF2 potential field. This section describes the formulation of the non-holonomic wheelchair path τ using this NF2 potential field in the C_{36} configuration space to guide the A_{wc} robot and the $\mathsf{C}_{wc0}\ldots\mathsf{C}_{wcn}$ configuration spaces to perform the collision-checking.

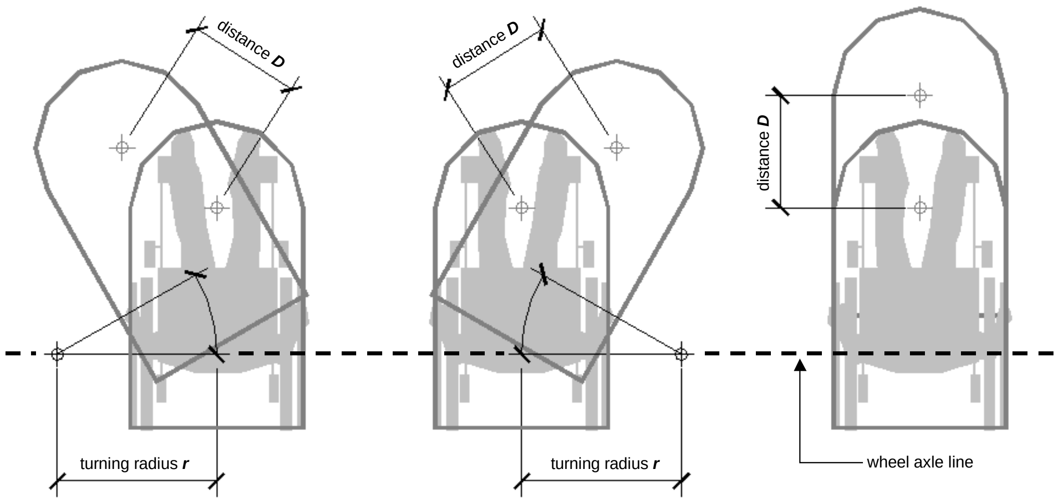


Figure 4.20: The three options (left, right, and straight) for the A_{wc} robot.

The motion planner describes the non-holonomic path by restricting the A_{wc} robot to three moves: a left turn, a right turn, and a straight-ahead move. Figure 4.20 illustrates these three options. The motion planner describes the vertex of the turning angle as the perpendicular length *r* from the centerpoint between the major wheelchair wheels. The motion planner records the actual position of the A_{wc} robot at the centerpoint of the half-dodecagon at the front of the robot. The displacement distance *D* from either turn (which is dependent on *r*) dictates the translation of the A_{wc} robot for the straight-ahead option.

The performance-based accessible route path planner restricts the value of *r* to two values, r_1 and r_2 using a two-step approach. As the wheelchair user nears a goal, the user naturally slows down allowing finer maneuvering with a smaller turning radius. The larger turning radius r_1 (r_1 equals 24 inches) is employed to move the A_{wc} robot to the goal point. When the wheelchair has moved within an 18-inch locus of the goal point, the motion planner switches to the smaller turning radius r_2 (r_2 equals 9 inches) to try to maneuver the A_{wc} robot to the goal point with an acceptable orientation.

The pull side approach maneuvering clearance geometry parameters in Figure 4.2 translate to the use of this two-step turning radius approach. The wheelchair user can use the faster r_1 turning radius when approaching the door, and the 60-inch depth of the maneuvering clearance implies the use of the slower, tighter turning radius r_2 in the proximity of the door.

The motion planner uses the NF2 potential field in the $\mathbf{C}_{36}$ configuration space to guide the $\mathbf{A}_{wc}$ robot using the following algorithm. Starting from the initial position and orientation $\mathbf{q}_{\texttt{init}}$, the motion planner examines the three move options, left, right, and straight ahead ($\mathbf{q}_{\texttt{left}}$, $\mathbf{q}_{\texttt{right}}$, and $\mathbf{q}_{\texttt{straight}}$ using r_1 for the robot turning radius. If $\mathbf{q}_{\texttt{left}}$ resides in the $\mathbf{C}_{36}$ and appropriate $\mathbf{C}_{wci}$ configuration spaces free space, the motion planner compares the (x,y) associated with $\mathbf{q}_{\texttt{left}}$ with the (x,y) associated with $\mathbf{q}_{\texttt{goal}}$. If the (x,y) values not equal, the motion planner looks up the potential field value $\mathbf{U}_{\texttt{left}}$, creates a node containing $\mathbf{q}_{\texttt{left}}$ and $\mathbf{U}_{\texttt{left}}$, and inserts the node into a priority queue which prioritizes nodes by their $\mathbf{U}$-value (the lower the value, the higher the priority). Finally, the motion planner inserts a pointer to the previous position (in this case, $\mathbf{q}_{\texttt{init}}$) in the node and marks $\mathbf{q}_{\texttt{left}}$ in the appropriate $\mathbf{C}_{wci}$ configuration space potential field as having been already visited. The motion planner repeats this procedure for $\mathbf{q}_{\texttt{right}}$ and $\mathbf{q}_{\texttt{straight}}$.

The motion planner continues this iterative process by removing the highest priority node (the node with the lowest potential value) from the priority queue and examining the three move options from the associated $\mathbf{q}$. Now, $\mathbf{C}_{wci}$ configuration spaces include the visited as well as the free space information, and the motion planner treats a visited $\mathbf{q}_{\texttt{init}}$ as an obstacle. When $\mathbf{q}$ is within an 18-inch locus of $\mathbf{q}_{\texttt{goal}}$, the planner starts generating new positions using the smaller turning radius r_2. The iterative process continues until either the motion planner empties the priority queue (indicating no path τ exists) or (x,y) associated with the current $\mathbf{q}$ matches the (x,y) associated with $\mathbf{q}_{\texttt{goal}}$. With a match, the motion planner examines the orientation θ associated with the current $\mathbf{q}$ against the θ associated with $\mathbf{q}_{\mathtt{goal}}$. If the current θ lies within the allowed range of $\theta_{\mathtt{goal}}$, the motion planner records the path $\boldsymbol{\tau}$. Otherwise, the motion planner continues the iterative process until the motion planner empties the priority queue (no path $\boldsymbol{\tau}$) or the current $\mathbf{q}$ matches with $\mathbf{q}_{\mathtt{goal}}$ for both position and the acceptable orientation range.

Finally, the performance-based analysis implements a one-step iterative process using only the r_2 turning radius to test for the turning circle requirement in a space. The $\mathbf{A}_{\mathrm{wc}}$ robot may not be able to achieve the goal point using the first step in the two-step process, so the motion-planner uses the tighter turning radius for this maneuvering requirement.

In determining the value for the turning radius r_1, a larger value represents a larger turning circle and a more comfortable path $\boldsymbol{\tau}$ for the wheelchair user. A trial-and-error method determines the largest possible value for r_1. The prescriptive-based accessible route development earlier in this chapter described the deficiencies of the prescribed accessible route parameters in the ADAAG. However, since the prescribed accessible route parameters in the ADAAG are designed to define usability, this research uses the prescriptive parameters to determine the values for the r_1 turning radius.

The motion planner utilizes the prescribed 36-inch path width to construct the $\mathbf{C}_{36}$ configuration space since a 36-inch width represents a comfortable width for the wheelchair user. Now, the trial-and-error method uses the non-holonomic motion-planning techniques described in the previous section on the two turning-around-an-obstruction configurations from the ADAAG Provision 4.3.3 shown in Figure 4.6 to determine r_1. (Note that the motion-process described above describes a two-step process using two r-values, but here, the motion planner uses a one-step process using r_1). The trial-and-error method begins with a value for r larger than the final value for r_1 (the larger the turning radius, the more comfortable the wheelchair path decrements this r

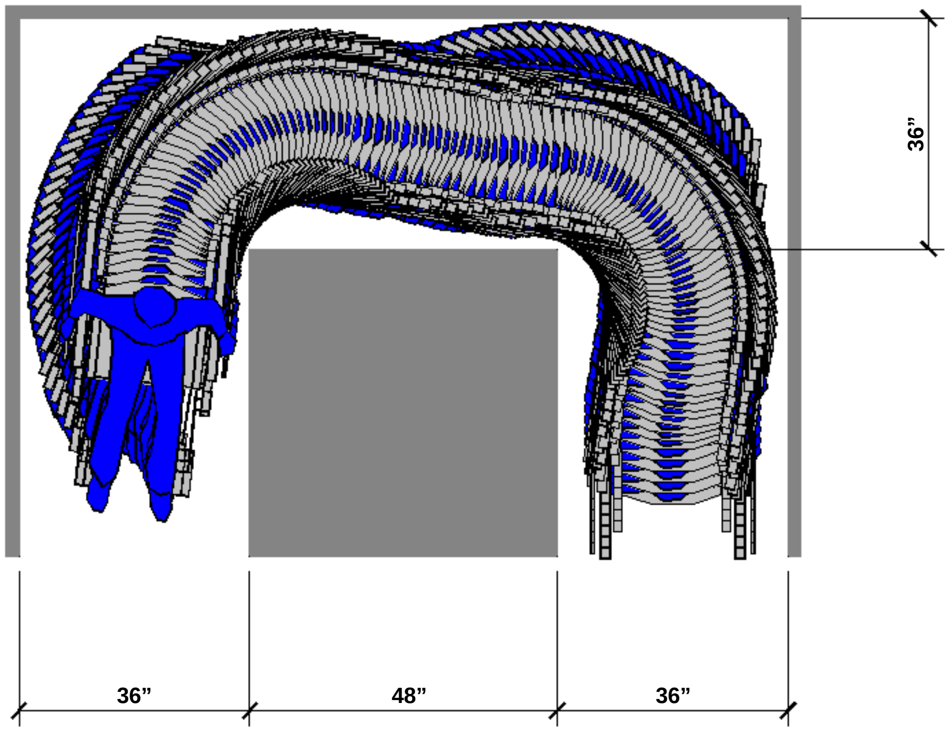


Figure 4.21: Motion-planning results for the first ADAAG 4.3.3 exception, r_1 = 24”.

value to until it finds a legal path τ). Figure 4.21 and Figure 4.22 illustrate the first legal paths with an r_1-value that works for both turning-around-an-obstruction configurations produced by the trial-and-error method: r_1 is equal to 24 inches.

As with the determination of r_1, a trial-and-error method uses a provision from the ADAAG to determine r_2:

4.2.3 Wheelchair Turning Space. The space required for a wheelchair to make a 180-degree turn is a clear space of 60 in (1525 mm) diameter (see Fig. 3(a)) or a T-shaped space (see Fig. 3(b)).

The trial-and-error method does not use the motion planner to determine r_2 since it only finds the maximum turning radius that can make the A_{wc} robot perform the turning

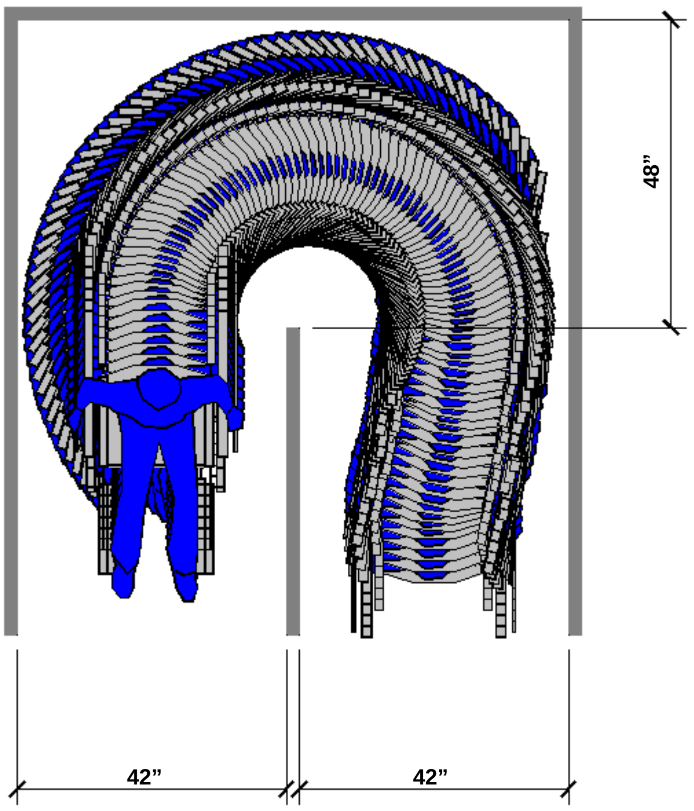


Figure 4.22: Motion-planning results for the second ADAAG 4.3.3 exception, r_1 = 24”.

maneuver in a 60-inch space. Figure 4.23 illustrates the turning maneuver that satisfies the 60-inch horizontal constraint using a turning radius r_2 of 9 inches.

Note that the y-dimension exceeds the 60-inch diameter clearance requirement. The ADAAG Appendix notes that, in practice, the vertical dimension should actually exceed 60-inches:

A4.2.3 Wheelchair Turning Space. These guidelines specify a minimum space of 60 in (1525 mm) diameter or a 60 in by 60 in (1525 mm by 1525 mm) T-shaped space for a pivoting 180-degree turn of a wheelchair. This space is usually satisfactory for turning around, but many people will not be able to turn without repeated tries and bumping into surrounding objects. The space shown in Fig. A2 will allow most wheelchair users to complete U-turns without difficulty.

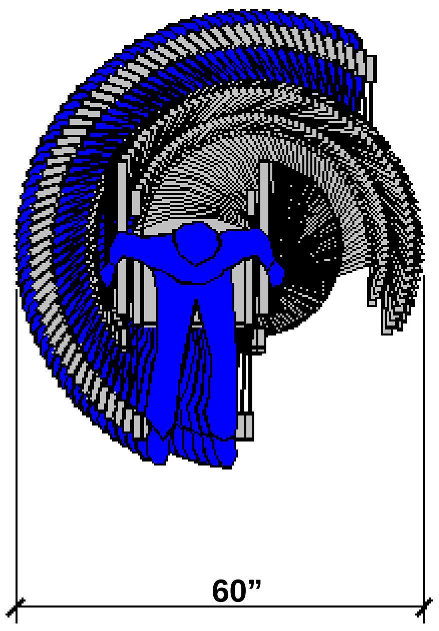


Figure 4.23: Motion-planning results for the turning radius, r_2 = 9”.

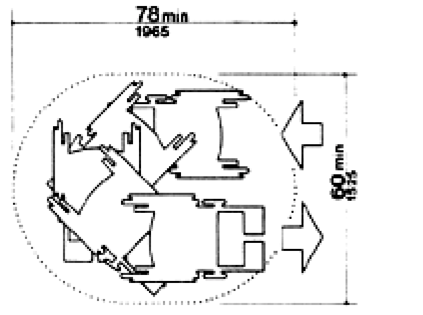


Fig. A2
Space Needed for Smooth U-Turn in a Wheelchair

Figure 4.24: ADAAG Figure A2 illustrating the actual turning clearance geometry [1].

Figure 4.24 illustrates an acceptable clearance oval, and the turning formulation in Figure 4.24 easily fits into the suggested oval geometry.

4.2.5 Performance-based Analysis Discussion

The performance-based motion-planning techniques developed in this chapter directly capture motion and behavior given the wheelchair's parameters as described in the ADAAG. This direct analysis obviates the need for the complicated exception analysis associated with the prescriptive-based ADAAG accessible route parameters, an artifact that is a consequence of the ADAAG's prescribed accessible route abstraction.

Similar to the prescriptive-based motion-planning analysis developed in this chapter, the performance-based motion planner can guarantee the discovery of the 36-inch wide path provided that one exists. However, it is possible that the non-holonomic planner developed for the actual wheelchair motion might not find a path even if one exists. One reason concerns the discretization of the configuration space:

As opposed to the solution described in Section 4.1.5 for the ADAAG-width-compliant motion planner, determining the necessary discretization granularity for the non-holonomic configuration space is not straightforward since the location of the wheelchair robot's next possible position (using trigonometric functions) may not correspond to the exact grid discretization. A possible extension to this research involves developing techniques that guarantee the discovery of a path if one exists given the described performance-based non-holonomic motion-planning techniques. Hsu et al. presents related work developing a randomized motion planner for robots under kinematic and dynamic constraints in which the probability that the planner fails to find a path when one exists converges toward zero [29]. The following describe some other possible extensions to the performance-based analysis.

Figure 4.15 illustrates the prescribed ADAAG dimensions for a wheelchair, and the performance-based accessible route analysis uses the ADAAG wheelchair to develop the A_{wc} robot parameters. The prescriptive nature of the code creates an indirect relationship between provision parameters and the wheelchair dimensions and behavior. No cause-

Table 4.1: Wheelchair parameters and influencing factors.

Parameter	Influencing Entity	Constraints
A_{36}	users, manufacturers	user comfort level wheelchair physical dimensions
A_{wc}	manufacturers	wheelchair physical dimensions
r_1 and r_2	users, manufacturers	user comfort level wheelchair physical dimensions wheelchair speed
r centerpoint	manufacturers	wheel dimensions, placement, action

and-effect relationship exists between the wheelchair constraints and the prescriptive route usability analysis.

In contrast, a designer can vary the parameters developed in this chapter. Varying specific parameters allows wheelchair manufacturers and users to test the behavior of a specific wheelchair model or assign personal preferences and simulate wheelchair movement in a specific design configuration. Varying the A_{36} robot's diameter (and influencing the C_{36} configuration space) allows the user to choose a preferred path width comfort level independent of the actual wheelchair parameters. Of course, the diameter should exceed the wheelchair width. Wheelchair manufacturers make wheelchairs with various physical dimensions, and the performance-based analysis can easily capture these dimensions to model the A_{wc} robot. In addition, the turning radius and centerpoint of the turn depends on the wheelchair's mechanical constraints. Finally, independent of these mechanical constraints, users have their own comfort level associated with the possible turning radii r_1 and r_2. Table 4.1 summarizes the variable parameters and the influencing factors.

The non-holonomic path-planning analysis developed in this chapter limits the wheelchair motion to three options: left, right, and straight. The motion planner imposes

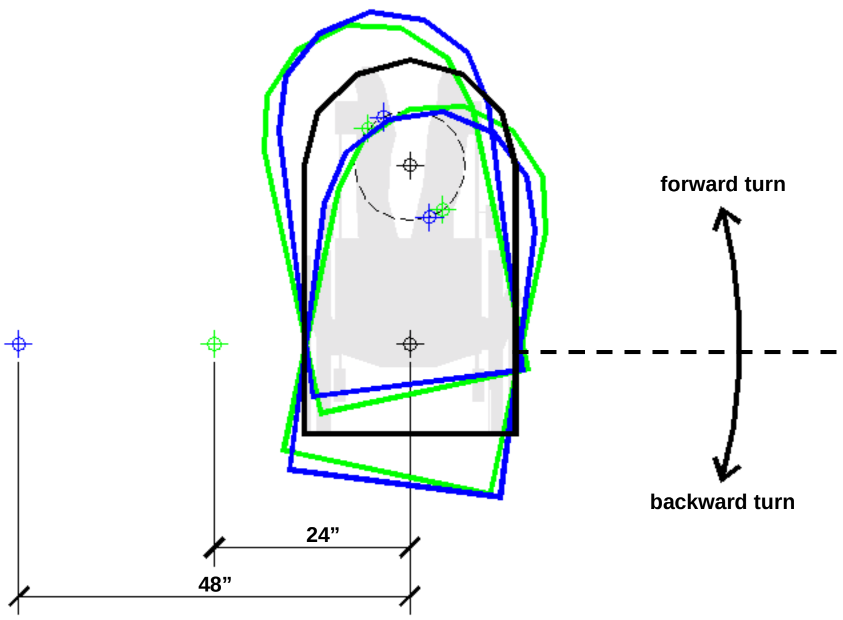


Figure 4.25: Left-hand-turn options (forward, backward, *r* = 24”, 48”).

this restriction in an attempt to capture the intent of the prescribed code and match the compliance results of the ADAAG accessible route provisions.

The number of options of forward motion can be increased to 1+2*f* where 1 represents the straight-ahead move and *f* represents the number of same direction (left or right) forward turns. As with the three-move formulation, the new positions **q** for all moves should be equidistant from the starting position. In addition, the motion planner can support backward motion. Reeds and Shepp describe optimal paths for a car-like robot that is allowed to reverse its direction [55]. Indeed several ADDAG provisions assume backward motion. Now, the number of options of motion can be increased to 1+2*f*+2*b* where *b* represents the number of same direction (left or right) backward turns. The motion planner would set a limit on the number of backups for a given path $\boldsymbol{\tau}$ according to the building code’s or user’s specifications. Figure 4.25 illustrates two left-hand turning radii supporting forward and backward motion (the same turning radii are used for both the forward and backward turns). The dashed circle represents the equidistant locus around the starting position of the A_{wc} robot and note that the turning angles of all four moves have been set for the equidistant movement.

4.3 A Comparison of Prescriptive- and Performance-Based Analysis Results

This section demonstrates the deficiencies of the prescriptive-based accessible route formulation by analyzing design configurations against the prescriptive parameters from the ADAAG and comparing the results to the developed performance-based analysis. Table 4.2 delineates the possible combinations of the two analyses. The performance-based analysis uses specific provisions from the ADAAG to instantiate the turning radius parameters, and by default, the tested configurations were both code-compliant and usable (Table 4.2, Entry 1). Providing examples that are both non-compliant and unusable (Table 4.2, Entry 2) can be trivially demonstrated with a less-than-36-inch-wide corridor. The prescriptive-based analysis is by nature limited in its description of possible design configurations, and this section presents an example of a non-compliant route that a wheelchair user can actually negotiate (Table 4.2, Entry 3). Similarly, the section presents an example of a code-compliant route that a wheelchair user cannot negotiate (Table 4.2, Entry 4).

Wheelchair users can comfortably use an infinite number of design configurations that do not comply with the prescriptive accessible route provisions from the ADAAG. Because of the prescriptive nature of the disabled access code, it cannot address all possible configurations; the code limits the special cases it addresses to the turn-around-an-obstruction exceptions, and Section 4.1.4.2 has described a configuration that reveals a conflict in the ADAAG.

Table 4.2: The possible prescriptive/performance analysis combinations.

	Prescriptive-based Formulation	Performance-based Formulation
1	Compliant	Usable
2	Non-compliant	Unusable
3	Non-compliant	Usable
4	Compliant	Unusable

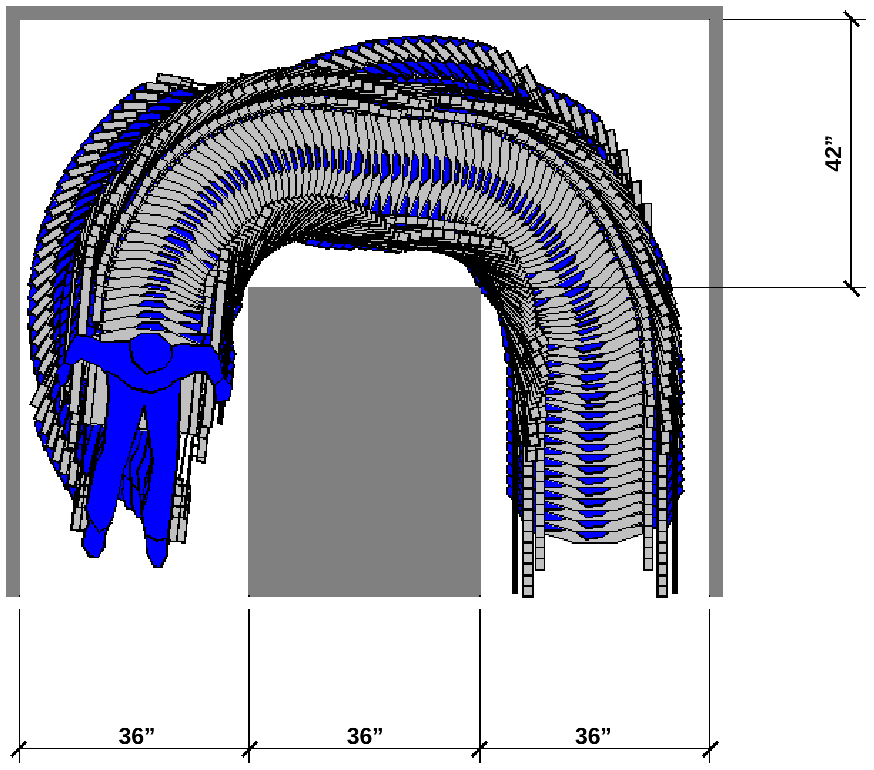


Figure 4.26: The non-compliant, usable example.

Example 1

This example presents a design configuration illustrated in Figure 4.26 that clearly falls under the U-turn-around-an-obstacle exception category: the width of the obstruction is less than 48 inches, and the configuration cannot be transformed into the 90-degree-turn-around-an-obstacle exception by making the obstruction wider than 48 inches. Following the parameters of the ADAAG Provision 4.3.3, the configuration fails to comply with the exception that:

- The widths of the first and third legs are less than 42 inches.
- The width of the second leg is less than 48 inches.

Using the performance-based parameters established in the performance-based analysis, the motion planning simulation returns a successful path $\boldsymbol{\tau}$ around the obstruction for the non-compliant configuration as illustrated in the figure. Thus, the configuration is usable by a wheelchair user.

Example 2

Wheelchair users cannot comfortably negotiate an infinite number of design configurations that comply with the prescriptive accessible route provisions from the ADAAG. This example demonstrates an example illustrated in Figure 4.27. Following the parameters from the ADAAG Provision 4.3.3, the design complies with the code in that:

- The accessible route is equal to or greater than 36 inches wide.
- Neither turn-around-an-obstruction exception applies.

Note that if the angle between the second and third leg equals 90 degrees instead of exceeding 90 degrees, the first turn-around-an-obstruction exception from Provision 4.3.3 would apply. The building official may contend that the exception applies with a small ε, but as ε grows, the configuration does not qualify for the exception. The ambiguity of at what point the prescribed configuration applies illustrates another deficiency of a prescriptive-based approach.

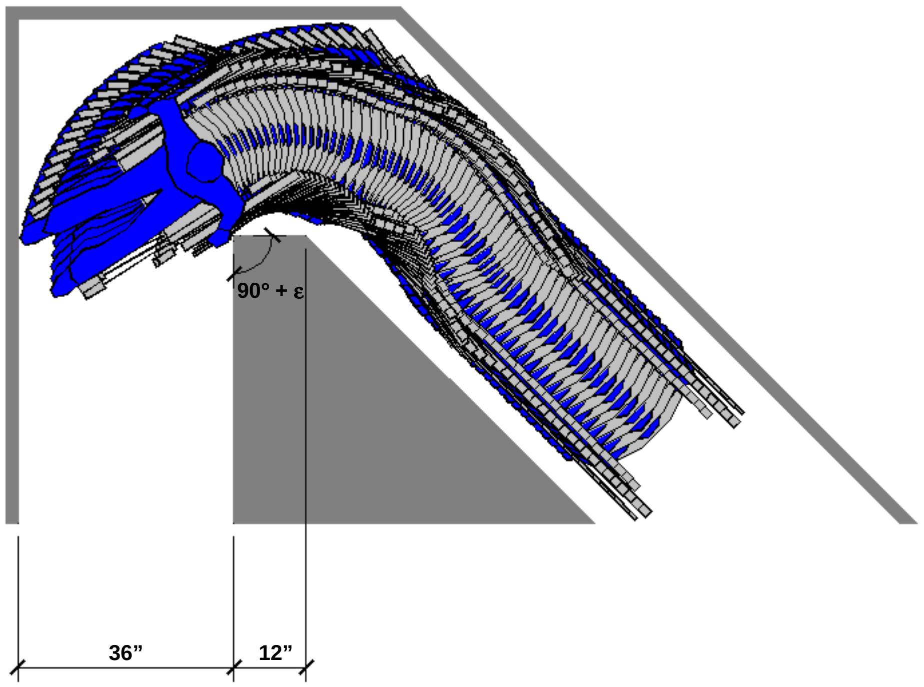


Figure 4.27: The code-compliant, unusable example.

Using the performance-based parameters, the motion planning simulation fails to return a path **τ** around the obstruction for the code-compliant configuration. For this example, a modified motion planner keeps track of the lowest potential-value node visited, and in the event of failure, as illustrated in Figure 4.27, the motion planner returns a path that gets the wheelchair as close to the goal point as possible. A trial-and-error method of iteratively extending the length of design-configuration’s second leg in increments of one inch and running the motion planner yields the usable design configuration shown in Figure 4.28. Alternatively, an iterative trial-and-error method could have been used to establish the second leg’s minimum width that ensures usability.

Finally, by changing the angle between the second and third legs of the route from 90°+ε to 90° or 90°-ε, the motion planner can be used to demonstrate the overly-restrictive

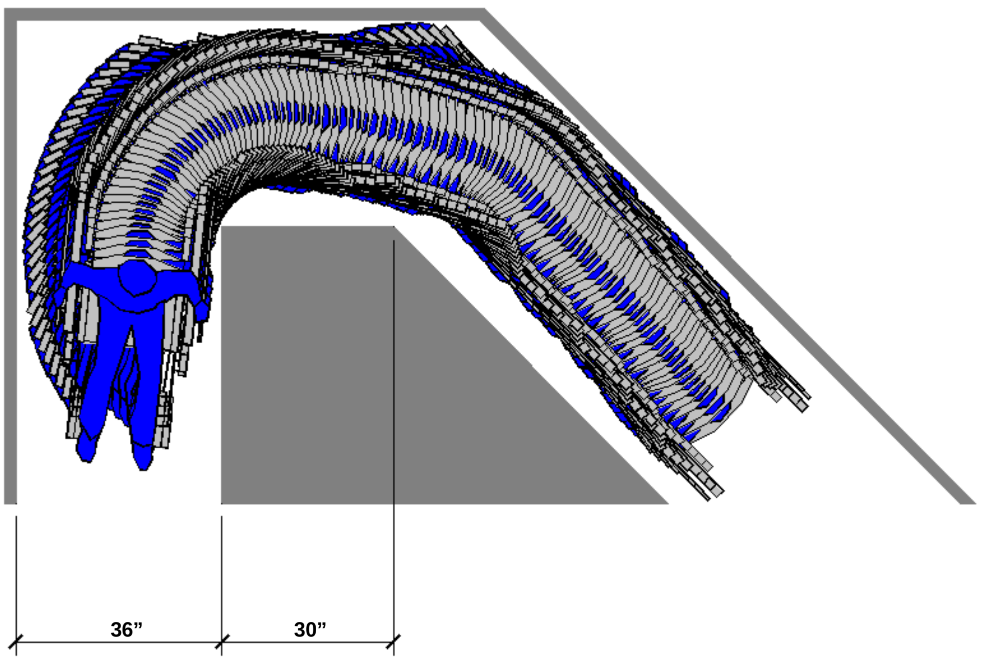


Figure 4.28: Modifying the middle leg of the route makes the configuration usable.

nature of the 90-degree-turn-around-an-obstruction exception from Provision 4.3.3. (While not explicitly stated, the exception should apply to angles less than 90 degrees since this configuration would constitute a more difficult accessible route). As illustrated in Figure 4.28, the second and third leg dimensions provide a viable path $\boldsymbol{\tau}$ around the obstruction, and these lengths are clearly less than the required 48-inch/48-inch exception requirement.

4.4 Wheelchair Manipulation and Animation

So far, this research has argued that if the behavior is quantifiable, performance-based analysis methods are superior to analogous prescriptive-based methods. While this research has successfully quantified critical accessible route-related behavior, this research develops an additional tool placing the designer in a virtual wheelchair that can be navigated through the facility design environment. Though the prescriptive- and performance-based methods provide information regarding the code-compliance and usability of a facility, visual presentation of this information can provide further insight into each type of analysis.

This research develops several algorithms associated with the manipulation and animation of the virtual wheelchair. The manipulation algorithms translate the joystick device output parameters to move the virtual wheelchair. The animation algorithms translate the generated accessible route to the various wheelchair movements associated with wheelchair motion.

4.4.1 Joystick Manipulation of the Wheelchair

This section maps the corresponding joystick positions to the wheelchair movement. The section will refer to Figure 4.29 to describe the relationship between the joystick position and the generated wheelchair motion.

Any (x,y) value with $x = 0$ constitutes straight motion with $y > 0$ corresponding to forward motion and $y < 0$ corresponding to backward motion with the major wheels moving with the same angular velocity ω in the same direction. The pairs $(0, \mathtt{y}_{\mathtt{max}})$ and $(0, \mathtt{y}_{\mathtt{min}})$ (A and E in Figure 4.29) correspond to the maximum forward and backward velocities $\mathtt{v}_{\mathtt{max}}$ and $-\mathtt{v}_{\mathtt{max}}$, and any velocity v corresponding to a joystick position on y-axis can be defined by the equation:

$$v = \mathtt{v}_{\mathtt{max}} * (y \; / \; \mathtt{y}_{\mathtt{max}})$$

For “right turn” wheelchair navigation, the designer moves the joystick within the two right quadrants of Figure 4.29. Intuitively, as the joystick position moves from A to B, the wheelchair begins to turn to the right. To achieve this behavior, the right wheel’s

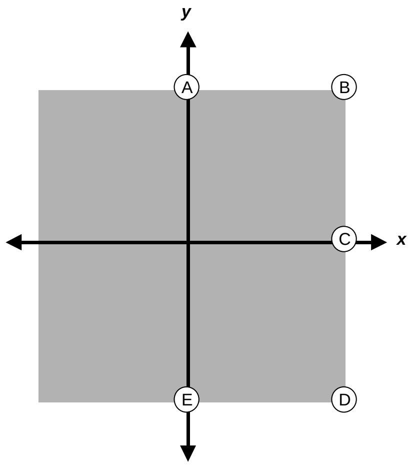


Figure 4.29: The joystick coordinate system.

angular velocity ω_r decreases. At point B, ω_r equals 0, and the wheelchair rotates around the right wheel's point-of-contact with the ground. As the joystick position moves from B to C, ω_r decreases from 0 (in the opposite direction of the left wheel's angular velocity ω_l). At point C, ω_r equals $-\omega_l$ and the wheelchair rotates at the midpoint between the two wheel's point-of-contact with the ground. Similarly, from C to D, ω_r decreases. At point D, ω_r equals 0, and from D to E, ω_r decreases from 0 until at point D, ω_r equals ω_l, and the wheelchair moves straight backward. The opposite "left turn" behavior occurs when the joystick position moves from A to E in the counterclockwise direction.

The *x*:*y* ratio determines the turning radius centerpoint and the magnitude of the *xy* vector determines the wheelchair velocity, at A, the turning radius *r* is infinite. At B, *r* is equal to half the distance between the two wheels, and at C, the turning radius *r* is 0.

Determining the turning radius *r* formally, if $y > 0$ (forward movement), set the sector that the left wheel travels through s_l equal to 1. The sector that the right wheel travels through is defined as:

$$s_r = \cos(2 * \text{atan}(|y / x|))$$

If $y < 0$ (backward movement), set s_r equal to 1. The sector that the left wheel travels through is defined as:

$$s_l = -\cos(2 * \mathrm{atan}(\,|y / x|\,))$$

Now, the angle that the wheelchair sweeps through is defined as:

$$\theta = (s_l - s_r) / d$$

where d equals the distance between the wheels. Finally, the turning radius is defined as:

$$r = (s_l / \theta) - (d / 2)$$

4.4.2 Wheelchair Animation Techniques

The design-aid framework allows the designer to qualitatively analyze an accessible route through the facility design by allowing the designer to experience the path from two points of view. In addition to allowing the designer to observe the wheelchair, the design-aid frameowork provides a "wheelthrough" view analogous to a "walkthrough" that is provided by many visualization packages. This section describes the animation techniques used to generate and coordinate wheelchair and the wheelchair user.

Figure 4.30 illustrates the geometric (rotational) relationship hierarchy of the wheelchair and the wheelchair user. The animation algorithm uses this hierarchy to generate appropriate behaviors of the moving parts of the wheelchair as well as the human locomotion from a two-dimensional path. Using these hierarchical relationships, given a polygonal path, the behavior of the wheels, both the major wheels and the casters, can be accurately modeled providing realistic animation of the wheelchair motion. The animation algorithm assumes manual locomotion, and the rotation of the back wheel depends on the wheelchair user's hand and arm movement. The animation algorithm simplifies the anthropomorphic constraints using inverse kinematics to determine the forearm and upper arm in relationship to the hand.

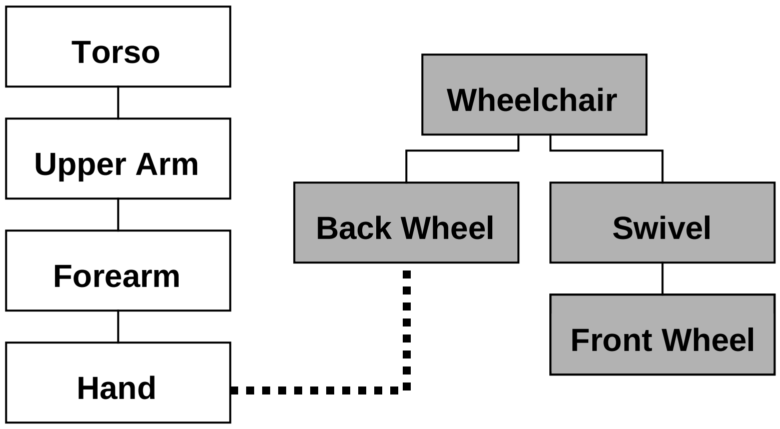


Figure 4.30: Wheelchair and wheelchair user geometry hierarchy.

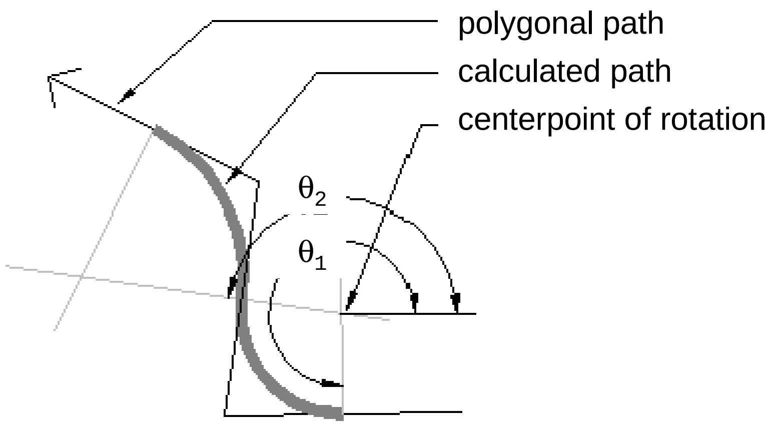


Figure 4.31: The smoothing of the polygonal path of the wheelchair.

Figure 4.31 illustrates the smoothing algorithm. The path-smoothing algorithm uses as input a polygonal path of equal-length segments. The smoothing algorithm examines a sequence of two segments and calculates an arc that sweeps from the midpoint of each segment and is tangential to each segment.

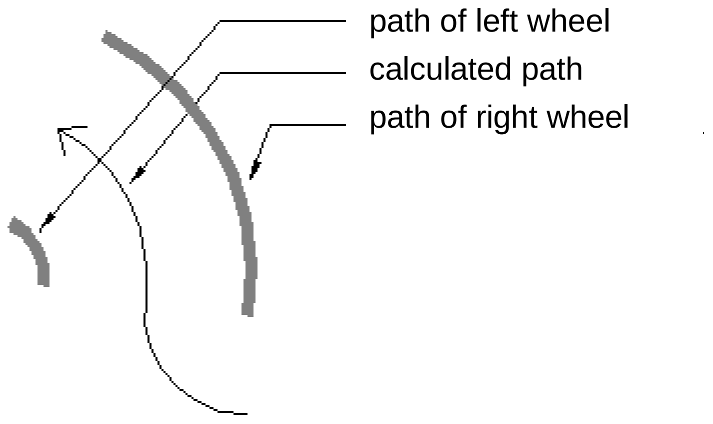


Figure 4.32: The paths of the major wheels for the given path of the wheelchair.

There are two exceptions to this computation:

1. When the segments are co-linear and pointing in the same direction.

2. When the segments are co-linear and pointing in opposite directions.

In the first case, the algorithm records the translation information holding the angles of rotation (θ_1 and θ_2) constant. In the second case, the algothim sets the point of rotation at the midpoint of the two segments (the same point) and θ_2 is set to $\theta_1+\pi$. At this point, a node in the data structure contains the centerpoint information, the angle sweep information, and the translation information of a single arc.

Figure 4.32 illustrates the path of the major wheels of the wheelchair. The rotation of the left and right major wheels is calculated as follows. Here, the algorithm calculates the length of the arc that is swept by each wheel dependent on the calculated path of the center of the wheelchair. Now, the algorithm calculates the sweep of each wheel dependent on the wheel's radius.

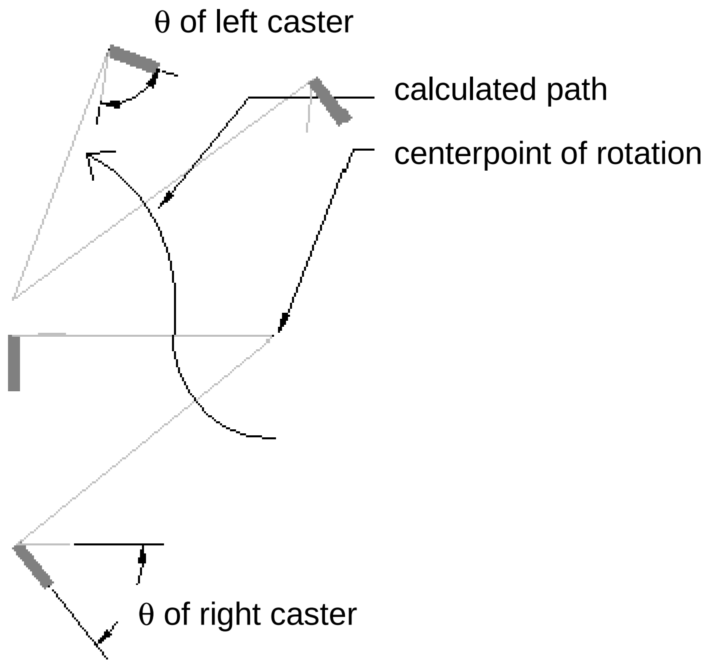


Figure 4.33: The coordination of the swiveling casters.

It is interesting to note that if the radius of arc of the calculated path of the center of the wheelchair is smaller than the distance of a wheel from the centerpoint, then the wheel's turning motion goes in the opposite direction. When a wheelchair is spinning on its axis, the wheels are rotating in completely opposite directions.

Next, the algorithm calculates the swivel of each caster as illustrated in Figure 4.33. The caster tries to position itself perpendicular to the vector from the centerpoint of the rotation of the wheelchair to the point of swivel. The previous position of the caster is the starting point, and the caster will then tend towards the desired next position.

The caster may not achieve the final desired position. The algorithm allocates 1/18 π per timestep using a the timestep of 1/10th of a second. If the caster cannot reach the desired goal angle before the end of the calculated arc, it tries to achieve to the next position based on the rotation of the wheelchair at the next arc. It is also interesting to note that

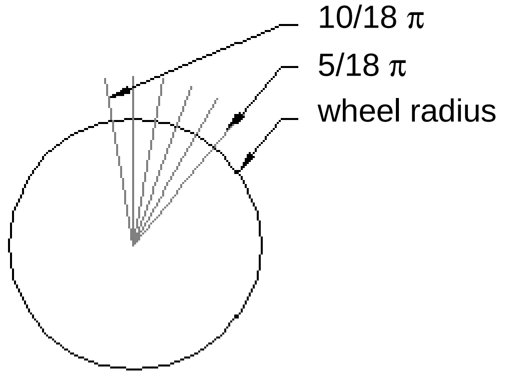


Figure 4.34: Arm position range-of-motion.

each caster's desired angular position is different (except when the wheelchair is only moving forward and not rotating).

Finally, the algorithm coordinates each arm motion is coordinated with the corresponding major wheel. Here, the positions of the arm have been pre-calculated for six positions ranging from 5/18 π to 10/18 π as shown in Figure 4.34. The range corresponds to the range that the model of the figure can reach the wheel.

At each keyframe, the animation environment interpolates the arm position between the six pre-calculated positions. When the arm position goes beyond the range, the arm is placed at the opposite position to start pushing (or pulling) the wheel again.

4.5 Summary

This chapter first presented an automated analysis of the accessible route R components given the ADAAG prescribed parameters. This chapter showed that the prescriptive formulation has the following deficiencies:

- In the case of the maneuvering clearances associated with several building components, the clearance geometries (that are used to determine the usability of these components) depend on the accessible route approach to the components. Thus the ADAAG must prescribe multiple clearance geometries if there are multiple approaches, and the developed analysis methods must test for the existence of all the possible clearance geometries.

- The prescribed path width does not sufficiently provide usability in all configurations. For this reason, the ADAAG adds special cases (exceptions in the form of additional prescribed geometries) to address this deficiency, and the developed analysis method must test for the general case (the general prescribed path width) as well as the exceptions.

- Most importantly, the mapping from the accessible route design intent to prescribed parameters can lead to limited, incorrect, or conflicting formulations. As noted in this chapter, the exceptions to the prescribed general path width only address two special configurations whereas many more exist and are not addressed by the prescriptive-based code. In addition, this chapter has described a configuration in which the prescribed geometries of the path width exceptions lead to conflicting and incorrect analysis.

Next, this chapter described a motion-planner developed to capture wheelchair motion as the wheelchair user moves along an accessible route. The chapter discussed several advantages that the performance-based approach has over the prescriptive-based approach. Specifically, the motion-planning simulation alleviates the need for multiple clearance geometries associated with different approaches to a building component and the need for prescribed exception configurations to the basic prescribed accessible route width requirement.

While the prescriptive-based approach cannot address all possible configurations and thus is limited in describing the usability of a facility, the performance-based formulation does not suffer from this limitation. The performance-based formulation provides a more accurate analysis of facility usability since it directly models the wheelchair behavior. The chapter also described variations to the non-holonomic path formulation extending from the code-compliant parameters. The flexibility of the performance-based approach allows varying the parameters to directly affect changes to the wheelchair motion behavior.

The chapter then presented examples comparing the prescriptive-based code-compliant accessible route analysis with the performance-based usable accessible route analysis. The first example demonstrated a non-compliant but usable design configuration. The second example demonstrated a code-compliant but unusable design configuration and a modification to the original configuration to make it usable. The direct comparison of the two analysis methods and the ability to vary the wheelchair parameters (and thus the behavior of the path-planner to accommodate specific users and wheelchairs) show the advantages of the performance-based accessible route analysis over the prescriptive-based analysis.

Finally, the chapter described the wheelchair manipulation and animation algorithms used to realize the interaction and visualization of the virtual wheelchair. These interaction and visualization tools provide the designer with qualitative insight into the disabled access problem. From the wheelchair user’s viewpoint, the designer can observe what the wheelchair user observes, and the tool can influence critical design issues such as window and signage placement, issues that are beyond the scope of the prescriptive- and performance-based analyses. The joystick interaction provides the designer with a virtual environment that most closely resembles the interaction with the actual facility by accurately mapping joystick manipulation to wheelchair motion. Similar to the flexibility of the performance-based analysis, manufacturers can customize this mapping to the particular constraints of individual wheelchair models.

Chapter 5

The Design-aid Framework as a Distributed Object Service Environment

This research develops and implements the design-aid framework giving the designer access to the analysis tools developed in the previous chapters. These tools include:

- The prescriptive-based and performance-based disabled access analysis of a facility design.
- The ability to manipulate a virtual wheelchair through the facility design.
- The ability to transfer the facility design data from a commercial CAD package to the design-aid framework giving the designer access to the analysis tools.

Three questions motivated the implementation development:

1. How can the research provide the developed analysis tools in a modular fashion that can be generalized for the integration of other services and tools into the design-aid framework?

2. How can the research leverage this modular approach to take advantage of disparate and distributed computing platforms?

3. Does the current offering of computer development tools and environments enable or hinder the development of the design-aid framework?

To answer these questions, this chapter describes the concepts used to realize the design-aid framework as an Internet-based Distributed Object Service Environment (DOSE) and the use of DOSE to develop the disabled access design-aid framework. This research uses the concepts developed in [25] and reifies notion of a *service*. To fully-leverage the power of the Internet, engineering and design services should be able to interact in a formal yet flexible manner. Services should be able to combine with existing services to provide added functionalities. The distributed object environment provides object transparency—an application accesses a `Service` object using the same protocol regardless of the object's location, either local or remote, and independent of the computer system platform assuming the platform supports the `Service` object interface.

The chapter is organized as follows:

- Section 5.1 describes the three-tiered `Service` object architecture.

- Section 5.2 describes the Visual Interactive Environment Workbench (VIEW) and its interaction with the various accessible route design aids that are implemented as aggregations of `Service` objects.

- Section 5.3 describes the implementation and related issues of the design-aid framework.

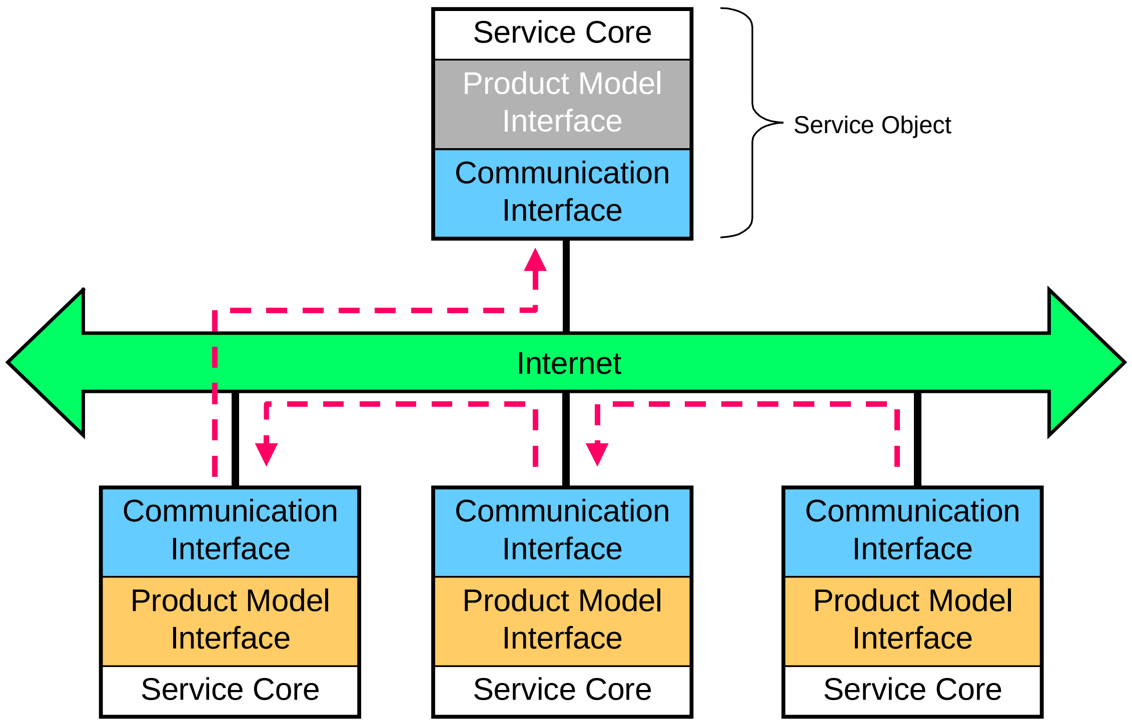


Figure 5.1: The conceptual diagram of a Distributed Object Service Environment (DOSE) instance.

5.1 The Three-Tiered Architecture

Figure 5.1 shows the conceptual network-enabled DOSE with four `Service` objects. In this environment, each individual service adheres to a three-tiered architecture. The first tier, a communication protocol or interface, gives the application services a common means to send and receive design data over the Internet. The middle tier, the optional common product model interface, is a standard protocol that describes the design data (in Figure 5.1, the top `Service` object has no product model indicated by the different color of the product model interface layer from the other `Service` objects). The third tier is the core of the design service—the design service extracts the appropriate information such as the building design through the common product model interface and then either modifies the design data or generates a report based on the analysis of the data.

One service can register with another service in the infrastructure. The registration and query of a `Service` object is based on a predetermined constraint language. When the core of a parent design service executes its analysis, it may send parts of the analysis to the child services that have registered with the parent service.

The communication protocol makes certain methods of the `Service` object public as shown in Figure 5.2. Following the object-oriented paradigm, the “exposed” methods are the points of entry into a service, but the actual implementation of these methods is dependent on the service. The `Service` object consists of two registration methods, `registerService()` and `registerDecompositionService()`, that allow a service to register with another service. A service registers using the `registerService()`method with a broker that will advertise the service. Any child service called by the parent service must register with the parent service using the `registerDecompositionService()`method. The structure of the `analyze()` method in Figure 5.2 is similar to the `Analyze()` method of the `Intent` object described in Chapter 3. The pseudo-code and subroutines of the `analyze()` method are shown as a suggested design guide in the figure but are not implemented in the interface since these methods do not need to be “exposed.” The `Service` object provides methods to send and receive data, `puts()` and `gets()`, in the form of string arrays. The `Service` object also provides both polling and callback mechanisms (`getStatus()` and `notification()`) to communicate with child services. Hence, the `Intent` object can be implemented as a `Service` object—with the registration and analysis methods (along with the suggested `analyze()` subroutines), the `Service` object supports the type of problem decomposition described in Chapter 3.

```
module DOSE
{
   interface Service;
   typedef sequence<string> StringArray;

   interface Service
   {
      void           registerService(in Service service);
      void           registerDecompositionService(in Service service);
      Service        getRegisteredService(in string id, in string type);
      Service        getRegisteredDecompositionService(in string id, in string type);

      string         getServiceId();
      string         getServiceType();
      void           putServiceId(in string id);
      void           putServiceType(in string type);

      void           analyze(in Service service, in string session, in string command);

/*
 * this loop shows the internal implementation of the analyze() method in pseudocode
 *                    {
 *                       preProcess();
 *                       for (int i=0; i < number of decomposition services; i++) {
 *                          pre-process the decomposition service (i);
 *                          execute the Analyze() method of decomposition service (i);
 *                          post-process the decomposition service (i);
 *                       }
 *                       postProcess();
 *                    }
 */

      boolean        getStatus(in string session, in string command);
      void           notification(in Service service, in string session, in string command);

      StringArray    gets(in string session);
      void           puts(in string session, in StringArray strings);
   };
};
```

Figure 5.2: The DOSE communications protocol.

Some services may not need the product model layer. For example, a brokering service that simply registers and advertises services does not need to process any design data. All design-related services, however, will use the product model layer. The DOSE implemented in this research assumes a common product model across all services using the product model described in Chapter 3. However, since the communications layer is decoupled from any product model semantics, the environment itself does not constrain the design data semantics, and services that utilize disparate product models must first use an intermediate translation service to transform the facility data into a usable format.

The product model interface receives the product model data from the communication layer as an array of strings via the `puts()` and `gets()` methods and stores the design data according to the individual service's needs. While not required, making the product model and the storage scheme consistent across the infrastructure makes it easier to reuse methods to extract the critical data from the product model and send this data to the core of the design service.

The `Service` object core layer executes the service's analysis using the `analyze()` method. The product model layer stores the design data, and the service core layer either uses this data directly or transforms the design data from the product model to a view or a diagram which is unique to the application. For the two decomposition subroutines associated with the `analyze()` method, the `Service` object must extract the relevant data, send it to the predetermined child services (via the `puts()` method), and instruct the child service to execute its `analyze()` method.

If the process that has initiated the `analyze()` method is not a `Service` object, the executing `Service` object does not have a mechanism to inform the initiating process upon completion of the task. Rather, the `Service` object simply waits to be polled (via the `getStatus()` method) by the initiating process. If the initiating process is a parent `Service` object, then the executing `Service` object (the child `Service` object) notifies the parent `Service` object (via the `notification()` method) upon completion of the task. If the initiating process expects to receive data from the child `Service` object, it executes the `gets()` command to retrieve the data.

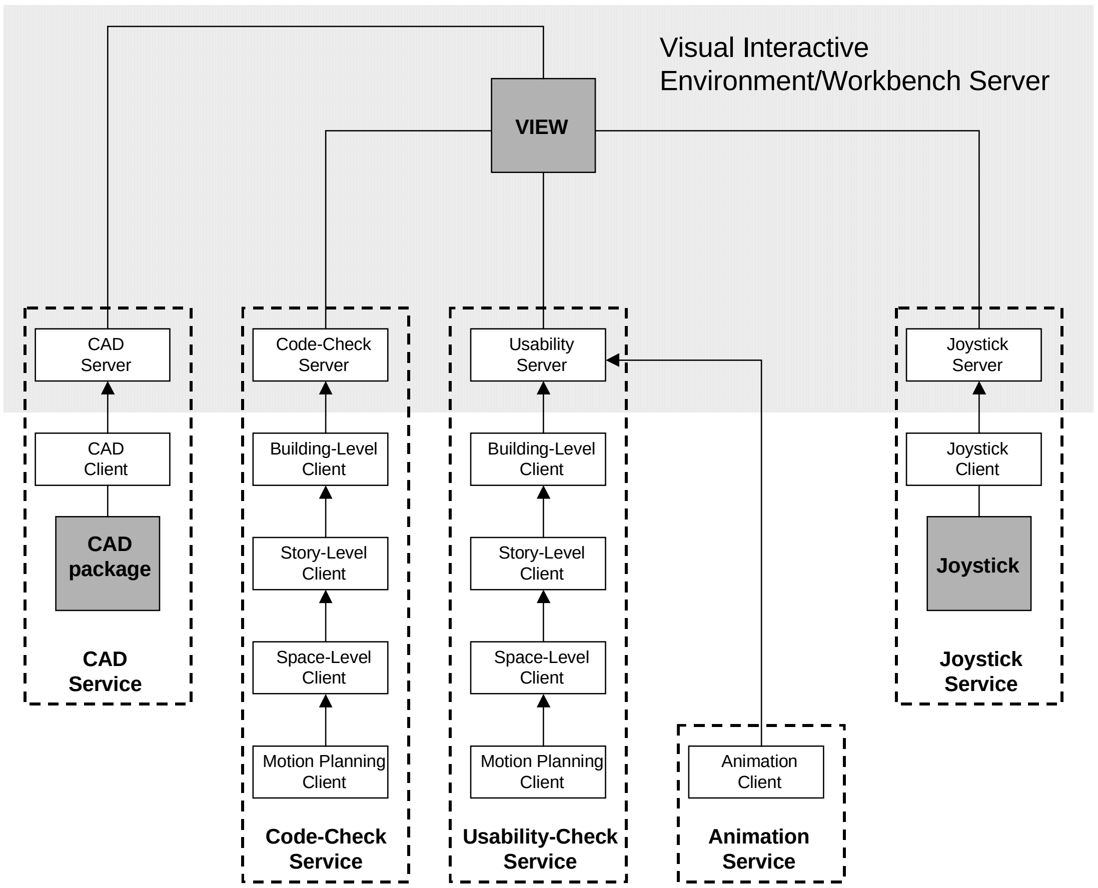


Figure 5.3: The DOSE diagram of the design-aid framework.

5.2 A Visual Interactive Environment/Workbench (VIEW) for Accessible Route Analysis

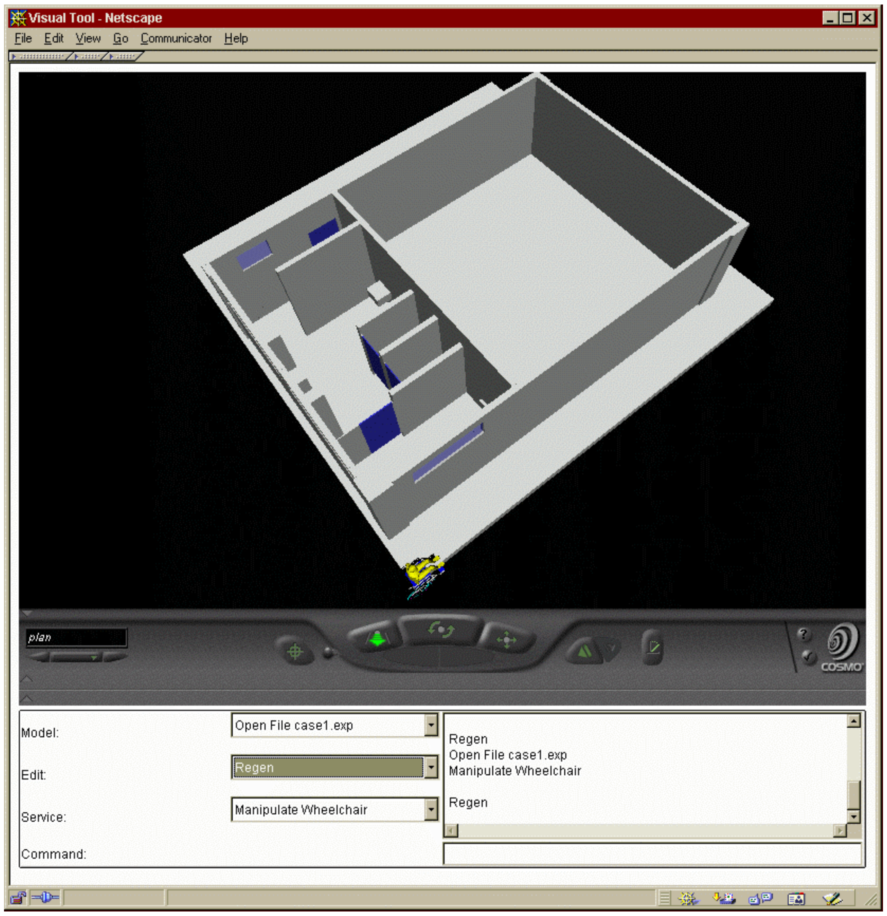
Figure 5.3 illustrates the design-aid framework from the point-of-view of the `Service` object infrastructure. The solid white boxes represent instances of `Service` objects, the gray boxes represent non-`Service` object applications, processes, or devices, and the dashed boxes aggregate `Service` and non-`Service` objects to form composite services. Connecting arrows indicate registration of a child `Service` object with its parent `Service` object.

Consistent with the design-aid framework described in Chapter 2 (illustrated in Figure 2.1 and Figure 2.6), the Visual Interactive Environment/Workbench (VIEW) constitutes the hub of the design-aid framework. Since the VIEW is not a `Service` object, it must poll each of the four composite services via the `getStatus()` method of the *Server* `Service` objects to attain the state of each composite service. These *Server* `Service` objects reside on the same computer system as the VIEW, but the other components of the composite services can exist anywhere and on disparate systems on the network thus exploiting the distributed object transparency paradigm.

The VIEW consists of a graphical user interface (GUI) controlled by the user via the keyboard, a mouse, and a joystick. Figure 5.4 illustrates the VIEW GUI with the bathroom facility design. Using the mouse, the designer can orient and navigate through the facility, pick and manipulate specific building components, and invoke the analysis services. The joystick provides the designer with a mechanism to manually move a wheelchair through the facility.

The following sub-sections describe the VIEW and the composite services in Figure 5.3 that communicate with the VIEW:

- Section 5.2.1 describes the VIEW module that provides the designer interaction between a commercial CAD package and the VIEW.
- Section 5.2.2 describes the integration of the prescriptive-based and performance-based analyses into the design-aid framework.
- Section 5.2.3 describes the module that allows the designer to move the wheelchair through the facility using a joystick analogous to the control device used with a motorized wheelchair.



Model: Open File case1.exp

Edit: Regen

Service: Manipulate Wheelchair

Command:

```
Regen
Open File case1.exp
Manipulate Wheelchair

Regen
```

Figure 5.4: The Visual Interactive Environment/Workbench (VIEW).

5.2.1 The VIEW CAD Service

The *CAD Service* enables a designer using a CAD package to download a facility design to the design-aid framework. The *CAD Client* `Service` Object registers with the *CAD Server* `Service` Object, and the CAD package uploads the design data to the VIEW using this client service as the intermediary between the CAD package computer system and the VIEW system. The *CAD Client* Service resides on the same system as the CAD package. Leveraging the distributed object paradigm, the CAD package usually resides on a different system than the VIEW (the designer can and probably does interact with

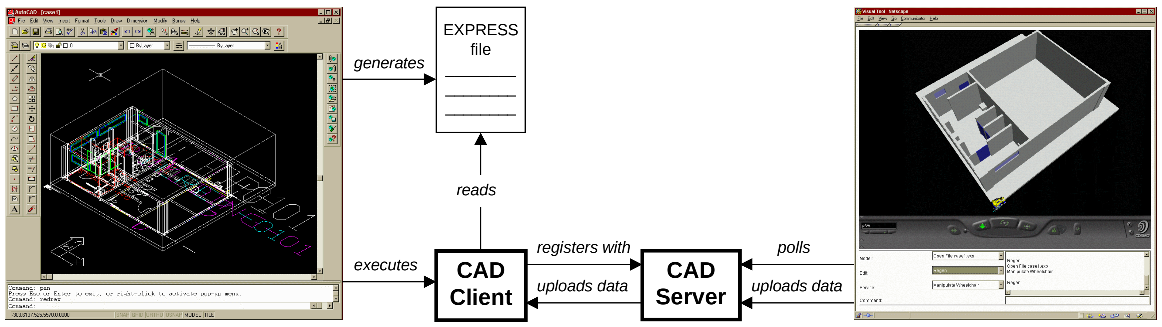


Figure 5.5: The CAD-to-VIEW interaction.

the VIEW through the web-browsing environment on the same computer system that the designer is running the CAD package). Figure 5.5 illustrates the described CAD-to-VIEW interaction.

The designer develops the facility design using AutoCAD with AutoCAD "blocks" that describe the building components adhering to the product model specifications developed in this research. To upload the facility design to the VIEW, the designer issues a command associated with an AutoLisp function that generates the developed product model EXPRESS file of the design, and the AutoLisp function executes the *CAD Client*. The *CAD Client* reads the EXPRESS file, transforms the design data to the *CAD Client*'s internal product model, and notifies the *CAD Server* that the design data is ready for uploading. The VIEW can then initiate the sequence of `gets()` commands that uploads the design data from the *CAD Client* to the VIEW via the *CAD Server*.

5.2.2 The VIEW Analysis Services

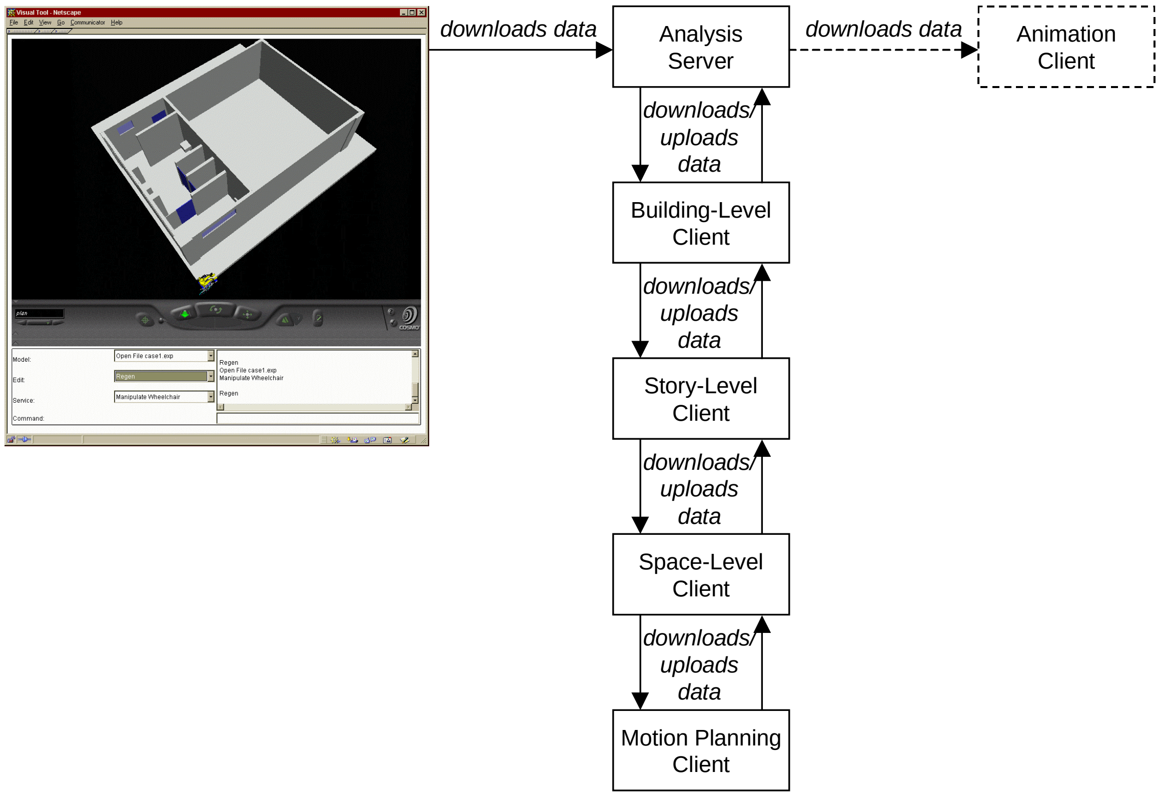
The *Code-checking Service* and the *Usability-checking Service* can be discussed together since the architecture of these composite services are similar even though the internal analysis mechanisms of the individual `Service` objects vary. The similarity confirms that the same design-intent model can be used for both the prescriptive-based code-checking formulation and the performance-based usability formulation.

Each composite service is used for a complete facility analysis or an individual route analysis. Each composite service adheres to the recursive analysis decomposition developed in Chapter 3:

- The *Building-level* `Service` object registers with the front-end *Analysis Server* `Service` object.
- The *Story-level* `Service` object registers with the *Building-level* `Service` object.
- The *Space-level* `Service` object registers with the *Story-level* `Service` object.
- Finally, the *Motion-planning* `Service` object registers with the *Space-level* `Service` object.

The *Usability Analysis Service* provides one additional service. The *Animation* `Service` object registers with the top-level *Usability Analysis Server* `Service` object and is executed to animate a particular route. Each of the `Service` objects can reside on any computer system. Ideally, the VIEW (and appropriate the *Server* `Service` object) would reside on a separate system than the other `Service` objects, and the motion-planner would reside on a machine that can handle the computationally intensive algorithms. Figure 5.6 illustrates the VIEW-to-*Analysis Service* interaction and the hierarchy of services.

The designer initiates the analysis from the VIEW, and the VIEW starts an analysis session. The VIEW downloads the facility design data to the *Analysis Server* `Service` object, and the *Analysis Server* `Service` object in turn download the facility design data to the *Building-level Client* `Service` object. Starting at the building-level, the analysis begins the decomposition process according to the architectural view of the facility. The *Building-level Client* `Service` object decomposes the facility into stories, and downloads the story-level information to the *Story-level Client* `Service` object as separate sessions.



Model: Open File case1.exp
Edit: Regen
Service: Manipulate Wheelchair
Command:

Regen
Open File case1.exp
Manipulate Wheelchair

Regen

Figure 5.6: The VIEW-to-*Analysis Service* interaction.

The facility decomposition continues down the hierarchy of registered services. When a client service receives notification via the `notification()` method from a child service that a session has completed, it uploads the new and updated facility data from that session using the `gets()` method. When all the sessions have reported back to the parent service, the parent service resolves the new and updated facility data and sends the facility data to the next registered child service in the sequence. If there are no other registered child services, the session will notify its own parent service, and that parent service will upload the revised facility design data.

The prescriptive-based analysis developed in Chapter 4 provides information about all the accessible-route-related building components in the given facility design in the form of a report. In addition, the VIEW allows the designer to interactively pick the building components that indicate the end points of an individual path within the accessible route graph and provides the visualization of this path. This visualization is especially instructive for examining possible violations to the turn-around-an-obstruction exceptions in ADAAG Provision 4.3.3.

The VIEW allows the designer to pick the initial and goal points in the facility design. The initial and goal points are constrained to the accessible route formulation developed in Chapter 3. If the user picks a building component that does not constitute an $\mathsf{R}_{\mathtt{init}}$ node, the VIEW simply informs the designer to pick another building component. Similarly, the VIEW restricts the designer's $\mathsf{R}_{\mathtt{goal}}$ choice. For example, the designer cannot choose a wall building component as the goal of the accessible route.

Once the designer has established the initial/goal pair, the VIEW sends the critical information to the prescriptive-based analysis module. The prescriptive-based analysis calculates the path and displays as much of the path as possible showing all relevant maneuvering clearance boxes and delineating the points that compose the $\mathsf{R}_{\mathtt{seg}}$ portions of the path with 36-inch spheres.

The performance-based module that interacts with the VIEW uses the same interactive procedure used by the prescriptive-based module. The designer interactively chooses the initial and goal building components, and the VIEW imposes the same restrictions on these choices. The VIEW sends the relevant information to the performance-based analysis module, and the analysis returns the path information.

In contrast to the static prescriptive-based accessible route visualization, the performance-based module generates a path animation using a wheelchair avatar. The animation GUI provides the designer with two viewpoint options of the wheelchair traversing the accessible route: an observer's and a wheelchair user's point-of-view. The designer can switch between these views to gain visual insight into the characteristics of the generated route.

When all child services have reported back up the hierarchy to the *Usability Analysis Server* `Service` object, the Server object downloads the relevant facility design data to the *Animation Client* `Service` object, and the *Animation Client* generates the accessible route animation.

5.2.3 The VIEW Joystick Service

This research includes the *Joystick Service* as there was no standard joystick interface to the web-browser environment. Incorporation of the *Joystick Service* in the design-aid framework illustrates the generality of the DOSE. The *Joystick Service* does not use the optional product model layer of the three-tiered architecture in the same manner as the aforementioned composite services. It simply sends a stream of joystick data to the VIEW. The *Joystick Service* architecture is similar to the CAD Service architecture with a single *Client* `Service` object registering with the *Server* `Service` object. The VIEW then uses the joystick data to manipulate the wheelchair in the facility.

The *Joystick Client* `Service` object resides on the same machine as the joystick device. Again, following the distributed object paradigm, the joystick can manipulate the wheelchair in the VIEW from any system on the network but, in this case, would probably be most useful residing on the system that is browsing the VIEW.

The VIEW joystick module allows the designer to interactively maneuver the wheelchair through the facility design as a wheelchair user would manipulate a motorized wheelchair. As with the VIEW performance-based module, the joystick module allows the designer to choose between an observer's and a wheelchair user's viewpoint. This module assumes independent forward and backward control of the wheelchair's major wheels. Figure 5.7 illustrates the described joystick-to-VIEW interaction.

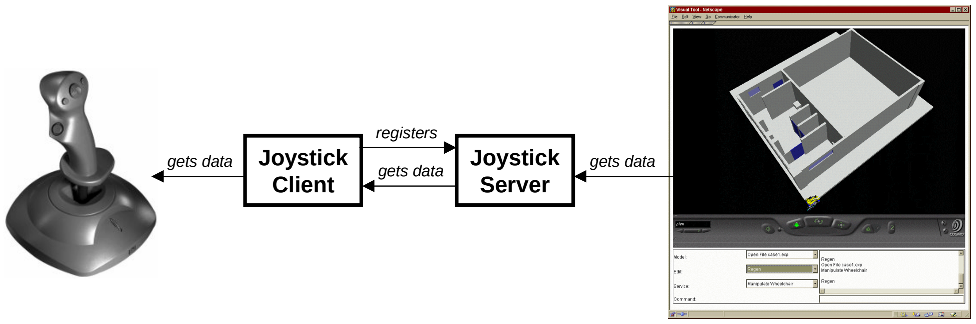


Figure 5.7: The Joystick-to-VIEW interaction.

5.3 The Design-aid Framework Implementation

This section discusses the design-aid framework application package and its implementation. Figure 5.8 illustrates the overview diagram of the design-aid framework implementation from a `Service` object point-of-view. In Figure 5.8, the `Intent` and product models are implied in the VIEW and the code-checking and usability-check services.

As shown in the figure, the VIEW and the *Server* `Service` objects reside on the same system, in this case, a Sun Ultra 1 workstation running Solaris 2.6. The *Client/Application* portion of the *CAD Service* and the *Client/Device* portion of the *Joystick Service* reside on Intel Pentium-based PC systems running Microsoft Windows NT Workstation 4.0. The computationally-intensive motion-planning and animation-generating `Service` objects reside on a Sun Ultra 10 Solaris 2.6 workstation. The prototype uses Sun's JavaIDL CORBA implementation for the DOSE thus limiting the platform choices to the Solaris operating environment and Microsoft Windows (Intel platform). Finally, the VIEW Java-VRML-CORBA interaction limits the web-browser to the Microsoft Windows Intel platform (either Microsoft Internet Explorer or the Microsoft Windows version of Netscape Communicator can display the VIEW web page).

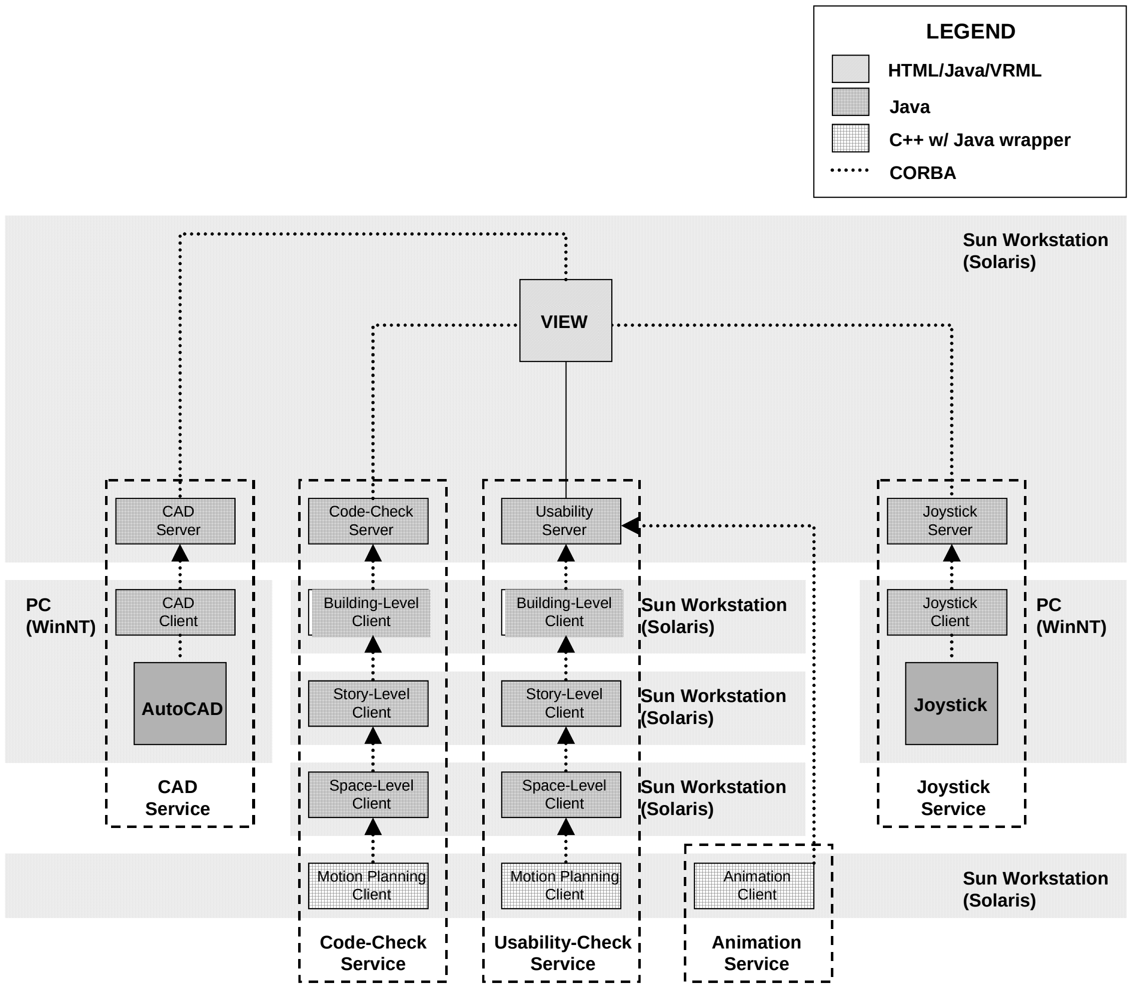


Figure 5.8: The DOSE diagram of the design-aid framework implementation.

As shown in Figure 5.8, all connections between the implemented components utilize a CORBA protocol. The DOSE communication interface shown in Figure 5.2 is in fact the Interface Definition Language (IDL) schema used to generate the DOSE computer-language-specific interfaces. This research chose the Java programming language for the following reasons:

- Java is a platform independent object-oriented programming language.

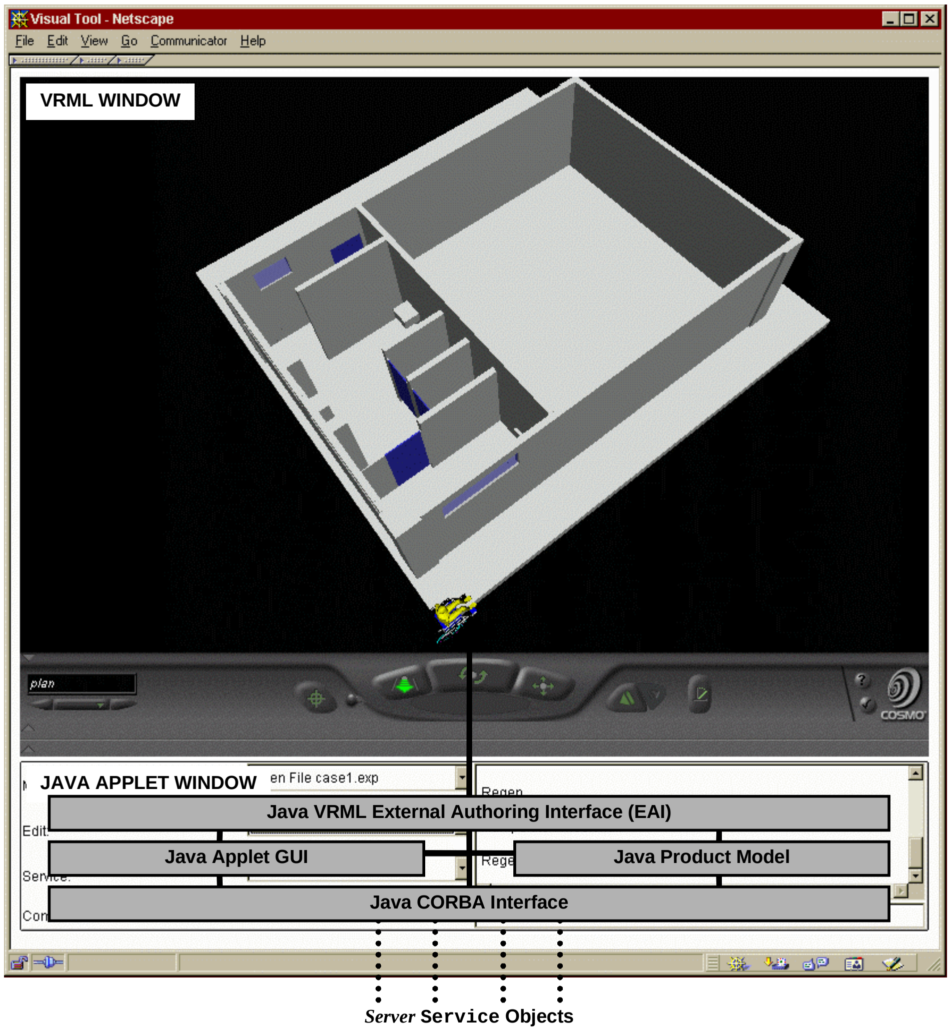


Figure 5.9: The VIEW VRML-Java-CORBA implementation.

- Java supplies straightforward interfaces to C++ libraries that the research developed for the computationally-intensive algorithms.
- The research implemented the VIEW in a web-browsing environment, and the predominant browsers provide a Java applet interface.

Similar to Java, CORBA provides a platform-independent object-oriented implementation of the distributed object environment. Sun Microsystems provides the tools to develop Java-CORBA applications as part of its Java 2 distribution.

As previously noted in this chapter, the VIEW is not a `Service` object or a composite service. Rather, two services, the *CAD Service* and the *Joystick Service* provide data to the VIEW, and the two analysis services receive data and are executed by the VIEW. Users access the VIEW via the World Wide Web, and the VIEW web page contains a VRML window, the graphics GUI, and a Java applet, the text GUI. Figure 5.9 illustrates the relationship among the different modules of the Java applet as well as the VRML window. The following sub-sections describe the Java applet modules.

5.3.1 The VRML External Authoring Interface (EAI)

The VRML External Authoring Interface (EAI) provides a message-passing mechanism between the VRML window and a Java applet. For a detailed description of the EAI, see [45]. A VRML scene consists of nodes such as shapes and attributes of shapes. These nodes populate a hierarchical structure that makes up the scene graph. The EAI allows a Java applet to access these nodes in two ways utilizing the VRML event model [46]. An *EventOut* notifies the applet of a modification of a node known as an event (such as a change in position of a shape) via the Java callback mechanism, and the applet can modify a node using an *EventIn*. In other words, an *EventOut* passes a message from the VRML scene graph to the applet, and an *EventIn* passes a message from the applet to the VRML scene graph.

The VIEW applet implements the “touchTime” *EventOut* message to inform the applet when a user clicks on a graphical object in the VRML window. The “addChildren” and “removeChildren” *EventIn* send messages to the VRML window to add and remove graphical objects from the VRML scene graph. In addition, the VIEW applet implements the “translation,” “center,” and “rotation” messages to manipulate the wheelchair through the facility design.

5.3.2 The Applet Graphical User Interface

The applet graphical user interface (GUI) allows the user to issue commands and provides feedback to the VRML window during certain command sequences from the joystick device. The VIEW command set is very limited and has been constructed to demonstrate the functionality of the design-aid framework. Commands include opening a facility design from a static file, moving and deleting building components, and executing the analysis programs.

During certain command sequences, the applet reports the information back from the VRML window. For example, when the user wants to see a specific usable wheelchair route, the applet GUI prompts the user for the initial point (a building component). The user can pick a building component, and the applet receives the “touchTime” *EventOut* message informing which shape the user picked, and the applet reports the corresponding building component. Similarly, the applet reports the building component that the user picks for the goal point and executes the route-generating analysis.

Finally, a browser contains a status line at the bottom of the browser frame. The applet can print text data on this status line, and the VIEW applet uses this mechanism to report the joystick device coordinates to the user.

5.3.3 The Product Model

The applet stores the product model that describes the facility design. The applet populates the product model either via the applet GUI when the user picks a static facility design file or when it receives the facility design information from the CAD package.

Once the applet populates the product model, the applet generates the VRML scene graph via the EAI using the relevant *EventIn* and *EventOut* message-passing mechanisms. In addition, when the user picks a building component in the VRML window, the product model module ascertains which building component maps to the shape that has been picked and reports the finding to the applet GUI.

Finally, when the user modifies the facility design, the applet modifies the product model as illustrated by the following example. If the user wants to delete a building component, the user picks the graphical shape corresponding to the building component in the VRML window. The EAI sends the "touchTime" *EventOut* message from the VRML window to the product model module, the product model resolves the building component, and the product model reports to the applet GUI. If the user confirms the deletion, the product model is modified, and the product model module sends the "removeChildren" *EventIn* message from the applet to the VRML window to delete the pertinent graphical shapes.

5.3.4 The CORBA Interface

The applet communicates with the services as illustrated in Figure 5.8 via the applet CORBA interface. The VIEW communicates with each composite *Server* `Service` object, and applet tailors its communication call for each composite service. The VIEW locates the *Server* `Service` objects using the CORBA Naming Service and can then communicate with the composite services using the methods published in the DOSE IDL file. The Java 2 platform provides a CORBA Naming Service [59], *tnameserv*, and an *idltojava* utility to generate the Java source code supporting interfaces and classes from the IDL file.

Figure 5.10 illustrates the typical CORBA implementation showing the relationship among the *Joystick Client Service* and the supporting and generated interfaces and classes. The gray boxes illustrate the interface and class that the *idltojava* utility generates from the DOSE IDL file. The `Service` interface in the IDL file directly maps

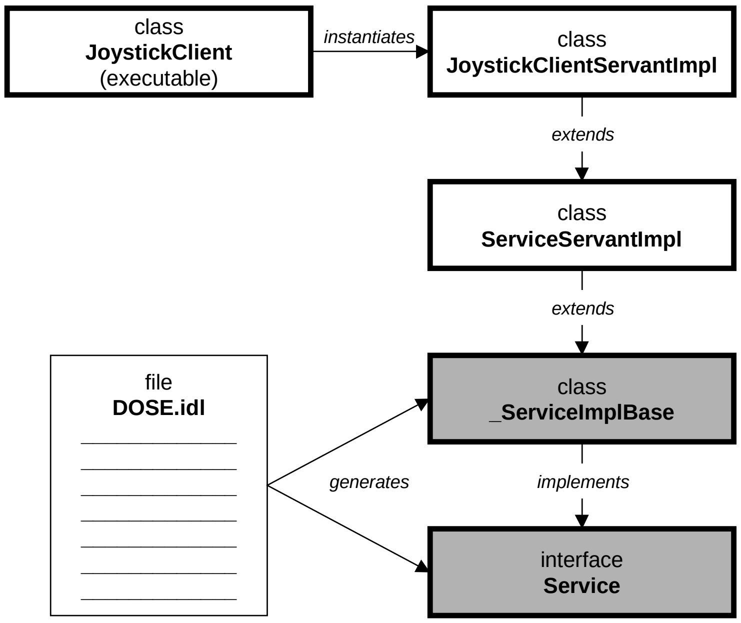


Figure 5.10: The *Joystick Client Service* implementation hierarchy.

to the generated Java `Service` interface, and the generated `_ServiceImplBase` class implements this `Service` interface. The actual `Service` object, the `JoystickClientServantImpl` class is a grandchild class of the `_ServiceImplBase` class (the `JoystickClientServantImpl` parent class, `ServiceServantImpl`, includes methods and attributes common to all DOSE services). The executable class, `JoystickClient`, instantiates the `JoystickClientServantImpl` class and, among other tasks, registers it with the *Joystick Server Service*. The research develops and implements the other service components in a similar manner.

The following paragraphs examine *CAD Service*, the *Analysis Services*, and the *Joystick Service*:

The *CAD Service* CORBA Implementation

The VIEW does not execute analysis related to the *CAD Service*. Rather, it simply receives facility design data. The *CAD Server* executable (*CADServer*) instantiates the *CAD Server* `Service` object (*CADServerServantImpl*) and registers this `Service` object with the CORBA Naming Service. The VIEW applet, in turn, can locate the *CAD Server* `Service` object via the CORBA Naming Service. The VIEW simply polls the *CAD Server* `Service` object using the `getStatus()` method. When this method returns `TRUE`, the VIEW uses the `gets()` method of the *CAD Server* `Service` object to retrieve the facility design data.

Once the applet receives the facility design data, the applet populates the product model and the VRML scene graph. If the applet already contains a facility design and the VRML window already has the corresponding scene graph, the applet first deletes the current model and VRML scene graph.

The *Analysis Services* CORBA Implementation

The research uses the same interface for both analysis services. The analysis *Server* executable instantiates the analysis *Server* `Service` object and registers this `Service` object with the CORBA Naming Service. The VIEW applet, in turn, can locate the *Server* `Service` object via the CORBA Naming Service.

The applet executes the analysis *Server* `Service` object's the `getStatus()` method to ascertain if the analysis service is busy. If the service is busy, the user can still request the analysis to be run. If the user wishes to run the analysis, the applet executes the analysis *Server* `Service` object's `puts()` method to upload the facility design data and then issues the `analyze()` command to the analysis *Server* `Service` object.

The *Joystick Service* CORBA Implementation

The VIEW does not execute any analysis related to the *Joystick Service*. Rather, it simply receives the joystick coordinate data. Note that although the *Joystick Client* is written in Java, it loads platform-specific joystick-related dynamic link libraries that use the Application Programming Interface (API) for a Microsoft Windows device.[7] Thus, while the VIEW and the *Joystick Server* can run on any platform that has a CORBA and Java Virtual Machine implementation, the *Joystick Client* can currently only run on an Intel-based Microsoft Windows computer system.

The joystick device sends a continuous stream of data. Thus, the VIEW does not use the `getStatus()` method and simply uses the `gets()` method to retrieve the coordinate data from the *Joystick Server* `Service` object. The *Joystick Server* executable *JoystickServer* instantiates the *Joystick Server* `Service` object, *JoystickServerServantImpl* and registers this `Service` object with the CORBA Naming Service. The VIEW applet, in turn, can locate the *Joystick Server* `Service` object via the CORBA Naming Service.

Once the applet receives the joystick device coordinate data, it calculates the movement of the wheelchair. The applet then sends the appropriate messages to the VRML window via the *EventIn* mechanism to move the wheelchair.

5.4 Summary

This chapter described a distributed computational environment that supports the interaction of the design-aid framework concepts and leverages the computational power realized by using the distributed object paradigm. This research has optimized the DOSE for the manipulation of design data and the decomposition of this data into analysis-specific views. However, as illustrated by the joystick device-driver implementation, the research has kept the environment general enough to support a wide range of services.

[7] I would like to thank Taisuke Fukuno, whom I have never met, but whose joystick device driver source code I obtained from the World Wide Web and adapted for the *Joystick Client Service* developed in this thesis. He has made the source code at http://sariel.miyako.co.jp/~uni/mmpackage.html.

The DOSE framework is used to realize the Visual Interactive Environment/Workbench (VIEW) that provides the designer with three types of visual tools. The VIEW provides a means to visualize the output of the accessible route analysis methods described in Chapter 4. With the prescriptive-based analysis, the VIEW allows the user to examine a user-chosen static code-compliant accessible route. The VIEW performance-based module animates a user-chosen wheelchair path and allows the user to examine this path from either an observer's or a wheelchair user's viewpoint.

This chapter described the implementation of the design-aid framework. The framework utilizes a complementary distributed object environment and object-oriented programming language, CORBA and Java, and both the product model for the facility design and the analysis model utilize object-oriented paradigms. In addition, the user interface, the VIEW, uses the ubiquitous web-browsing environment as well as a widely accepted complementary graphics environment, VRML.

CORBA and Java provide a platform-independent environment, a desirable characteristic for the distribution of processes across heterogeneous systems on a network. The Java programming language also provides interfaces to other programming language libraries such as C++ that was used to develop the computationally-intensive algorithms. Java's thread mechanism allows the framework to take advantage of multiprocessing systems without changing the code to accommodate the hardware platform, and `Service` objects can take advantage of the thread feature to execute concurrent sessions.

Chapter 6

Test Case Example

This chapter presents a case study of a Stanford University building to test the disabled access design-aid framework and the performance-based methods developed in this research. Recall that the performance-based approach is able to determine the usability of a facility, and usability does not necessarily equate to code-compliance. The automated analysis of the building, the Career Development Center (CDC), shows several inaccessible areas. These results were compared to the actual usability of the facility with the help of a cooperating wheelchair user, Joe Cavanaugh, a Stanford University student. In certain situations, the wheelchair user validated the initial generated analysis. In two situations, however, the performance-based formulation was altered to match the actual accessibility of certain facilities.

The chapter is organized in the following manner:

- Section 6.1 provides a description of the CDC.
- Section 6.2 describes the generated analysis and the modifications made to the facility design and the performance-based methods.
- Section 6.3 summarizes the results of the analysis of the modified facility design.

- Section 6.4 summarizes the chapter including the recommended changes and suggestions for the current and future CDC buildings.

6.1 The Career Development Center

The Career Development Center (CDC) is centrally located on the Stanford University campus on White Memorial Plaza between the Stanford Bookstore and the Clock Tower. Figure 6.1 shows the entrance side of the building, and Figure 6.2 shows a close-up of the entrance. The CDC offers career-related guidance and resources to Stanford University students and alumni. These services include written research material, computers, the availability of guidance counselors, and potential-employer/student interviews.

The plan in Figure 6.3 shows the current usage of the building. The entrance leads users directly to the library/research area, and there are offices to the left and to the right of the space. The bathrooms and the interview rooms are in what is now the central core of the building. The University has slated the current CDC building for demolition in the next couple of years. The CDC will be relocated to a new building (yet to be constructed) in another part of the campus.

The CDC was built in the late 1930s and at the time of the 1967 construction documents, the facility was known as the Old Bookstore and was to be converted into a placement service center. Renovation of the building included the addition of the still-existing eleven interview rooms. The 1985 as-built construction documents delineate the expansion plans of the facility as it is in its current state. These documents indicate that the building was already being used as the Career Planning and Placement Center (CPPC), the former name of the CDC. As shown in the Existing Conditions and Demolition Plan Figure 6.4, plans included the renovation of the existing Men’s and Women’s bathroom and, as shown in the New Construction Plan in Figure 6.5, the renovation addressed accessibility issues for the bathrooms.

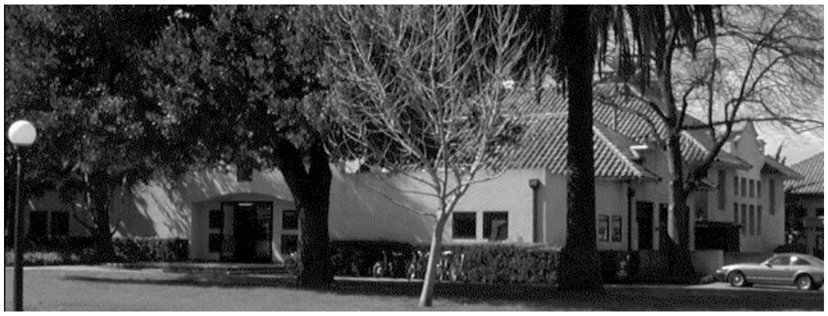


Figure 6.1: The Stanford University Career Development Center (CDC).

Figure 6.2: CDC entrance close-up.

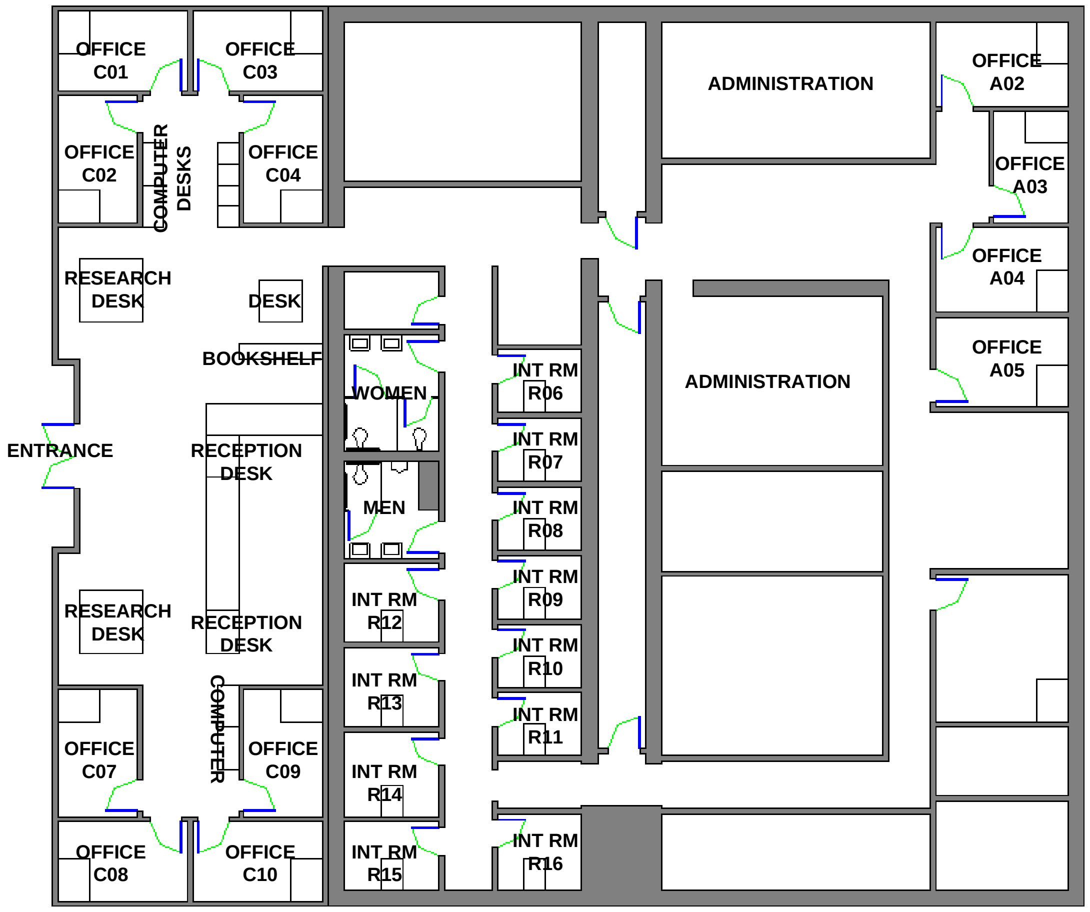


Figure 6.3: CDC plan view.

As shown in Figure 6.4 and Figure 6.5, the Men’s bathroom renovation called for:

- removal of the stall partitions
- addition of grab bars around the existing toilet
- replacement of the other existing fixtures with accessible fixtures

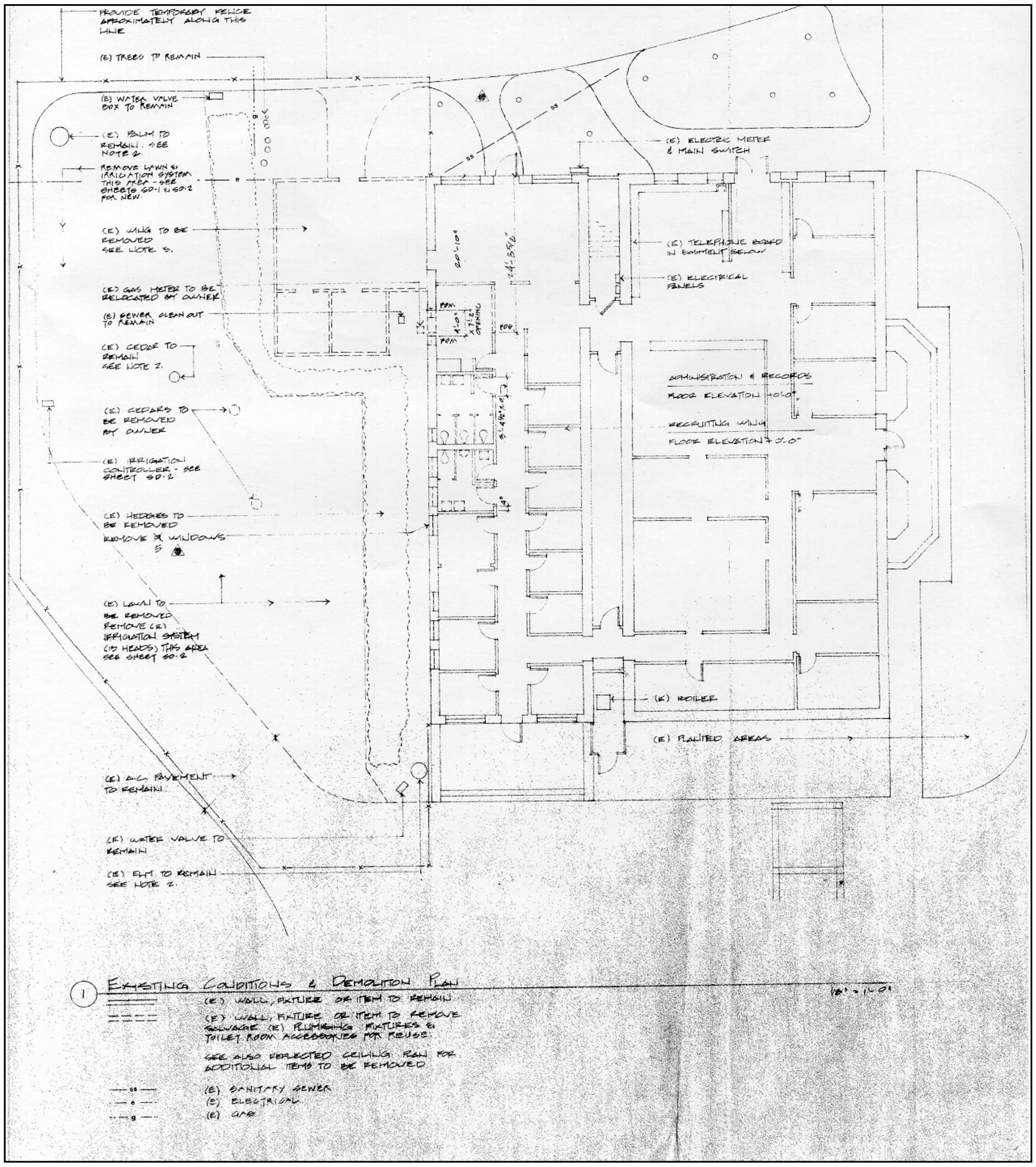

PROVIDE TEMPORARY FENCE APPROXIMATELY ALONG THIS LINE
(E) TREES TO REMAIN
(E) WATER VALVE BOX TO REMAIN
(E) PALM TO REMAIN. SEE NOTE 2.
REMOVE LAWN & IRRIGATION SYSTEM THIS AREA - SEE SHEETS SD-1 & SD-2 FOR NEW.
(E) WING TO BE REMOVED SEE NOTE 5.
(E) GAS METER TO BE RELOCATED BY OWNER
(E) SEWER CLEAN OUT TO REMAIN
(E) CEDAR TO REMAIN SEE NOTE 2.
(E) CEDARS TO BE REMOVED BY OWNER
(E) IRRIGATION CONTROLLER - SEE SHEET SD-2
(E) HEDGES TO BE REMOVED
REMOVE (E) WINDOWS 5
(E) LAWN TO BE REMOVED REMOVE (E) IRRIGATION SYSTEM (15 HEADS) THIS AREA SEE SHEET SD-2
(E) A.C. PAVEMENT TO REMAIN
(E) WATER VALVE TO REMAIN
(E) ELM TO REMAIN SEE NOTE 2.
(E) ELECTRIC METER & MAIN SWITCH
(E) TELEPHONE BOARD IN BASEMENT BELOW
(E) ELECTRICAL PANELS
ADMINISTRATION & RECORDS
FLOOR ELEVATION 40'-0"
RECRUITING WING
FLOOR ELEVATION ± 0'-0"
(E) BOILER
(E) PLANTED AREAS
20'-10"
24'-3 5/8"
1
EXISTING CONDITIONS & DEMOLITION PLAN
1/8" = 1'-0"
(E) WALL, FIXTURE OR ITEM TO REMAIN
(E) WALL, FIXTURE OR ITEM TO REMOVE SALVAGE (E) PLUMBING FIXTURES & TOILET ROOM ACCESSORIES FOR REUSE.
SEE ALSO REFLECTED CEILING PLAN FOR ADDITIONAL ITEMS TO BE REMOVED
(E) SANITARY SEWER
(E) ELECTRICAL
(E) GAS

Figure 6.4: CDC 1985 as-builts, existing conditions and demolition plan.

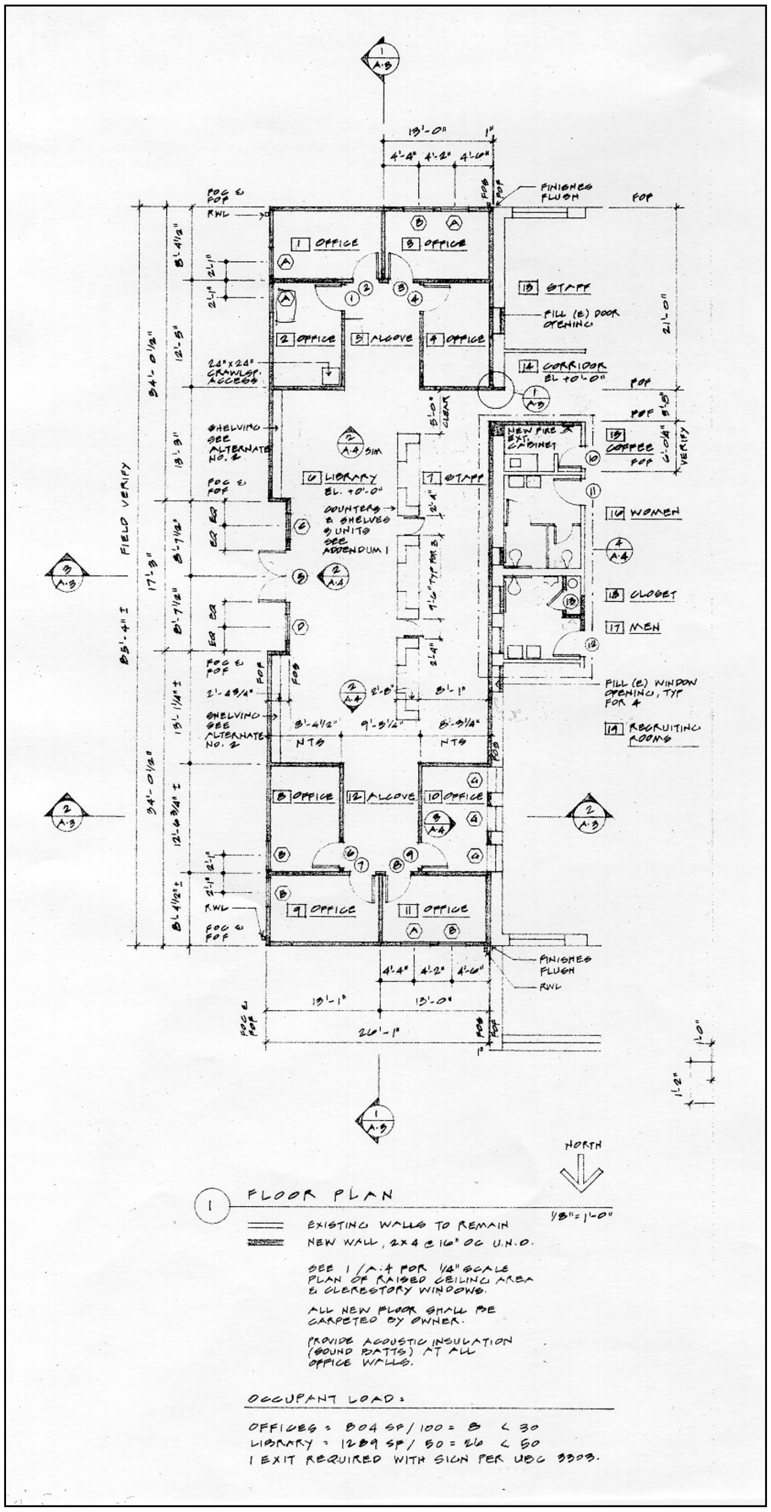

FLOOR PLAN
1/8"=1'-0"
EXISTING WALLS TO REMAIN
NEW WALL, 2x4 @ 16" OC U.N.O.
SEE 1/A.4 FOR 1/4" SCALE PLAN OF RAISED CEILING AREA & CLERESTORY WINDOWS.
ALL NEW FLOOR SHALL BE CARPETED BY OWNER.
PROVIDE ACOUSTIC INSULATION (SOUND BATTS) AT ALL OFFICE WALLS.
OCCUPANT LOAD:
OFFICES = 804 SF/100 = 8 < 30
LIBRARY = 1289 SF/50 = 26 < 50
1 EXIT REQUIRED WITH SIGN PER UBC 3303.
NORTH
OFFICE
ALCOVE
LIBRARY
STAFF
CORRIDOR
COFFEE
WOMEN
CLOSET
MEN
RECRUITING ROOMS

Figure 6.5: CDC 1985 as-builts, new construction plan.

The Women’s bathroom renovation called for:

- removal of the middle toilet
- addition of grab bars around the existing toilet designated for accessible use

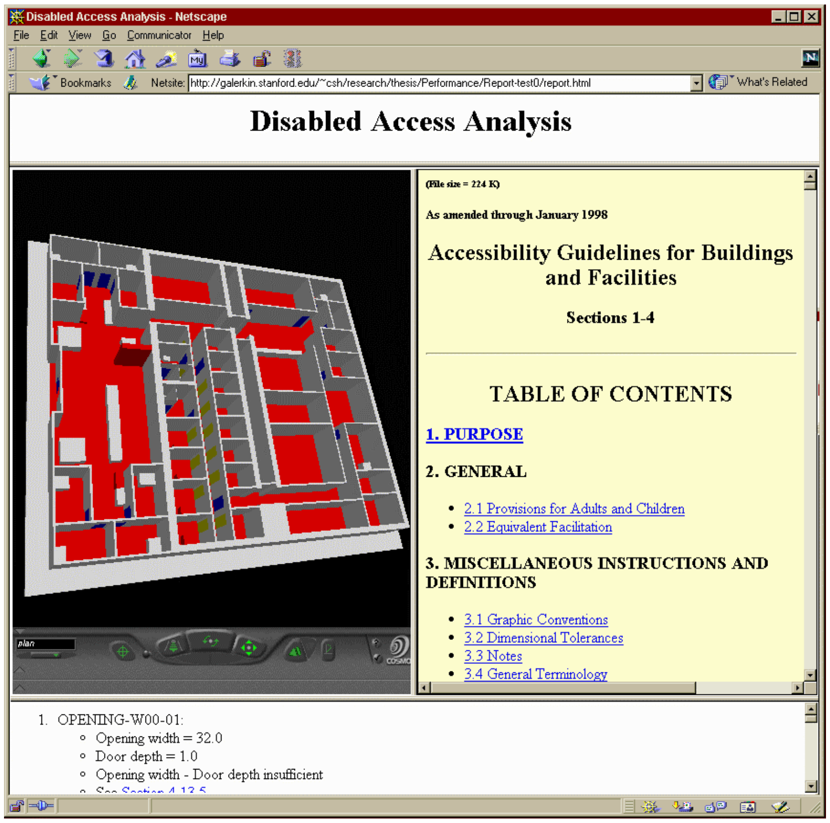
In addition, as shown in the New Construction Plan in Figure 6.5, the architects designed the other new facilities (the guidance counselor offices and the library and research area) to be accessible providing:

- adequate clearances to open doors
- adequate maneuvering clearances in the corridors
- addition of new stall partitions provide wheelchair use to the designated toilet
- replacement of the other existing fixtures with accessible fixtures

6.2 The Analysis

This section presents the automated analysis of the CDC. Note that even though the analysis uses the developed performance-based methods, the comments associated with inaccessible building components have links to the prescriptive provisions of the ADAAG document as an informative guideline. The ADAAG addresses new construction, modification to existing structures, and historical buildings under the 1990 Americans with Disabilities Act. The last major modifications to the CDC were carried out before 1990 so, technically, the CDC does not fall under any of these three categories. Still, it is useful to analyze the CDC against the ADAAG document, and the comparision examines the CDC as a newly constructed facility.

Figure 6.6 shows the generated analysis report with a view of the modeled CDC. The darker color (red in the generated VRML frame) in relation to the wall color (white in the


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html
What's Related
plan

Disabled Access Analysis

(File size = 224 K)

As amended through January 1998

Accessibility Guidelines for Buildings and Facilities

Sections 1-4

TABLE OF CONTENTS

1. PURPOSE

2. GENERAL

- 2.1 Provisions for Adults and Children
- 2.2 Equivalent Facilitation

3. MISCELLANEOUS INSTRUCTIONS AND DEFINITIONS

- 3.1 Graphic Conventions
- 3.2 Dimensional Tolerances
- 3.3 Notes
- 3.4 General Terminology

1. OPENING-W00-01:
   - Opening width = 32.0
   - Door depth = 1.0
   - Opening width - Door depth insufficient

Figure 6.6: The initial CDC disabled access analysis report: the facility is inaccessible.

generated VRML frame) indicates an inaccessible component. Specifically, the building components that are inaccessible are red, and the floor of the entire facility has also been set to red indicating the whole facility indicating that critical components of the facility are inaccessible. The user can click on the inaccessible building component, and the associated comment appears in the bottom frame. These comments have links to ADAAG provisions that appear in the right frame.

Comparing the analysis results with an individual wheelchair user's ability to use the facility represents strictly a qualitative test as there are many different levels of disability, but such a comparison still provides insight into the analysis. Joe Cavanaugh, a wheelchair user, compared his ability to negotiate the facility with the analysis results.

Mr. Cavanaugh considers himself to have better-than-average mobility for a wheelchair user. Indeed, he is able to comfortably negotiate spaces that are far more restrictive than the ADAAG permits. His better-than-average mobility is due to his physical arm strength which allows him to use a smaller, more efficient wheelchair as well as to his ability to quickly move backwards and forwards several times, a sequence of motions that violates the ADAAG accessible route parameters.

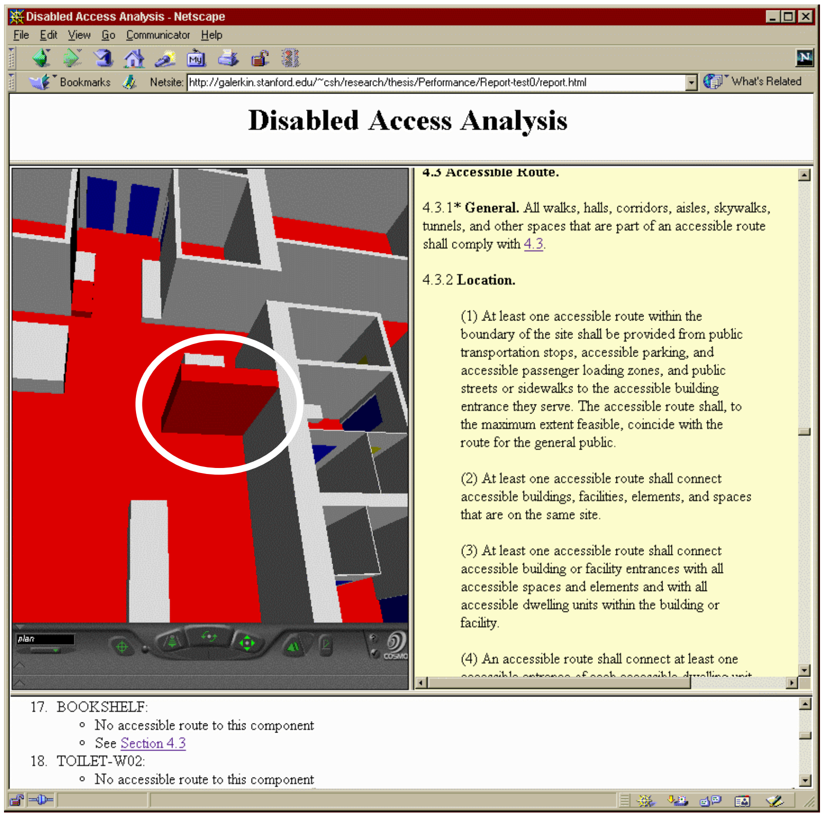
The discrepancy between Mr. Cavanaugh's mobility and the usage parameters set forth by the ADAAG illustrates the difficulty in providing a performance-based access code that encompasses all wheelchair users and provides guidelines for usage and comfort. However, adjustments to the performance-based analysis tailored to a group of similar users might provide better insight to the actual accessibility of a facility for these users than a prescriptive-based analysis.

The following discusses the results of four facilities, namely a set of bookshelves, the Women's Bathroom, the Men's Bathroom, and the Interview Rooms.

<u>Bookshelves</u>

The analysis reports that there is no accessible route to the bookshelf in the library/research area as illustrated in Figure 6.7. The viewpoint shows the back of the bookshelf, and, according to the analysis, the desk in front of the bookshelf blocks access to this building component.

The analysis assumes a front approach to the bookshelf. However, as shown in Figure 6.8, the wheelchair user has comfortable access to a bookshelf using a side approach. Indeed, the ADAAG makes provisions for both front and side approach to storage facilities including bookshelves, and the original exclusion of this approach was an oversight in the developed motion-planning goal point description for this building component. From the ADAAG:


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html
What's Related
plan
COSMO

Disabled Access Analysis

4.3 Accessible Route.

4.3.1* **General.** All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.

4.3.2 **Location.**

(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.

(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.

(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

(4) An accessible route shall connect at least one

17. BOOKSHELF:
    - No accessible route to this component
    - See Section 4.3
18. TOILET-W02:
    - No accessible route to this component

Figure 6.7: The disabled access analysis report: the bookshelf is inaccessible.

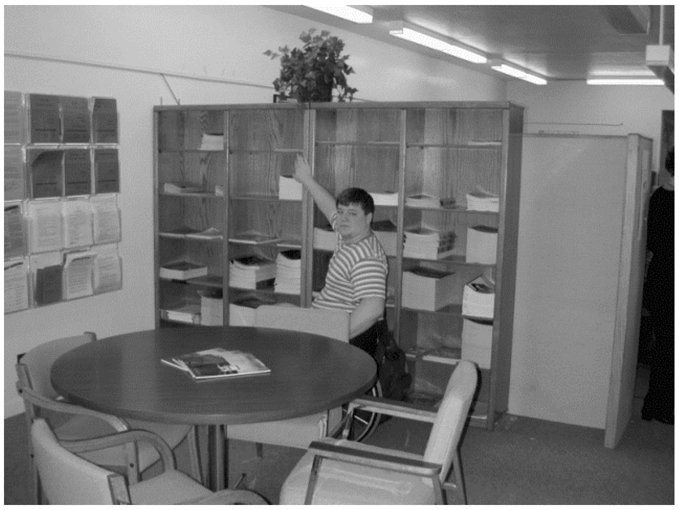


Figure 6.8: Wheelchair user access to bookshelf (desk size and position unmodified).

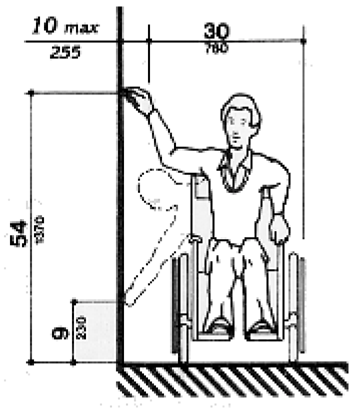


Figure 6.9: The ADAAG graphic illustrating side approach reach parameters.

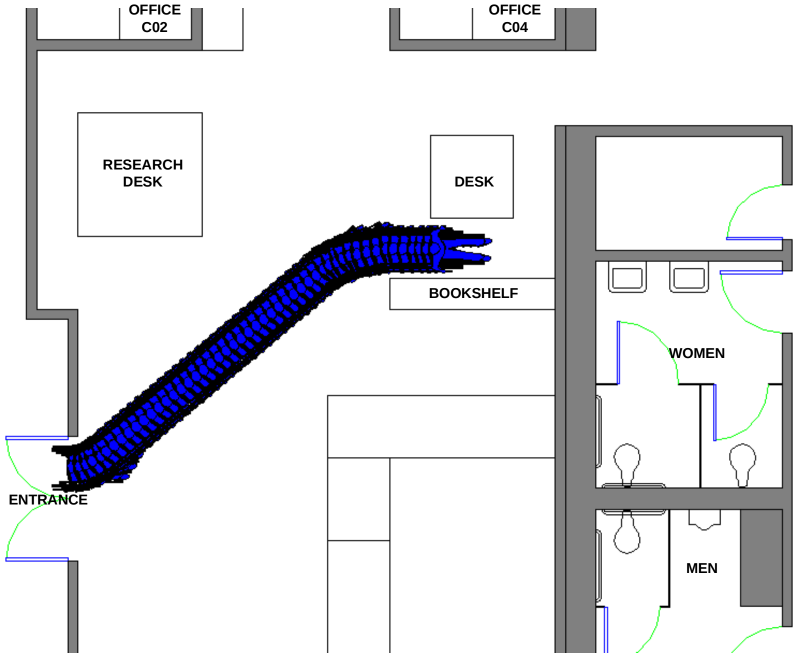


Figure 6.10: Wheelchair route to bookshelf enabling the side-approach method.

4.25 Storage.

4.25.1 General. Fixed storage facilities such as cabinets, shelves, closets, and drawers required to be accessible by 4.1 shall comply with 4.25.

4.25.2 Clear Floor Space. A clear floor space at least 30 in by 48 in (760 mm by 1220 mm) complying with 4.2.4 that allows either a forward or parallel approach by a person using a wheelchair shall be provided at accessible storage facilities.

In addition, as indicated by the wheelchair user's extended arm in Figure 6.8, bookshelves should be restricted by reach parameters as suggested by the ADAAG graphic shown in Figure 6.9.

Figure 6.10 shows the motion-plan generated accessible route to the bookshelf enabling the side-approach goal for storage-related building components.

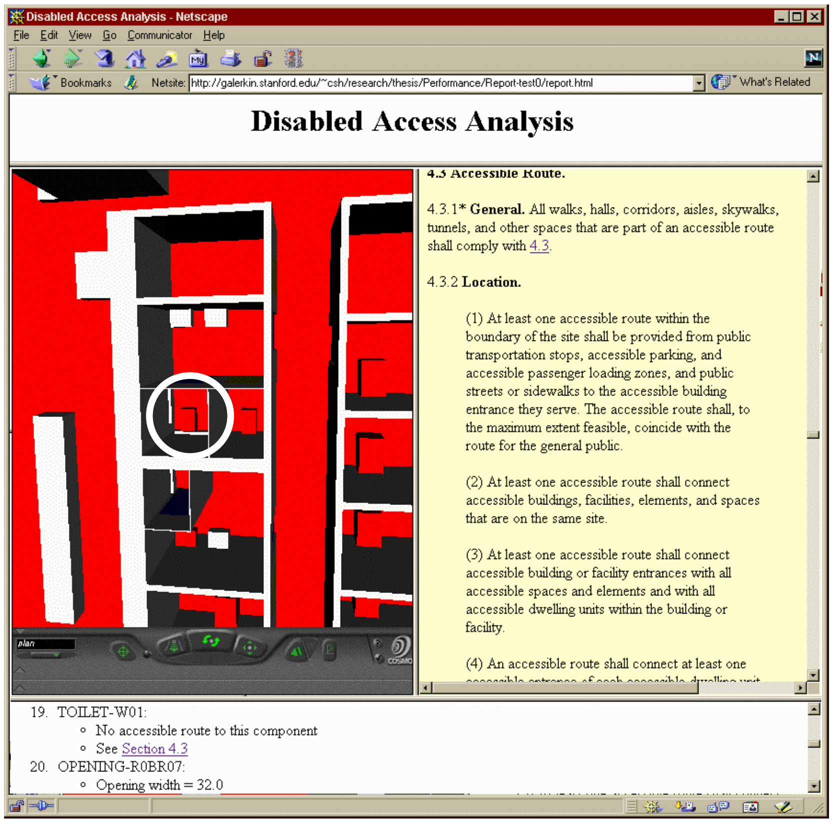
<u>Women's Bathroom</u>

The analysis reports that there is no accessible route to the accessible toilet in the Women's Bathroom as illustrated in Figure 6.11. The performance-based parameters used the ADAAG toilet stall clearance areas as a guideline for the motion-planning goal parameters for toilets, and the accessible stall violates these guidelines. Since there are no accessible toilets, the bathroom is not considered to be accessible, and in turn, the whole facility is deemed inaccessible.

However, as shown in Figure 6.12, the wheelchair user has comfortable access to this toilet. The user in fact has easily positioned himself for side transfer, a position that is more difficult to achieve than a diagonal transfer for this given stall.

By slightly adjusting the toilet goal parameters, the analysis now shows that the toilet is accessible. The modified goal position and orientation of the wheelchair for diagonal transfer is shown in Figure 6.13.

The generated path from the bathroom entrance to the toilet with the modified goal parameters is shown in Figure 6.14. Note that the path is not continuous between adjacent spaces (the bathroom entry area and the stall), since the motion planner


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html
What's Related
plan

Disabled Access Analysis

4.3 Accessible Route.

4.3.1* **General.** All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.

4.3.2 **Location.**

(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.

(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.

(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

(4) An accessible route shall connect at least one

19. TOILET-W01:
    - No accessible route to this component
    - See Section 4.3
20. OPENING-R0BR07:
    - Opening width = 32.0

Figure 6.11: The disabled access analysis report: the women's toilet is inaccessible.

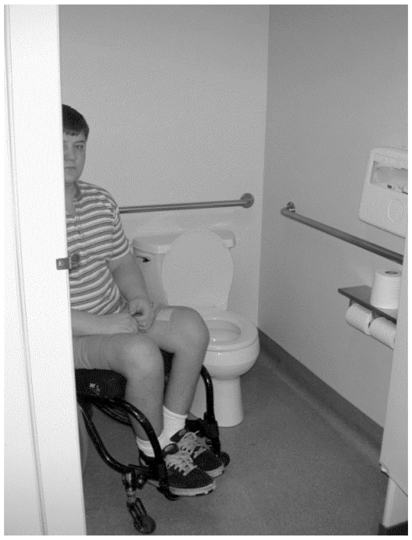


Figure 6.12: Wheelchair user access to the women's toilet.

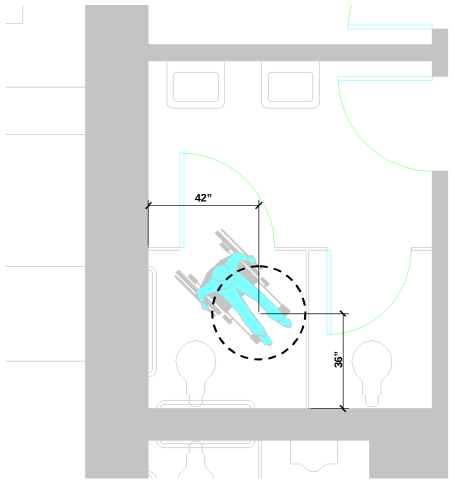


Figure 6.13: Modified goal parameters for women's toilet.

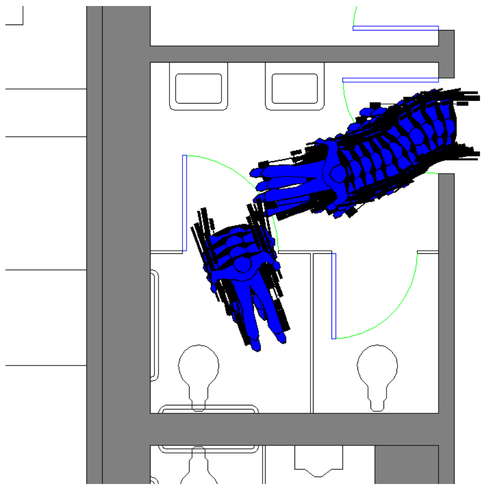


Figure 6.14: Wheelchair path to the women's toilet using the modified goal parameters.

generates a path from the initial point of one door opening to goal point of another door opening. As stated in 4.2.2 research has developed the goal point/initial point pair of an opening such that a usable wheelchair path exists for the wheelchair user to move from the goal point to the initial point of an opening.

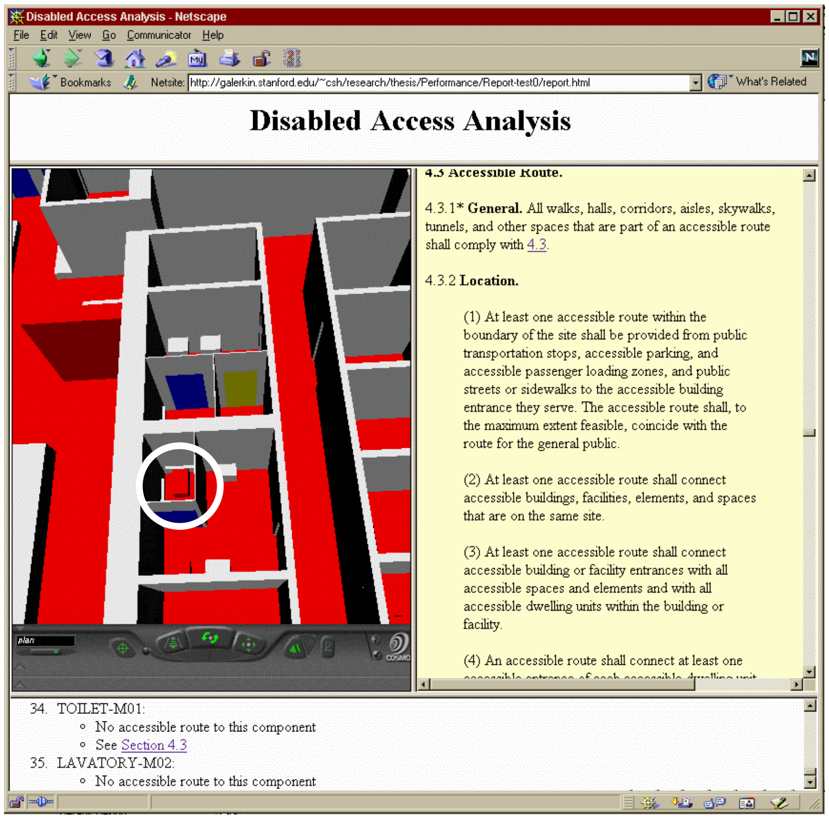
The goal parameters of the toilet had been prescribed in accordance to an ADAAG approach parameter. This example illustrates that even this parameter is too restrictive, and that the performance-based method should be more flexible by describing a range of possible goal areas and orientations.

<u>Men's Bathroom</u>

The analysis reports that there is no accessible route to the accessible toilet in the Men's Bathroom as illustrated in Figure 6.15. As with the analysis of the Women's bathroom, since there are no accessible toilets, the bathroom is not considered to be accessible, and in turn, the whole facility is deemed inaccessible.

Figure 6.16 confirms the inaccessibility of the toilet. Here, the wheelchair user is not able to pass through the stall doorway. The original plans for this toilet did not include the partition walls as shown in Figure 6.5. These walls must have been added to ensure privacy for the toilet user. Ironically, the addition of these walls has made the toilet inaccessible.

Without the partition walls, the motion planner can generate an accessible route to the stall. Figure 6.17 shows the path from the entrance to the accessible toilet with these partition walls removed. Again, the path is discontinuous between adjacent spaces.


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html
What's Related
plan

Disabled Access Analysis

4.3 Accessible Route.

4.3.1* **General.** All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.

4.3.2 **Location.**

(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.

(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.

(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

(4) An accessible route shall connect at least one

34. TOILET-M01:
   - No accessible route to this component
   - See Section 4.3

35. LAVATORY-M02:
   - No accessible route to this component

Figure 6.15: The disabled access analysis report: men’s toilet is inaccessible.

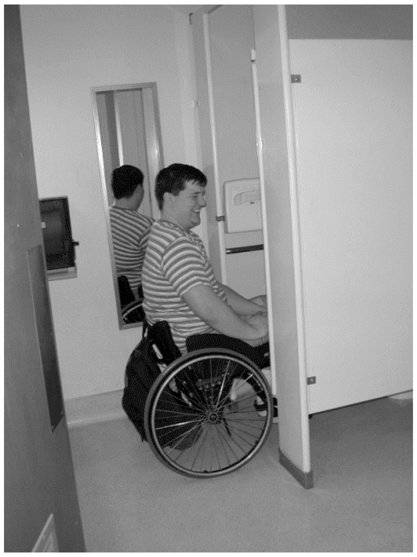


Figure 6.16: Wheelchair user unable to pass through men’s toilet stall door.

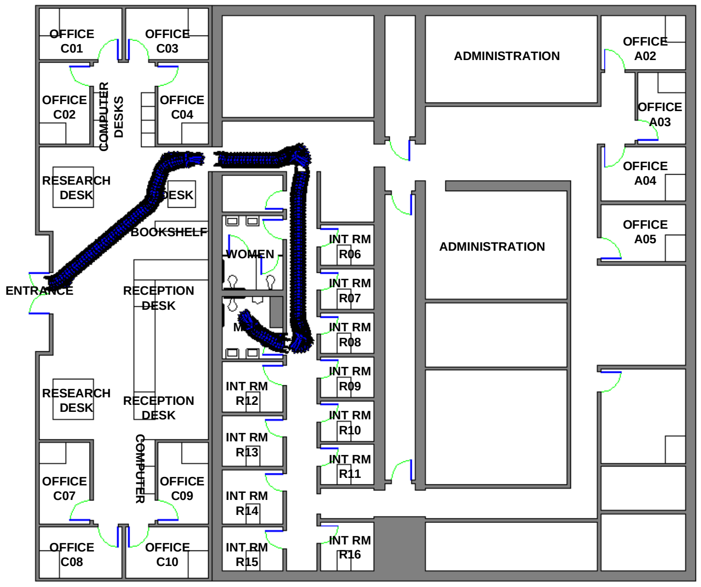
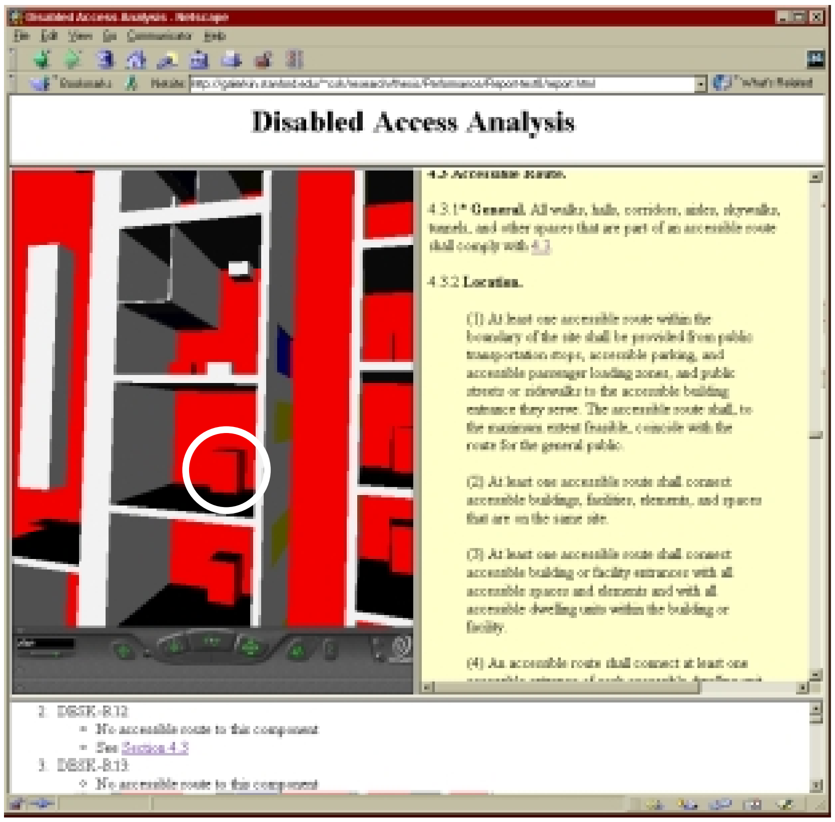


Figure 6.17: Wheelchair route to men's toilet (stall partitions removed).

Interview Rooms

The automated analysis reports that there are no accessible routes to any of the Interview Room desks. Figure 6.18 highlights one of the desks. The motion planner does not even evaluate the existence of an accessible route from an Interview Room door to a desk because the doorway widths do not comply with the ADAAG. Recall that this comfort level determination has been taken directly from the ADAAG in the construction of accessible routes.

Figure 6.19 confirms the narrowness of the doorway. Even with his wheelchair (the chair is smaller than the chair used in the ADAAG provisions), the wheelchair user has difficulty negotiating the doorway. Once through the doorway, as shown in Figure 6.20, the closeness of the interview desk to the door makes it difficult but not impossible to position the wheelchair at the interview desk provided that an interview chair is not obstructing the path.


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite:
What's Related

Disabled Access Analysis

4.3 Accessible Route.

4.3.1* **General.** All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.

4.3.2 **Location.**

(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.

(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.

(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

(4) An accessible route shall connect at least one

2. DESK-R12
   - No accessible route to this component
   - See Section 4.3
3. DESK-R13
   - No accessible route to this component

Figure 6.18: The disabled access analysis report: interview room desks are inaccessible.

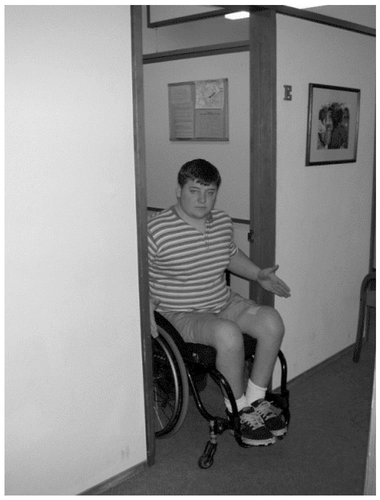


Figure 6.19: Wheelchair user in an interview room doorway.

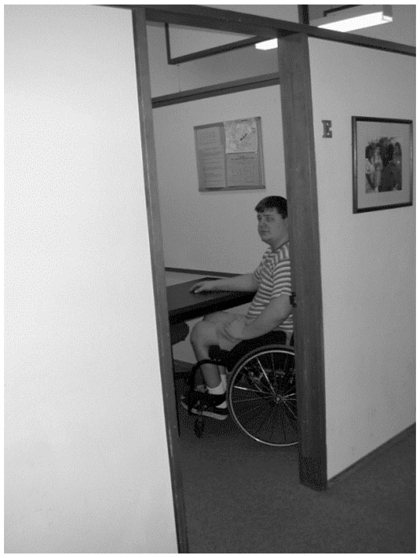


Figure 6.20: Wheelchair user at an interview desk (after numerous backups).

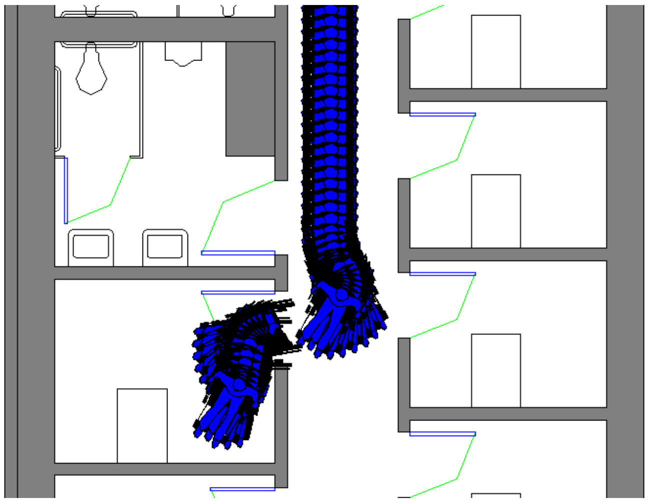


Figure 6.21: Wheelchair route to an interview desk (doorway width and desk position modified).

After modifying an Interview Room in the facility by widening the doorway and placing the interview desk further from the door, the motion planner can generate an accessible route to the interview desk. The generated path is shown in Figure 6.21.

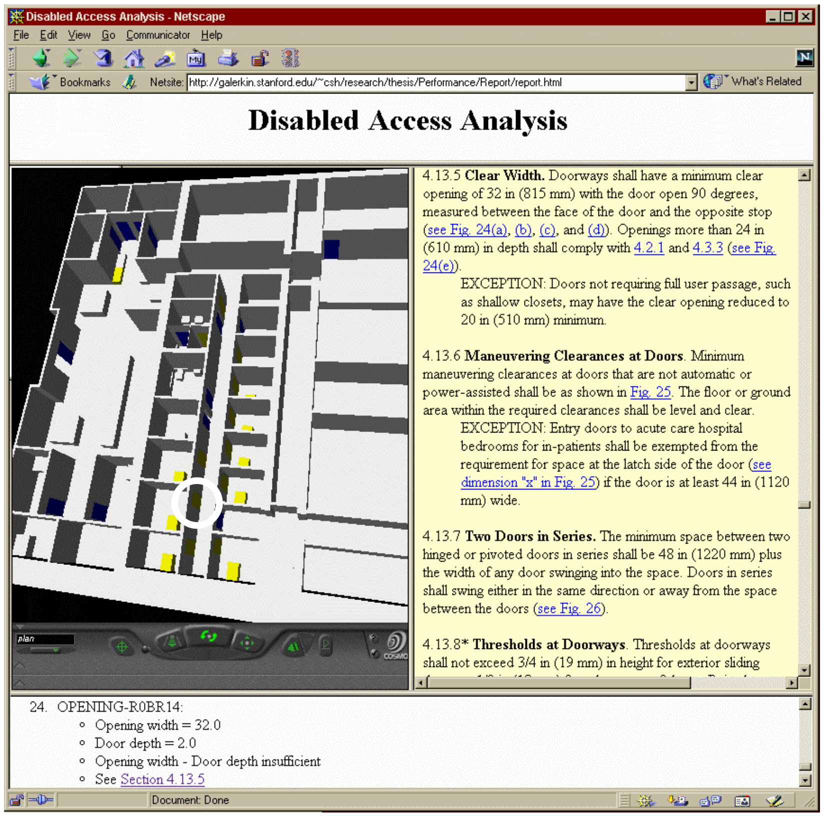
In general, having some fraction of a set of identically functioning spaces be accessible is acceptable under the equivalent access to facilities intent. In this case, upon initial inspection, providing one accessible Interview Room satisfies this intent. However, as noted by CDC staff member Betsy Cuisinot, it is difficult to match disabled interviewees with interviewers in an accessible room beforehand, and it is unfair and at times impossible to have these already nervous candidates wait for an accessible room to be made available. Thus, following the intent of the code, the interview process is a special case in which to truly satisfy accessibility, all interview rooms should be made usable for wheelchair users.

6.3 The Analysis of the Modified Design

Figure 6.22 shows the report generated for the analysis of the modified design. The floor of the facility is no longer red (a dark color) indicating that the performance-based analysis of the CDC (with the modifications discussed in the previous section) finds that the facility is now accessible:

- The motion-planning parameters have been adjusted to reflect the usability of the bookshelf and the women's bathroom.
- The stall partitions in the men's bathroom have been removed.
- One interview room has been modified (the door opening has been widened and the interview desk has been moved).

This particular analysis assumes that one accessible Interview Room is sufficient, and there is only one accessible Interview Room in the design since only one of the Interview Room doorways was widened in this modified design. The figure shows the comments associated with a doorway that was not widened, and thus the associated interview desk does not have an accessible route.


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report/report.html
What's Related
plan
Document: Done

Disabled Access Analysis

4.13.5 **Clear Width.** Doorways shall have a minimum clear opening of 32 in (815 mm) with the door open 90 degrees, measured between the face of the door and the opposite stop (see Fig. 24(a), (b), (c), and (d)). Openings more than 24 in (610 mm) in depth shall comply with 4.2.1 and 4.3.3 (see Fig. 24(e)).

EXCEPTION: Doors not requiring full user passage, such as shallow closets, may have the clear opening reduced to 20 in (510 mm) minimum.

4.13.6 **Maneuvering Clearances at Doors**. Minimum maneuvering clearances at doors that are not automatic or power-assisted shall be as shown in Fig. 25. The floor or ground area within the required clearances shall be level and clear.

EXCEPTION: Entry doors to acute care hospital bedrooms for in-patients shall be exempted from the requirement for space at the latch side of the door (see dimension "x" in Fig. 25) if the door is at least 44 in (1120 mm) wide.

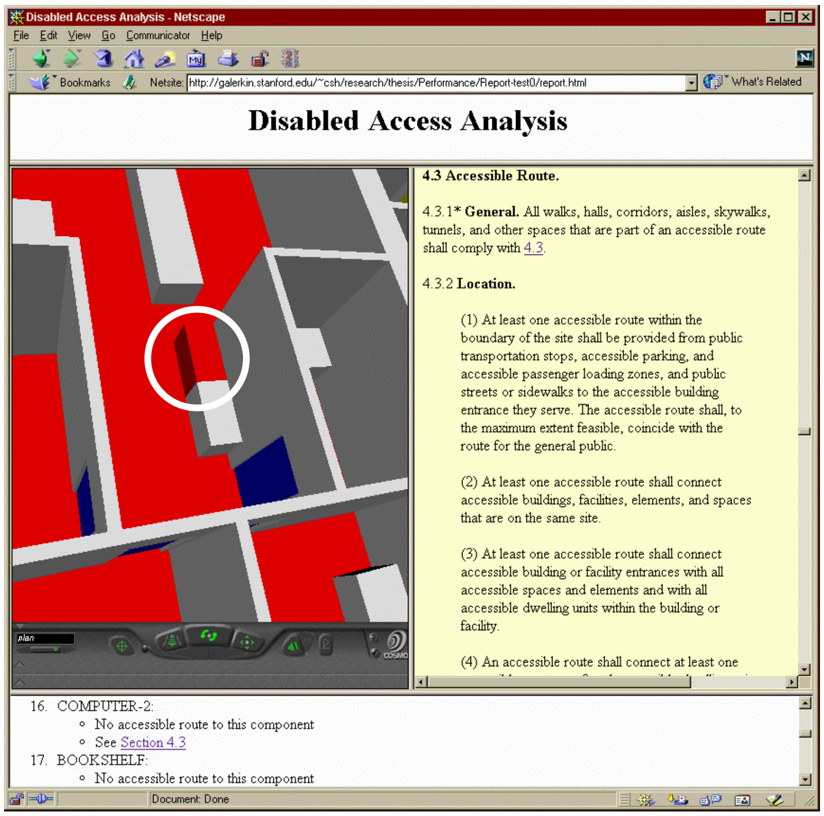
4.13.7 **Two Doors in Series.** The minimum space between two hinged or pivoted doors in series shall be 48 in (1220 mm) plus the width of any door swinging into the space. Doors in series shall swing either in the same direction or away from the space between the doors (see Fig. 26).

4.13.8* **Thresholds at Doorways**. Thresholds at doorways shall not exceed 3/4 in (19 mm) in height for exterior sliding

24. OPENING-R0BR14:
   - Opening width = 32.0
   - Door depth = 2.0
   - Opening width - Door depth insufficient
   - See Section 4.13.5

Figure 6.22: The disabled access analysis report for the modified design.

The analysis has two errors. The analysis reports that two of the four computer desks in the library/research area do not have accessible routes. Figure 6.23 shows the analysis of one of these desks. Visual inspection indicates that these desks do lie on an accessible route, and Figure 6.24 shows the generated path to an adjacent desk. The error can be traced to the structure of the facility model. The analysis program has the limitation allowing only rectangular spaces that are separated by walls and openings. The offending computer desk does not lie completely in a corridor space as shown in Figure 6.25 and is therefore not considered in the analysis of the space. To circumvent this problem, the analysis must allow a space to be constructed using a collection of different shapes.


Disabled Access Analysis - Netscape
File Edit View Go Communicator Help
Bookmarks
Netsite: http://galerkin.stanford.edu/~csh/research/thesis/Performance/Report-test0/report.html
What's Related
plan
Document: Done

Disabled Access Analysis

4.3 Accessible Route.

4.3.1* **General.** All walks, halls, corridors, aisles, skywalks, tunnels, and other spaces that are part of an accessible route shall comply with 4.3.

4.3.2 **Location.**

(1) At least one accessible route within the boundary of the site shall be provided from public transportation stops, accessible parking, and accessible passenger loading zones, and public streets or sidewalks to the accessible building entrance they serve. The accessible route shall, to the maximum extent feasible, coincide with the route for the general public.

(2) At least one accessible route shall connect accessible buildings, facilities, elements, and spaces that are on the same site.

(3) At least one accessible route shall connect accessible building or facility entrances with all accessible spaces and elements and with all accessible dwelling units within the building or facility.

(4) An accessible route shall connect at least one

16. COMPUTER-2:
    - No accessible route to this component
    - See Section 4.3
17. BOOKSHELF:
    - No accessible route to this component

Figure 6.23: The disabled access analysis report: computer desk is inaccessible.

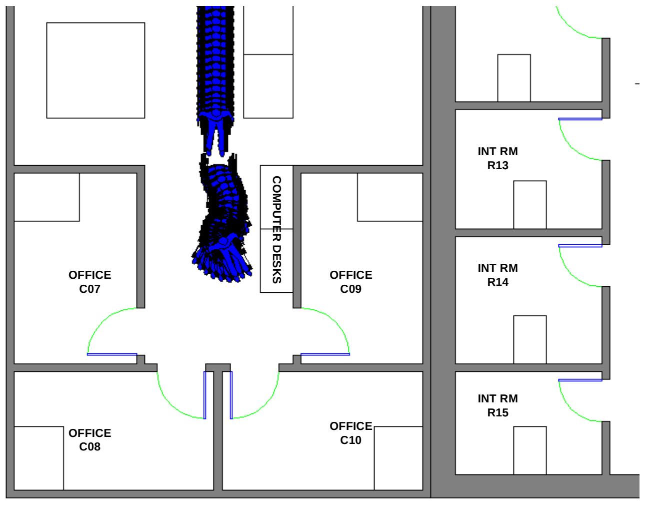


Figure 6.24: Wheelchair route to adjacent computer desk.

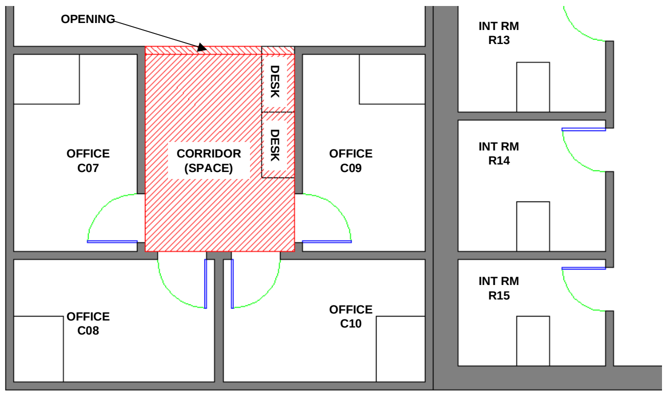


Figure 6.25: Upper computer desk is not fully contained by the corridor space.

6.4 Summary and Discussion

This chapter presented the automated performance-based analysis of a facility at Stanford University Campus. The results and a comparison with an actual wheelchair user's interaction with the facility influenced several modifications to the performance-based methods. In addition, the analysis revealed some of the shortcomings of using the prescriptive-based parameters to develop the performance-based parameters. Specifically, setting the goal point based on the ADAAG-prescribed clearance of a building component is overly restrictive.

As shown in the Interview Room example, the equivalent access to facilities intent depends on a clear understanding of the functionality of a set of spaces. While basing usability on intent instead of a set of prescribed provisions provides the most flexibility, the execution of this intent must cover all situations or at provide the facility to cover future provisions.

Upon completion of the analysis of the CDC, the findings and a set of recommendations were presented to the University's ADA/Section 504[8] Compliance Officer, Rosa Gonzalez. One recommendation affects the current CDC building and makes suggestions on the future design of the future CDC building. Ms. Gonzalez agreed with these recommendations and will have them implemented.

As noted by the analysis and as confirmed by the onsite investigation, the addition of the stall partitions in the Men's bathroom not only inhibits comfortable usage of the toilet facility, it completely inhibits the use of the toilet for the wheelchair user as shown in Figure 6.16. The analysis also showed that the removal of these stall partitions would restore the accessibility of the facility.

With the removal of the partitions, the Men's bathroom would revert back to a single-occupancy from a multiple-occupancy toilet. However, ensuring usability of the facility for wheelchair users should be the main priority. Therefore, the recommendation to remove these partitions was made to Ms. Gonzales. Though the current CDC is slated for impending demolition, Ms. Gonzales stated that she would recommend the removal of the partitions since the facility may be operational for several more years.

The future CDC will also have interview facilities, and while the ADAAG does not specifically address the accessibility of Interview Rooms, only some fraction of the rooms will probably be required to be accessible since, by definition, the rooms serve the same functionality. However, as noted in the analysis of the CDC Interview Rooms, the unique interview environment makes it necessary for all Interview Rooms to be usable by persons with disabilities.

From the experience of analyzing the current CDC, the recommendation was made to Ms. Gonzalez that in the event that all the Interview Rooms are not ADAAG code-

[8] The United States Department of Education Office of Civil Rights Section 504 of the Rehabilitation Act of 1973 requires that no qualified handicapped person shall, on the basis of handicap, be excluded from participation in, be denied the benefits of, or be subjected to discrimination under any program or activity which receives or benefits from Federal financial assistance.

compliant, they should at least be made usable. The performance-based motion-planning accessible route methods developed in this thesis could be a viable tool to determine the usability of these new rooms. As the intent of the ADAAG is to give equivalent access to facilities, whether or not these facilities actually meet some set of prescribed measurements should be secondary to providing actual usability. Ms. Gonzalez concurred and said that an effort would be made to make sure that all the rooms were, if not code-compliant, at least made usable as measured by some alternative metrics.

Chapter 7

Discussion and Summary

This research has developed a framework to aid the designer with the understanding of disabled access issues in the design of a facility, specifically the wheelchair access to a facility's building components. Two goals motivated the development of the disabled access design-aid framework:

- Providing the designer with a set of disabled access analysis tools that complement the prescriptive-based code-checking analysis process.
- Providing designers with an integrated computer environment in which they can access these tools.

These two goals have generated the following research objectives:

- Development of a conceptual model (the design-intent model) that extracts the intent of the disabled access code to provide a computational environment that supports automated analysis.

- Development of performance-based methods that capture wheelchair movement and behavior and show with examples the superiority of these methods to the prescriptive-based building code methods

- Development of a flexible computer environment that allows the seamless integration of the conceptual model, the performance-based methods, and other complementary analysis mechanisms.

This chapter discusses the research findings and examines possible future extensions to the work. Through the investigation of the ADAAG, the development of the wheelchair motion-planning methods, and the analysis of a University facility using the developed framework and methods, this research found inconsistencies and contradictions in the ADAAG. As a result, this chapter presents a set of suggested alternative provisions to the ADAAG. In addition, this chapter examines the developed research both in the context of future extensions and limitations of the disabled access work and the issues involved with applying the developed framework and methods to other forms of design-related analyses.

The chapter is organized as follows:

- Section 7.1 describes example alternative ADAAG provisions.

- Section 7.2 describes future directions and limitations of the presented research.

- Section 7.3 provides the final discussion of the developed research.

- Finally, Section 7.4 summarizes the research presented in this thesis.

7.1 Proposed Alternative ADAAG Provisions

This research has developed an automated analysis of facility designs against the intent of the ADAAG. Accessibility analysis methods have been developed in contrast to the static prescribed ADAAG provisions. As a result, this section presents example provision recommendations that provide the designer with alternative and more accurate methods to demonstrate wheelchair accessibility as well as to support the automated disabled access analysis process. The last alternative provision presented in this section evolved as a result of the test case analysis of the Career Development Center (CDC) and discussions with the employees of the CDC as well as with Rosa Gonzales, the University's ADA/Section 504 Compliance Officer.

An Alternative, Computable Definition of the Accessible Route:

The ADAAG defines an accessible route as follows [1]:

3.5 Definitions.

Accessible Route. A continuous unobstructed path connecting all accessible elements and spaces of a building or facility. Interior accessible routes may include corridors, floors, ramps, elevators, lifts, and clear floor space at fixtures. Exterior accessible routes may include parking access aisles, curb ramps, crosswalks at vehicular ways, walks, ramps, and lifts.

This research developed and demonstrated a computable form of the accessible route as described in Chapters 3 and 4. The recommendation is to incorporate into the ADAAG an alternative definition that would accept a computable accessible route. This alternative definition defines the elements and the sequence of elements that compose any accessible route through a facility. As with the ADAAG accessible route definition, alternative accessible route definition does not define the accessibility of the individual components; these definitions are presented subsequently in other proposed provisions. The proposed alternate definition is given as follows:

An accessible route R is defined as a composition of accessible components:

R	=	$R_{init} + \Sigma R_{sos} + R_{goal}$

where:

R_{sos}	=	R_{seg} <+ R_{open} + R_{seg}>,
R_{init}	=	the initial point *(the starting point of an accessible route),*
R_{goal}	=	the goal point *(the ending point of an accessible route),*
R_{seg}	=	a segment of the accessible route within a space,
R_{open}	=	the clearance area at an opening*, and*
<>	=	optional arguments

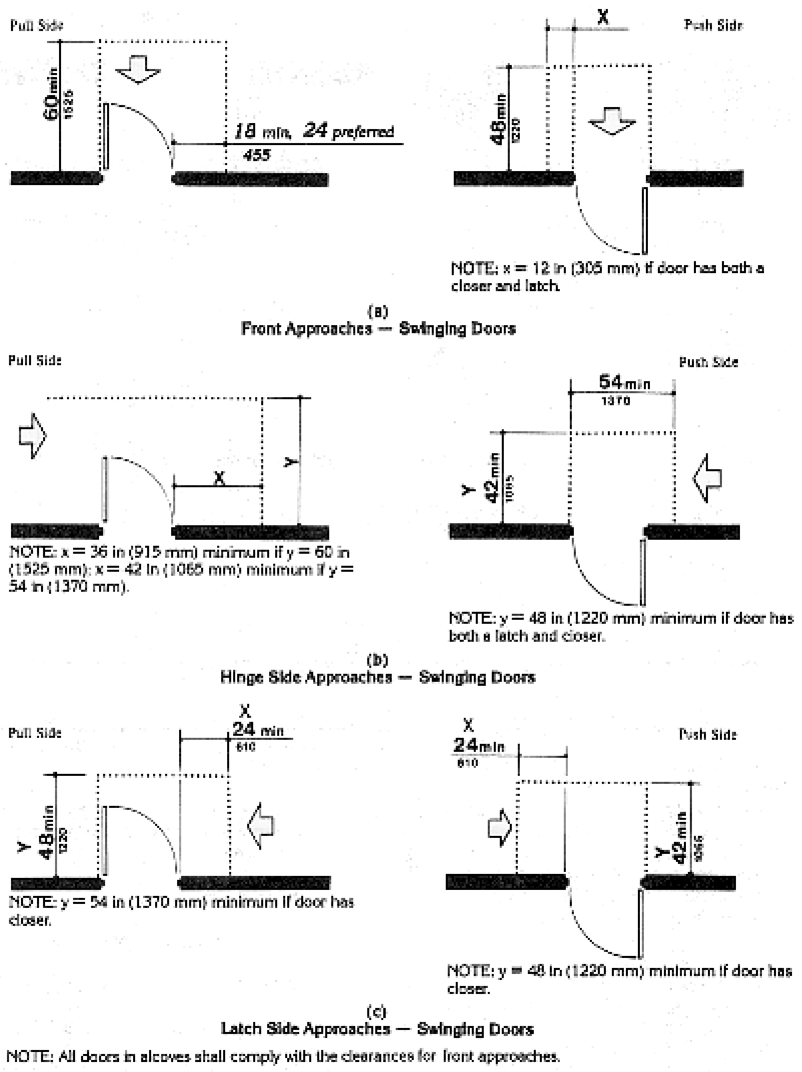
R_{init} and R_{goal} nodes may also be instances of R_{open} nodes.

An Alternative Definition for Maneuvering Clearance:

The ADAAG defines multiple maneuvering clearances at building components to address the different approaches to these components. As noted in Chapter 4, the static nature of the accessible route test methods dictates the need for multiple clearance boxes. In addition, accessibility of doors can be demonstrated even if the maneuvering clearances are violated. If the ADAAG allowed dynamic accessible route test methods, a simpler, more accurate "approach" definition could be used as an alternative to the maneuvering clearance definition.

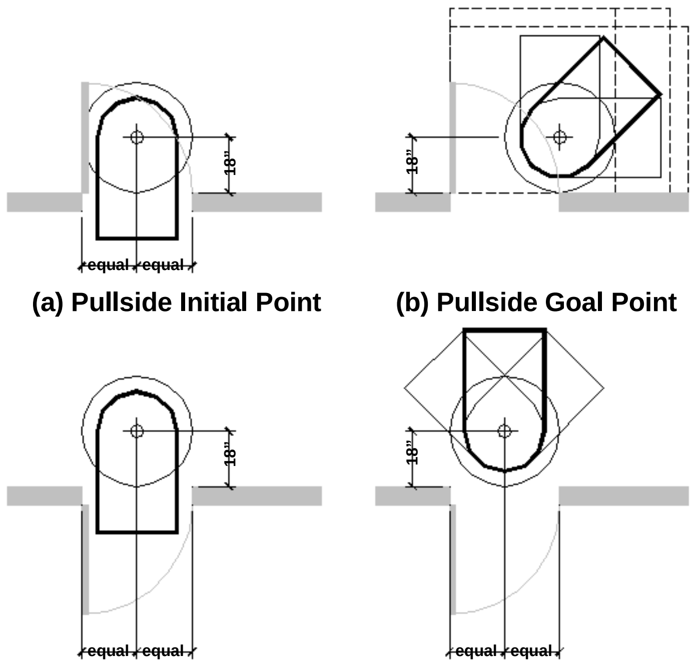
The following is the ADAAG maneuvering clearance definitions at doors [1]:

Maneuvering Clearances at Doors. Minimum maneuvering clearances at doors that are not automatic or power-assisted shall be as shown in Fig. 25 (shown below). The floor or ground area within the required clearances shall be level and clear.


Pull Side
60min 1525
18 min, 24 preferred 455
X
Push Side
48min 1220
NOTE: x = 12 in (305 mm) if door has both a closer and latch.
(a)
Front Approaches — Swinging Doors
Pull Side
X
Y
NOTE: x = 36 in (915 mm) minimum if y = 60 in (1525 mm); x = 42 in (1065 mm) minimum if y = 54 in (1370 mm).
Push Side
54min 1370
Y 42min 1065
NOTE: y = 48 in (1220 mm) minimum if door has both a latch and closer.
(b)
Hinge Side Approaches — Swinging Doors
Pull Side
X 24 min 610
Y 48min 1220
NOTE: y = 54 in (1370 mm) minimum if door has closer.
X 24min 610
Push Side
Y 42min 1065
NOTE: y = 48 in (1220 mm) minimum if door has closer.
(c)
Latch Side Approaches — Swinging Doors
NOTE: All doors in alcoves shall comply with the clearances for front approaches.

The following proposed alternate definition reduces the number of geometries that must be tested by directly modeling the door approach:

Approach to Doors may be alternatively defined as shown in the below figure as long as an accessible path to the approach point is also defined. The floor or ground area within the required clearances shall be level and clear.



(a) Pullside Initial Point **(b) Pullside Goal Point**

(c) Pushside Initial Point **(d) Pushside Goal Point**

<u>A Definition for Accessible Route Width:</u>

As demonstrated in Chapter 4, the ADAAG prescribed accessible route width does not accurately address all possible facility configurations. The chapter illustrated a non-compliant usable example and a possibly-compliant unusable example. To address the shortcomings in the accessible route width definition, the following is a proposed alternative definition:

Variances will be allowed for accessible route widths and configurations that adhere to the ergonomic and anthropometric parameters set forth in this document (the ADAAG) that can be demonstrated visually, either manually or computationally.

An example of a visual demonstration would be the path generated by the motion-planning techniques developed in this research.

An Alternative Definition for Equivalent Access to Facilities:

Finally, this research recommends a more general definition for equivalent access to facilities than currently exists in the ADAAG since all possible facility functions and their accessibility equivalents cannot possibly be defined in the ADAAG *a priori*:

> Equivalent access to facilities or facility components that is not explicitly defined in this document (the ADAAG) shall be defined on a case-by-case basis by the relevant Department of Justice enforcing body.

7.2 Future Directions and Limitations

7.2.1 Extending the Disabled Access Analysis Research

Section 4.2.5 discussed several of the limitations of the performance-based motion-planning techniques developed to address wheelchair accessibility analysis. This section examines some of the other limitations and proposals to address these limitations. In addition, this section examines other aspects of the disabled access code that must be addressed to provide full disabled access usability analysis. Finally, this section outlines a proposal to gather information about the interaction of "digital actors" to examine the usability of a facility over an established time frame.

This research has attempted to develop motion-planning techniques to directly model wheelchair motion and behavior, but even within this performance-based approach, the research has prescribed goal point parameters assuming usability of a particular building component once the motion planner has directed the wheelchair robot to the goal point. A logical extension to this assumption is the direct modeling of the wheelchair user interacting with the building component. The technique would involve combining the wheelchair robot planner developed in this research with path-planning algorithms to automatically generate sequences for human figures as described by Koga et al. [40].

Modeling the wheelchair user's actions provides a more complete analysis of the user's interaction with a particular building component.

This research has not addressed other disabled access issues such as sight and hearing impairment. Specifically addressing sight disabilities, motion-planning techniques can be applied to model the motion and behavior of blind persons through a facility. For this particular problem, however, the configuration of the facility should not be pre-calculated as the individual must "discover" the path as he or she moves toward the goal. Kuffner and Latombe have developed techniques that allow the digital actor to discover the surrounding area through visual perception, and these techniques could be adjusted to allow the digital actor to discover the configuration by modeling cane interaction with the facility [42].

Finally, an even more dynamic approach than the wheelchair analysis developed in this research involves developing a time-elapsed simulation using multiple digital actors assigned typical tasks and collecting statistics on the actions and reactions of the actors. The collection of actions, such as the number of required backups and the average time needed to execute a particular task, would provide insight into the usability of a facility by directly modeling the "quality of life" of the individual or set of individuals in the facility.

7.2.2 Applying Motion Planning to other Design-related Processes

While this research has applied motion-planning techniques to wheelchair behavior, more generally, motion-planning techniques can be applied to design and construction analyses that involve movement of particles, people, and equipment or the coordination of a combination of these elements. Specifically, these techniques can directly capture the motion and behavior of the elements in question. The techniques that will be developed will be domain-specific, and accurate modeling of the particular behavior is not a trivial issue. Thus the development of domain-specific performance-based analyses depends on the researcher's ability to accurately model the relevant behavior.

Motion-planning techniques naturally lend themselves to the egress problem as the main issue concerns the movement of people through a space (and through the facility) along an exit path or route. As with the wheelchair accessible route problem, motion planning can be used to directly model person movement, and the egress motion planning extends the basic motion-planning problem. As with the sight impaired example presented in the previous section, techniques involving discovering the configuration space apply to this problem since the individuals may not know the layout of the facility *a priori* [42].

Egress motion planning would also need to take into consideration the coordination of multiple moving objects [37]. The multiple moving objects motion-planning problem extends the notion of the *configuration space* to a *configuration-time* space, and the *time* dimension is different from the other dimensions in that it is not reversible. For the egress problem, this difference reflects the actual situation: the amount of time needed to exit a space based on the exiting arrangement is a critical parameter in determining the safety of the space. Multiple robots dramatically increase the problem's dimension. High-dimensional configuration spaces necessitate the introduction of techniques such as randomized path planning to reduce the computation time [37].

Finally, motion-planning techniques could be applied to provide Architecture/Engineering/Construction (AEC) professionals with insight into construction and assembly sequences. Geem et al. develop mobility analysis for feasibility studies in industrial environments as part of the Motion for Logisistics (MOLOG) effort, a group dedicated to the development of "motion design technology in the framework of the logistics engineering of industrial installations" [19].[9]

[9] See http://www.laas.fr/molog/.

7.2.3 Extending the Design-aid Framework to Modeling Other Building Codes

This research has demonstrated the power of the design-aid framework and its interacting components (the design-intent model, the product model, and the document model), but has not addressed the generality of the framework. This section outlines the incorporation of egress analysis into the framework developed in this research. The process for developing the egress analysis model is identical to the disabled access modeling process:

1. Develop a design-intent model of egress analysis by extracting the intent hierarchy from an egress building code.

2. Develop the motion-planning performance-based methods that capture the behavior of persons exiting a facility and populate the design-intent model with these methods.

The motion-planning performance-based egress methods have been discussed in the previous section. Figure 7.1 shows a possible egress design-intent hierarchy. The intent of the Uniform Building Code (UBC) egress code can be refined to providing a safe exiting system, and an exiting system terminates to a public way or equivalent [26]. As shown in Figure 7.1 with the `OR` arc between the connections emanating from the safe exiting system intent, UBC alternatively allows an exiting system to terminate to an area of refuge for persons with disabilities.

As with the disabled access problem, the egress problem decomposes the analysis of a facility by architectural functionality (buildings, stories, and spaces), and the egress design-intent model shown in Figure 7.1 reflects this decomposition for both the public way intent and area of refuge intent.

In modeling other analyses such as fire and egress that are dependent on the architectural functionality decomposition or view of a facility (buildings, stories, and spaces), some portion of the design-intent model's hierarchy will incorporate the same intent

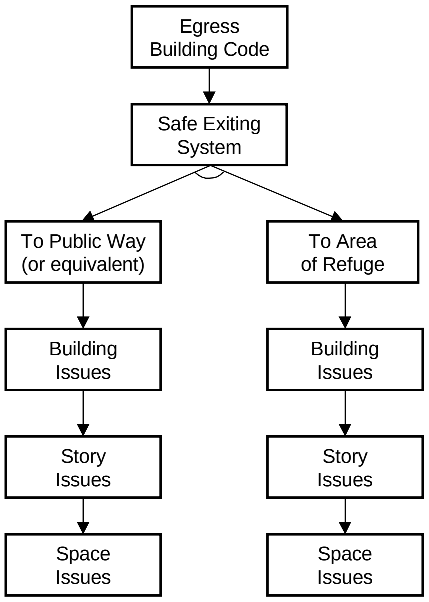


Figure 7.1: An egress design-intent model hierarchy.

decomposition developed for the disabled access problem. However, capturing all aspects of a code's intent is not always straightforward. The evolution of a certain prescribed provision may obscure the design intent of the provision. For example, the UBC specifies that exit doors must swing in the direction of egress if the serviced area has an occupant load of fifty or more [58]. The origin or intent behind this prescribed occupant load is not directly evident, but an occupant load of fifty or more classifies a space as an assembly area [58], and the exit door swing direction is related to the assembly area occupancy load criteria.[10]

7.3 Discussion

The research presented in this thesis consists of a collection of models and methods that, when combined, produce a powerful framework that enables the automated analysis of a

[10] Andrew Aldeman, former chief building official of the City of San Jose, pointed out this connection in a December 4, 1996 International Conference of Building Officials (ICBO) Peninsula Chapter meeting.

facility for disabled access usability. This section reviews the developed models, methods, and environments examining the developed research from a more general perspective.

7.3.1 The Design-aid Framework

The design-intent module of the design-aid framework provides a flexible approach to organize the intent of a design standard. Using this approach, this research has been able to develop methods associated with the generated intent of the disabled access code that analyze a facility's wheelchair usability. Casting the ADAAG as a hierarchical design-intent structure provides more flexibility than the prescriptive-based provisions that constitute the disabled access code.

The intent of equivalent access to facilities is an extremely powerful concept since if the code is organized around this concept as opposed to require the designer to adhere to a set of prescriptive provisions, the code could be more flexible and be tailored to accommodate non-standard situations. Requirements could be adjusted according to the parameters of the situation. For example, the ADAAG specifically prescribes parking parameters for a general and a limited number of special situations such as outpatient facilities and facilities that cater to the mobility impaired. Using the design-intent model with another flexible approach such as Garcia's Active Design Documents (ADD) model, one can imagine tailoring the parking requirements based on a dynamic set of requirements while still adhering to the concept of equivalent access [17].

This research developed the product model using the form-function-behavior (FFB) approach to provide the maximum flexibility in defining the parameters of both the facility description and the analysis results. The research quickly realized the power of this approach through the ease of incorporating new combined form, function, or behavior concepts to the analysis. For example, the research easily incorporated the concept of the accessible route in the model using the description (the form) of a polyline to describe the route and associating this geometric description with the accessibility concept (the function) of a system of building components.

This research has demonstrated the power of the design-aid framework for the disabled access problem and makes the argument that the interaction of the three developed models (the symbolic product model, the design-intent model, and the document model) that constitute the design-aid framework can be applied to other code-related analyses. The research has also indirectly validated Yabuki's Hyper-Object-Logic model since the system architecture of the design-aid framework was strongly influenced by Yabuki's design [60]. Based on the success of the design-aid framework for the disabled access problem, this research submits that the structure of the design-aid framework should be used to design other usability and code-related automated analyses applications. The design-intent model is a powerful tool for organizing the often ambiguous, contradictory, and insufficient abstractions of prescribed provisions that constitute a design standard. In addition the flexibility and interaction of the developed product model with the design-aid framework allows for the description of the functionality of building components or systems of building components as well as the analysis results.

This research has developed the design-aid framework and the motion-planning techniques to model wheelchair movement separately and has presented other design and construction-related analyses that would be suitable to motion-planning techniques in this chapter. However, the application of the motion-planning methods to the design-intent methods used to test the intent of the disabled access code underlines the strong connection between these techniques for the disabled access problem. More generally, this connection is applicable when the intent of a code or process can be demonstrated with the direct modeling of moving elements, either persons, particles, or building components.

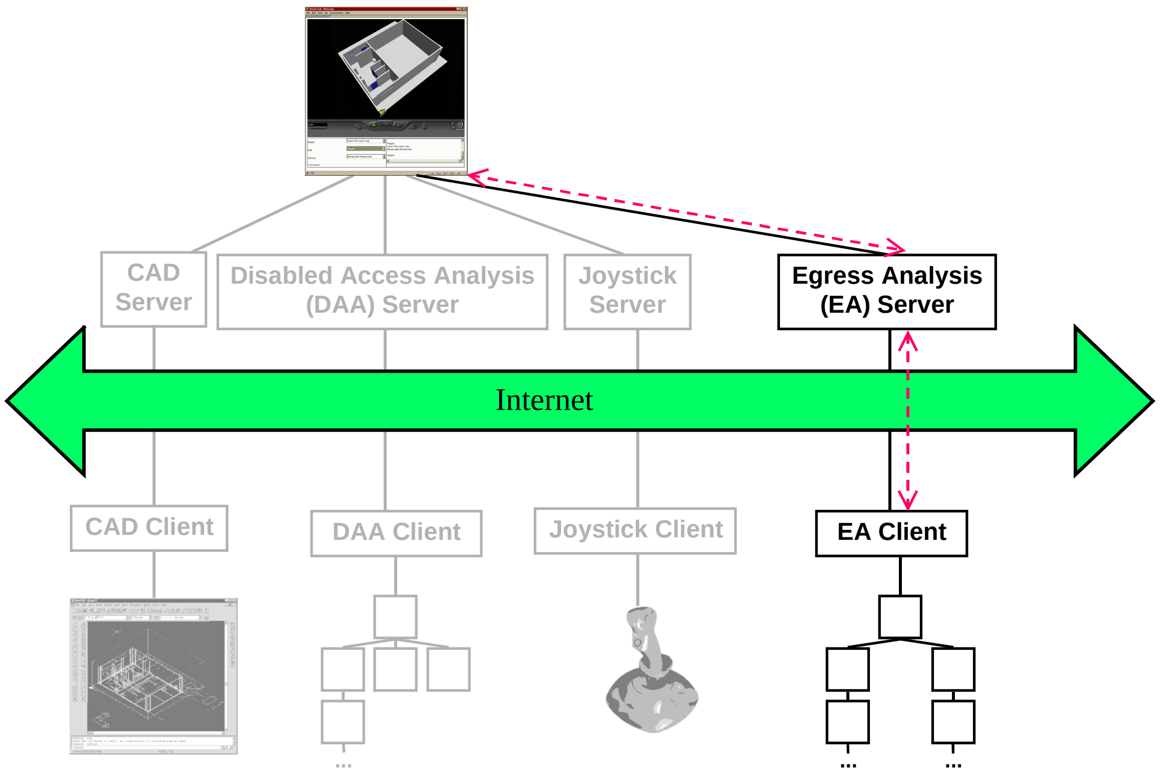


Figure 7.2: Adding egress analysis to the distributed object service environment (DOSE).

7.3.2 The Distributed Object Service Environment

Finally, the distributed object service environment (DOSE) represents a general component-based approach to aggregating engineering analysis systems. This research demonstrated the power and generality of the `Service` object three-tiered architecture by implementing the various accessibility tools of design-aid framework with this system architecture. Indeed, the proposed egress framework outlined earlier in this chapter could be incorporated in the developed system architecture as easily as the incorporation of each of the disabled access design tools. Figure 7.2 shows the incorporation of the egress design-aid framework into the distributed object service environment (DOSE).

The egress intents would map to instances of `Intent` objects, and these objects are implemented as `Service` objects in the DOSE. The DOSE would view the egress `Intent` objects as a composite service (an aggregation of `Service` objects). The Visual Interactive Environment/Workbench (VIEW) would send the facility design to this composite service through the *Egress Server*, a `Service` object that is analogous to the *Analysis Servers* developed for the disabled access analysis. The VIEW would have to be modified to be aware of the existence of the added *Egress Server*. Given the models and methods developed in this research, developing an egress analysis model and incorporating this model into the DOSE are straightforward steps.

To be even more general, the DOSE must be modified to accommodate a variety of analyses in the following manner. The communication protocol developed in this research can still be utilized for a variety of analyses, but a `Service` object will need to accommodate different product models, and some service components will simply be mapping services between disparate product models. Finally, while the DOSE communication protocol acts as a common interface to aggregate disparate analyses, the overhead associated with the distributed object environment makes it impractical as a protocol for high-bandwidth, interprocess communication among many nodes. In this case, an interprocess-specific direct communication protocol would be more appropriate, and the DOSE should consist of some hybrid of the standard distributed object communication protocol and interprocess-specific direct communication protocol.

7.4 Summary

This research provided a computer environment that aids the designer with disabled access design. Toward this goal, the research developed a disabled access design-aid framework consisting of the interaction among three models:

- The description of the facility (the developed symbolic product model)

- The intent of the Americans with Disabilities Act Accessibility Guidelines (ADAAG) (the disabled access design-intent model)
- A document model

The disabled access design-intent model extracts the facility design from an instance of the product model, the design-intent model analyzes the facility, and the design-aid framework adds the analysis results to the product model's building components. The disabled access design-aid framework then provides the designer with text and graphical analysis results of the facility utilizing the document model. As shown in the generated analysis of the test case, this research has designed of the components of the disabled access design-aid framework with the flexibility to analyze the facility using the performance-based methods while displaying the ADAAG provisions that motivated the specific analysis.

In summary, this research has focused on providing the designer with a computer environment with access to a variety of disabled access analysis tools. Toward this goal, the research has:

- Abstracted the design intent of the ADAAG to form a hierarchical model that enables the automated disabled access analysis of a facility design.
- Developed a computable form of the accessible route, the most critical concept related to access of a facility.
- Demonstrated a performance-based approach using motion-planning techniques to model the wheelchair accessibility analysis methods associated with design-intent model.
- Demonstrated the flexibility of the developed product model that incorporates both the facility design description and analysis results.

- Demonstrated the power of the design-aid framework in analyzing a facility for wheelchair accessibility by providing the interaction among the developed product model, the design-intent model, and a document model to generate the analysis results.

- Implemented the framework in a distributed object service environment representing a flexible component-based approach to aggregating engineering analysis systems.

Finally, from the results of the Career Development Center analysis, the research presented a set of recommendations to the University to increase the usability of the current and future CDC. The United States Department of Justice enforces the prescriptive-based ADAAG, and this research does not seek to replace the ADAAG with the developed framework and the performance-based analysis methods. However, in developing and testing the performance-based methods, this research has pointed out some of the inadequacies and contradictions of the prescriptive-based ADAAG. In addition, the research proposed a set of alternative provisions based on the performance-based work that will hopefully be deemed acceptable by the regulatory governing body, United States Access Board, the developer and maintainer of the ADAAG.

Bibliography

[1] *Americans with disabilities act accessibility guidelines,* Access Board, U.S. Architectural and Transportation Barriers Compliance Board, Washington, D.C., 1997.

[2] Godfried Augenbroe (Ed.), COMBINE 1 final report, CEC-JOULE report, Brussels, 1993.

[3] Godfried Augenbroe (Ed.), COMBINE 2 final report, CEC-JOULE report, Brussels, 1995.

[4] *AutoCAD customization manual, AutoCAD release 12, Publication 100891-01*. Autodesk, Inc., 1997.

[5] Aart Bihl, “Computer aided housing and sight layout: experience of research software in a production environment,” *Proceedings, PARC’79, International Conference on the Applications of Computers in Architecture, Building Design and Urban Planning*, 283-304, 1979.

[6] B-C Björk, “Basic structure of a building product model,” *Computer-Aided Design*, 1989, (22) 71-78, 1989.

[7] W.K. Chow and W.K. Mok, “CFD fire simulations with four turbulence models and their combinations,” *Journal of Fire Sciences*, 17(3), 209-239, 1999.

[8] *CIMsteel: the logical product model, Version 3.3*, University of Leeds, 1993.

[9] Marcel de Waard. *Computer aided conformance checking: checking residential building designs against building regulations with the aid of computers*, Ph.D. thesis, The Hague, The Netherlands, 1992.

[10] Lester E. Dubins. “On curves of minimal length with a constraint on average curvature with prescribed initial and terminal positions and tangents,” *American Journal of Mathematics*, (79) 497-516, 1957.

[11] Charles M. Eastman. *Building product models*, CRC Press, Boca Raton, Florida, 1999.

[12] John P. Eberhard, “Computer based code systems and the performance concept,” *Performance Concept in Buildings*, NBS Special Publication 361, Volume 2, 1972.

[13] Steven J. Fenves. “Tabular decision logic for structural design,” *Journal of Structural Engineering*, 92(6), 473-490, 1966.

[14] Steven J. Fenves, J.H. Garrett, H. Kiliccote, K.H. Law, and K.A. Reed. “Computer representations of design standards and building codes: U.S. perspective,” *The International Journal of Construction Information Technology*, 3(1), 13-34, 1995.

[15] Steven J. Fenves, R. Wright, F. Stahl, and K. Reed. *Introduction to SASE: standards analysis, synthesis and expression*, NBSIR 87-3513, 1987.

[16] Teresa Forowicz, “Modeling of energy demands for residential buildings with html interface,” *Automation in Construction*, 8(4):481-487, 1999.

[17] Ana Cristina Bicharra Garcia, H.C. Howard, M.J. Stefik. *Active design documents: a new approach for supporting documentation in preliminary routine design*,

Technical Report 82, Center for Integrated Facility Engineering, Stanford University, 1993.

[18] James H. Garrett, Jr. and M. Hakim. “Object-oriented model of engineering design standards,” *Journal of Computing in Civil Engineering*, 6(3):323-347, 1992.

[19] C. Van Geem, T. Simeon, J.P. Laumond, J.L. Bouchet, and J.F. Rit. “Mobility analysis for feasibility studies in cad models of industrial environments,” *IEEE International Conference on Robotics and Automation*, 1999.

[20] Michael P. Gibbens. *California disabled accessibility guidebook 1998*, Builder’s Book, Inc., Canoga Park, California, 1998.

[21] Wim F. Gielingh. *General AEC reference model*, ISO TC 184/SC4/WG1 3.2.2.1, TNO Report BI-88-150, 1988.

[22] *Glossary of building and civil engineering terms*, BS6100, British Standards Institute, London, England, 1992.

[23] George V. Hadjisophocleous and N. Benichou. “Performance criteria used in fire safety design,” *Automation in Construction*, 8(4):489-501, 1999.

[24] Charles S. Han, J.C. Kunz, and K.H. Law. “A client/server framework for on-line building code checking,” *Journal of Computing in Civil Engineering, ASCE*, 12(4), 181-194, 1998.

[25] Charles S. Han, J.C. Kunz, and K.H. Law. “Building design services in a distributed architecture,” *Journal of Computing in Civil Engineering*, ASCE, 13(1), 12-22, 1999.

[26] *Handbook to the uniform building code: an illustrative commentary*, International Conference of Building Code Officials, Whittier, California, 1991.

[27] E.M. Hoskins, “The OXSYS system”, *Computer Applications in Architecture*, 343-391, 1977

[28] H. Craig Howard, J.A. Abdalla, and D.H.D. Phan. *A primitive-composite approach for structural data modeling*, Technical Report 52, Center for Integrated Facility Engineering, Stanford University, 1989.

[29] David Hsu, R. Kindel, J. Latombe, and S. Rock. “Randomized kinodynamic motion planning with moving obstacles,” *Workshop on Algorithmic Foundations of Robotics*, 2000.

[30] *Initial graphics exchange standard, Version 5.1, NISTIR 4412.* U.S. National Bureau of Standards, Gathersburg, Maryland, 1991

[31] *Industry foundation classes release 1.5, IFC specifications development guide.* International Alliance for Interoperability, Washington D.C., 1997

[32] *Industry foundation classes release 1.5, Model architecture guide.* International Alliance for Interoperability, Washington D.C., 1997.

[33] *Industry foundation classes release 1.5, Specifications volumes 1-4.* International Alliance for Interoperability, Washington D.C., 1997.

[34] *Information technology—computer graphics and image processing—the virtual reality modeling language (VRML)—Part 1: Functional specification and UTF-8 encoding.* ISO/IEC 14772-1:1997, International Standards Organization, Geneva, Switzerland, 1997.

[35] *Integrated information support system, Volume V: Common data model subsystem,* Mantech Technology Transfer Center, WL/MTX, Wright-Patterson AFB, Ohio, 1985.

[36] Kenji Ito, Y. Ueno, R.E. Levitt, and A. Darwiche. *Linking knowledge-based systems to CAD design data with an object-oriented building product model*, Technical Report 17, Center for Integrated Facility Engineering, Stanford University, 1989.

[37] Jean-Claude Latombe. *Robot motion planning*, Kluwer Academic Publishers, Norwell, Massachusetts, 1991.

[38] Gregory Paul Luth. *Representation and reasoning for integrated structural design*, Ph.D. thesis, Department of Civil Engineering, Stanford University, 1991.

[39] Han Kiliccote. *A standards processing framework*, Ph.D. thesis, Department of Civil and Environmental Engineering, Carnegie Mellon University, 1996.

[40] Yoshihito Koga, K. Kondo, J.J. Kuffner, and J-C Latombe, "Planning motions with intentions," *Proceedings of SIGGRAPH'94, Computer Graphics Proceedings, Annual Conference Series*, 395-408, 1994.

[41] Helmut Krawinkler, "Advancing performance-based earthquake engineering," *Peer Center News*, 2(1), Pacific Earthquake Engineering Research Center, Richmond, California, 1999.

[42] James J. Kuffner, Jr. and J-C Latombe, "Fast synthetic vision, memory, and learning models for virtual humans," *Proceedings of CA'99: IEEE International Conference on Computer Animation*, 118-127, 1999.

[43] T. Kusuda and F.J. Powell, "Use of modern computer programs to evaluate dynamic heat transfer and energy use processes in buildings," *Performance Concept in Buildings*, NBS Special Publication 361, Volume 2, 1972

[44] *Manual of steel construction—Load & resistance factor design*, First Edition, American Institute of Steel Construction, Inc., 1986.

[45] Chris Marrin. *Proposal for a VRML 2.0 informative annex: External authoring interface reference*, an unpublished draft of a proposal to the ISO, 1997.

[46] Chris Marrin and B. Campbell. *Teach yourself VRML 2 in twenty-one days*, Sams.net Publishing, Indianapolis, Indiana, 1997.

[47] K.B. McGrattan, H.R. Baum, and R.G. Rehm, "Large eddy simulations of smoke movement," *Fire Safety Journal*, 30(2), 161-178, 1998.

[48] G.M. Nijssen and T. Halpin, *Conceptual schema and relational database design: A fact-oriented approach*, Prentice Hall, London, 1989

[49] *Performance concept in buildings*, NBS Special Publication 361, Volumes 1 and 2, United States Department of Commerce, Washington D.C., 1972.

[50] *Product data representation and exchange, Part 1: Overview and fundamental principles*. ISO 10303-1, International Standards Organization, Geneva, Switzerland, 1994.

[51] *Product data representation and exchange, Part 106: Integrated resources: Building core construction model (Draft)*. ISO 10303-11, International Standards Organization, Geneva, Switzerland, 1996.

[52] *Product data representation and exchange, Part 11: Description methods: The EXPRESS language reference manual*. ISO 10303-11, International Standards Organization, Geneva, Switzerland, 1994.

[53] *Product data representation and exchange, Part 225: Application protocol: Building elements using explicit shape representation*. ISO 10303-225, International Standards Organization, Geneva, Switzerland, 1999.

[54] Douglas Schenk and P. Wilson, *Information modeling the EXPRESS way*, Oxford University Press, New York, 1994.

[55] James A. Reeds and R.A. Shepp. “Optimal paths for a car that goes both forward and backward,” *Pacific Journal of Mathematics*, 145(2), 367-393, 1991.

[56] L.G. Seigel, “A performance approach to the design of fire-resistive buildings,” *Performance Concept in Buildings*, NBS Special Publication 361, Volume 2, 1972.

[57] *Title 24, Section 6—Energy efficiency standards for residential and nonresidential buildings,* California Energy Commission, Sacramento, California, 1999.

[58] *Uniform building code*, International Conference of Building Code Officials, Whittier, California, 1991.

[59] Andreas Vogel and K. Duddy. *Java programming with CORBA,* John Wiley and Sons, Inc., New York, N.Y., 1997.

[60] Nobuyoshi Yabuki. *An integrated framework for design standards processing,* Ph.D. thesis, Stanford University, 1992

[61] F.Y. Yokel and N.F. Somes, “Philosophy and scope of structural performance criteria,” *Performance Concept in Buildings*, NBS Special Publication 361, Volume 2, 1972.