ABSTRACT

Title of thesis: **COMPUTATIONAL FRAMEWORK FOR PARAMETRIC MODELING AND ARCHITECTURE-ENERGY ASSESSMENT OF BUILDING FLOORPLANS**

Eddie Tseng, Master of Science, 2015

Thesis directed by: Associate Professor Mark Austin
Department of Civil and Environmental Engineering
and ISR

Modern building systems can be exceedingly complex. In this research we develop a computational framework for the parametric modeling and architecture-energy assessment of building floorplans. Parametric representations of floorplans are formulated as multi-layer hierarchies, with adjacent layers coupled by dependency relationships. Software is developed for two approaches to floorplan specification: (1) scripting, and (2) interactive graphical techniques. Computational procedures are developed for assessment of building code regulations, electricity cost assessment, and simplified HVAC component selection and architecture-energy sensitivity analysis. A case study analysis of a two-apartment building system is presented.

Last Modified: April 27, 2015

COMPUTATIONAL FRAMEWORK FOR PARAMETRIC MODELING AND ARCHITECTURE-ENERGY ASSESSMENT OF BUILDING FLOORPLANS

by

Eddie Tseng

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2015

Advisory Committee:
Associate Professor Mark Austin, Chair/Advisor,
Assistant Professor Huan Xu,
Professor Andre Tits.

Acknowledgments

This master thesis would not have been possible to finish without the support from all the great people around me. As such, I dedicate my thesis to them.

I owe my deepest gratitude to my advisor, Dr. Mark Austin, who has guided me through my study and research with patience, enthusiasm and expertise. His guidance and effort helped me throughout the whole research and writing process. I simply could not wish for a better advisor for my master research.

I would like to thank the other members of my committee, Dr. Andre Tits and Dr. Huan Xu, for their insightful comments and assistance.

My sincere thanks also goes to my girlfriend and friends. They were always there to encourage me with their wishes and stood by me throughout my study at University of Maryland.

Finally, I would like to thank my parents for loving and supporting me throughout my entire life.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Modern buildings are advanced, self-contained and tightly controlled environments designed to provide a variety of services (e.g., vertical/horizontal transportation, sanitation, artificial lighting, fire protection, environmental conditioning, air quality, communication and security) to their occupants. The design of modern buildings is complicated by intertwined network structures in the arrangement of two- and three-dimensional spaces throughout the building, for the fixed circulatory systems (e.g., power and hvac), for the dynamic circulatory systems (e.g., air flows through rooms), and for wired and wireless communications [35, 58]. In established approaches to building systems development, the project stakeholder implicitly assume that it will be possible to control the complexity of developments through separation and decomposition of design concerns, leading to loosely coupled system architectures and well-defined hierarchies of behaviors. In practice, the need for new forms of functionality drives components from different network types to connect in a variety of ways. Since a change at almost any level may have system-wide consequences, formal procedures are needed for the assurance of robust operations, and in design and trade-study analysis, to understand trade-offs among competing criteria.

At the same time, each discipline will design to satisfy their concerns first but in the end, implementation of discipline-specific concerns must respect inter-disciplinary constraints. In the case of buildings this task is complicated by: (1) the need for single objects to help satisfy the functional concerns of multiple disciplines, and (2) physical network flows (e.g. flows of air and heat) that are influenced by decisions made in multiple disciplines. This leads to interesting multi-disciplinary tradeoff problems.

This project addresses these concerns through the development of model-based systems engineering (MBSE) procedures and computer-aided tools so that the computer can play a pro-active role in the architectural design, assessment, and trade-study/sensitivity analysis for energy-efficient building systems. The immediate research and development focuses on methods for a top-down parametric specification of building floorplans, where high-level adjustments to a design will be automatically propagated to lower-level representations. A second research objective is a preliminary framework for architecture-energy assessment of building floorplans. The expected benefits of these contributions are reduced likelihood of design errors, improved efficiency of design space exploration, and improved energy efficiency.

1.2 Related Work

Long-term Drivers. Given that in developed countries individuals spend as much as 80% of their life indoors (working, sleeping, shopping, and so forth), it is difficult

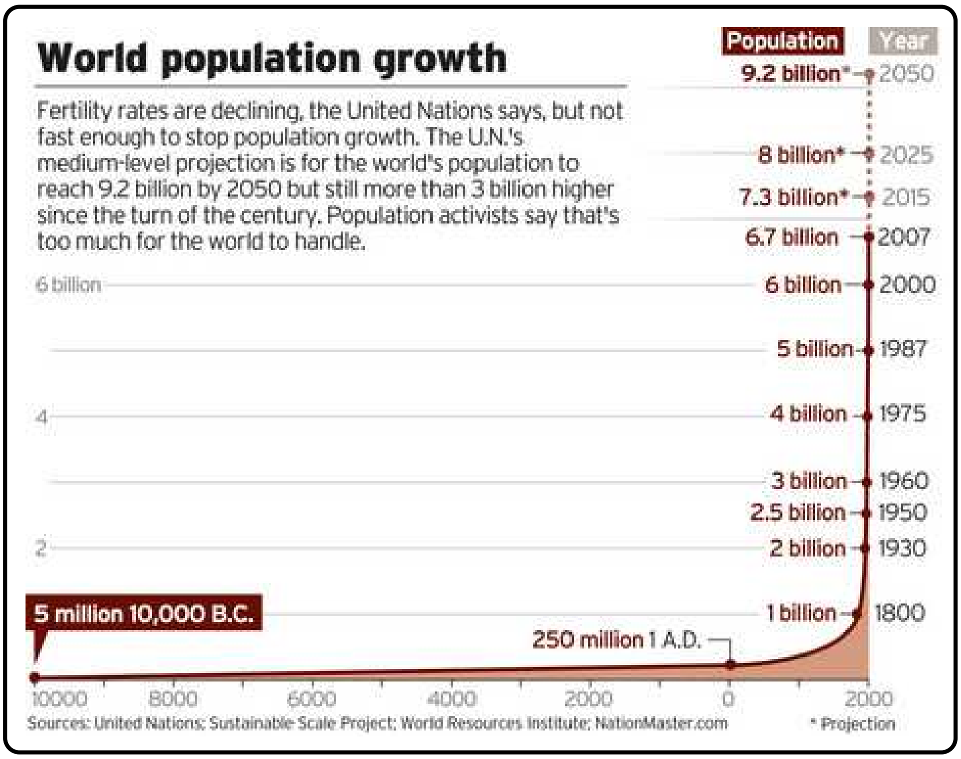


Figure 1.1: Growth in global population versus time.

to overstate the importance of buildings to the vitality and overall well-being of humanity [15, 44, 51]. There are several trends in place that indicate that moving forward, these factors will only grow in importance. First, as illustrated in Figure 1.1, the World's population is rapidly increasing – during the next thirty to forty years the World's population is expected to increase from approximately 7 billion today to somewhere in the range 9 to 10 billion. An increase of 2 to 3 billion people will almost certainly result in additional demands on limited resources; it might even contribute to global warming. And second, there is a general trend toward population urbanization. By 2050, 90% of Americans are expected to live in urban areas. Together, these trends point to a strong need for levels of energy efficiency and economy in building systems that exceed today's state-of-the-art practices. To put the overall problem in perspective, in developed economies (e.g., US and Western Europe) 40-60% of all energy consumption occurs in buildings. Half of this amount is used to provide a comfortable thermal and luminous environment to the occupants [13, 14, 16, 43, 61]. With demands for energy expected to increase into the foreseeable future, socio-economic pressures will drive the need for infrastructure that is increasingly sustainable and smart about its consumption of energy resources [38, 64]. As a case in point, a recent European Union directive [31] specifically requires that all buildings constructed in 2020 or later be “nearly net-zero buildings.” An immediate consequence of this mandate is a strong need for new models and tools to support performance-based approaches to building design. To achieve design configurations that are economical and have low resource consumption, longer term, models will need to capture interoperating domains such as transportation and energy grids, and support global optimization of energy production and consumption in the building environment [34, 38].

Focus on Frontend Development. Naive approaches to building optimization and trade-off often lead to design solutions that are functional, but at the expense of high energy consumption and maintenance [18]. Still, on a local scale even small improvements in energy modeling and efficiency of operations can lead to substantial economic benefits over the lifetime of a building. Recent studies [40] indicate that energy efficient buildings can be increased by more than 50% over current standards (ASHRAE 90.1), with proof points occurring for all sizes and climates. This occurs when control options include both active (e.g., heat pumps) and passive mechanisms (e.g., natural ventilation) linked to building dynamics (i.e., coupling of airflow and thermal properties) and prevailing climate conditions. Unfortunately, with today's technology (e.g., ability to sense and gather data throughout a structure), calibrating a building to achieve optimal performance can take years [40, 62].

Figures 1.2 and 1.3 indicate that the building architecture level provides the greatest leverage for impacting building performance and generating associated improvements to energy efficiency. Indeed, with good judgment and "relatively little expense" at the frontend of the building lifecycle, the savings can be substantial. Research in product design indicates that about 75–80% of the product life-cycle cost is determined by decisions made during the conceptual stages of design. Moreover, a poor concept can rarely be compensated during the latter stages of design [17, 63]. One important complicating factor stems from buildings being highly multidisciplinary. We can only expect such a process to work well if we have a good understanding of how decisions within a disciplinary domain impact performance, as well as interactions across such domains. While architects and structural engineers have well-developed procedures for handling interactions between the geometry of a building and structural engineering mechanisms (e.g., choice of an appropriate structural system), procedures for handling the interaction between architectural geometry and energy efficiency are far less mature. Consequently, architects and designers will have difficulty in making decisions regarding the adequacy of a design and in choosing rationally among different design alternatives.

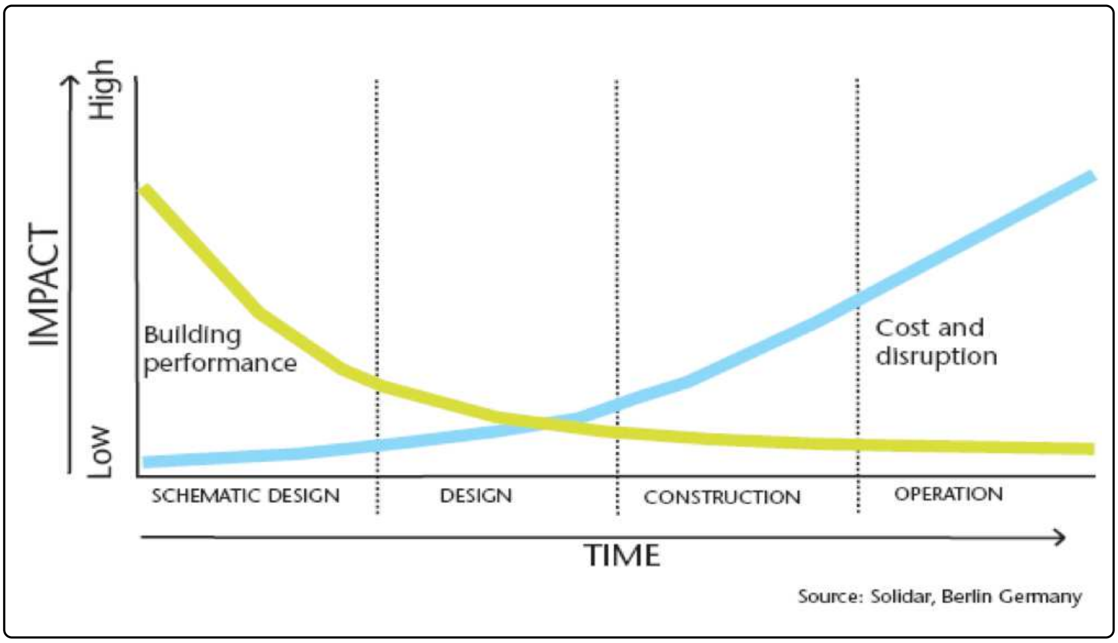


Figure 1.2: Opportunity for impacting building performance, and cost and disruption, at various stages in the building lifecycle. Source: Energy Efficiency in Buildings Summary – Business Realities and Opportunities, World Business Council for Sustainable Development.

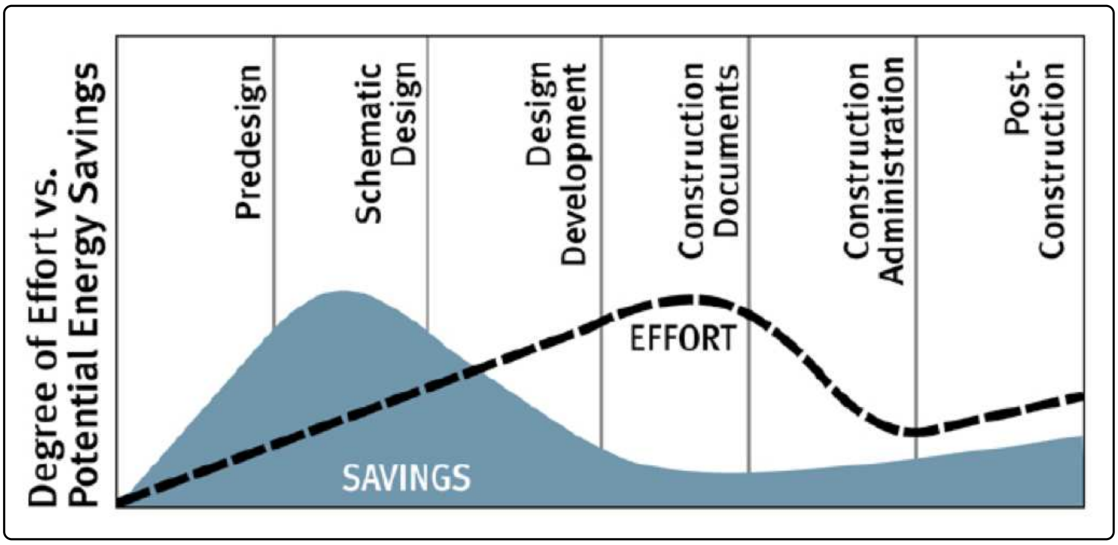
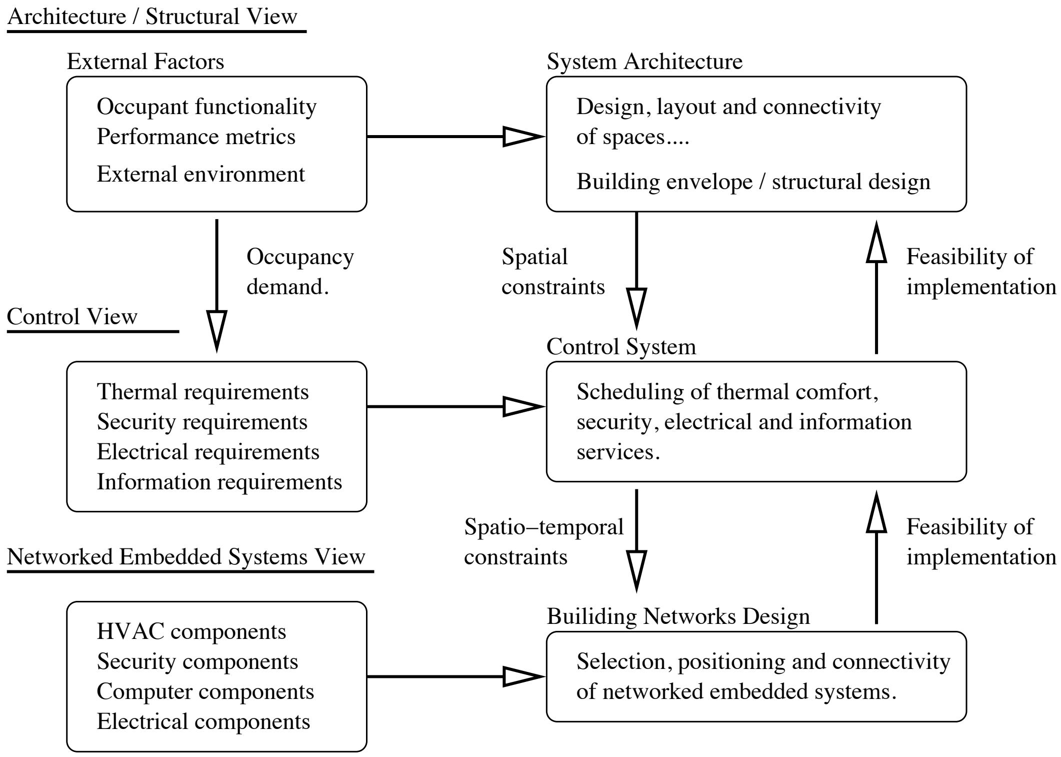


Figure 1.3: Degree of effort and potential savings at various stages of the building lifecycle. Source: Energy Efficiency in Buildings Summary – Business Realities and Opportunities, World Business Council for Sustainable Development.

1.3 State-of-the-Art Architectural Design of Buildings

Fundamentally, building architecture is about creating relationships that are geometric in nature [3, 24, 49, 57, 60, 66]. The architectural design process begins when a person or an institution has a problem that can be solved only by building. That person (the client) states his or her needs in a building program. During the earliest stages of design, architectural concerns are directed toward development of functional requirements and identification of relevant design rules, and economic and legal restrictions. The progressive transformation from required functionality (e.g., activities, uses, services) to form (spatial qualities, dimensions, use of materials, aesthetics) of a building's internal physical structure is a creative process that spans multiple disciplines. When new technologies come along (as happened during the industrial revolution) architects are given the freedom to design larger spans and more spacious buildings.

Established building design practice deals with the multidisciplinary nature of buildings with strategies of decision making that are sequential in their disciplinary application. Processes for the conceptual design of building architectures abstract from consideration the requirements that are not associated with the high-level functionality and aesthetics of the building. Consideration of the other participating disciplines (e.g., structural, mechanical and electrical engineering) is postponed until key decisions on the building architecture have been made. This leads to a graph of development activities shown in Figure 1.4. Traditional approaches to architectural design also tend to integrate measurable criteria only in



Architecture / Structural View
External Factors
Occupant functionality
Performance metrics
External environment
System Architecture
Design, layout and connectivity of spaces....
Building envelope / structural design
Occupancy demand.
Spatial constraints
Feasibility of implementation
Control View
Control System
Thermal requirements
Security requirements
Electrical requirements
Information requirements
Scheduling of thermal comfort, security, electrical and information services.
Spatio−temporal constraints
Feasibility of implementation
Networked Embedded Systems View
Builiding Networks Design
HVAC components
Security components
Computer components
Electrical components
Selection, positioning and connectivity of networked embedded systems.

Figure 1.4: Interaction of architectural, structural, control, and networked embedded system design activities (Source: Mark Austin [5]).

the later stages of the design process, leaving the assessment of design options in the early stages of the design process to experience and estimation of the designer [59].

Floorplan Definition and Modeling. A floorplan is a two-dimensional representation of a single floor of a building layout as as viewed from above. The plan shows placement of walls, doors and windows, fixtures for plumbing and electrical services, detail symbols and dimensions.

Figure 1.5 presents a framework for the multi-level development for building specifications, where architectural design can be viewed as a sequence of decisions and transformations designed to enclose usable (interior) space and support client

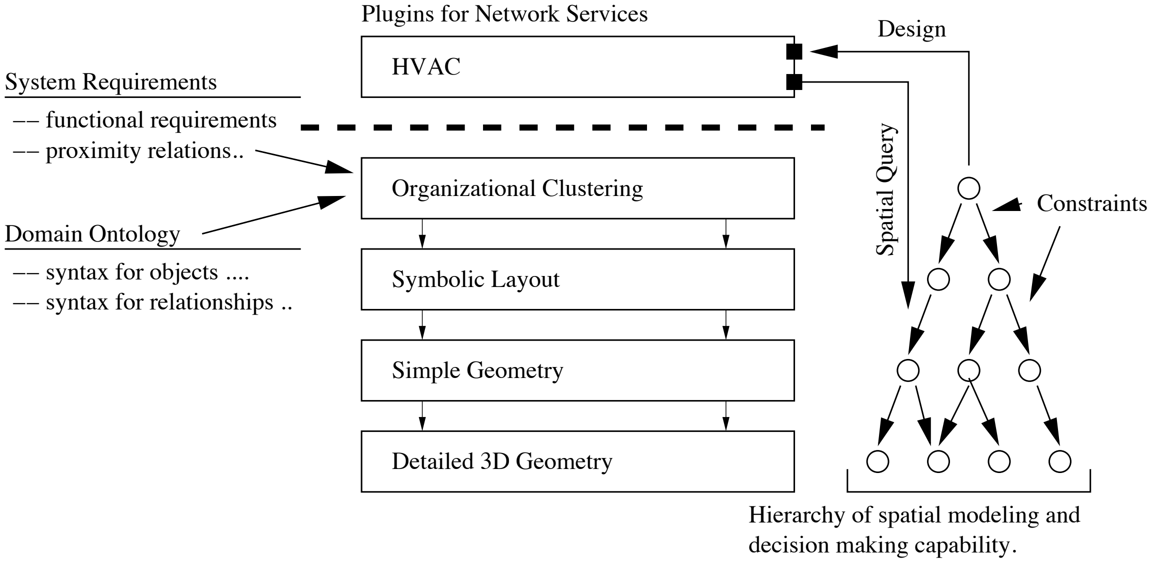

Plugins for Network Services
HVAC
Design
System Requirements
-- functional requirements
-- proximity relations..
Spatial Query
Organizational Clustering
Constraints
Domain Ontology
-- syntax for objects ....
-- syntax for relationships ..
Symbolic Layout
Simple Geometry
Detailed 3D Geometry
Hierarchy of spatial modeling and
decision making capability.

Figure 1.5: Framework for multi-level development for building architectures (spatial arrangements) augmented with network services (Adapted from Downs and Sequin [23, 54])

functions. According to the framework, floor plans are organized into levels of progressive detail. The organizational layer is concerned with clustering of spaces. The symbolic layout level focuses on room contours, symbolic representations of connected wall segments, and assignment of properties to regions. Simple geometry corresponds to thick walls, fleshed-out columns, cut-out doors and windows.

System behavior is enabled by the ability of the building occupants to function – the latter emanates from two sources: (1) functionality enabled by spaces and access to spaces, and (2) networked services (e.g, electrical, environmental micro-climates, security, etc.) integrated into the architectural domain. While many of these issues can be resolved with approximate/imprecise models of the final components to be used [19], it is important to note that few opportunities exist to test the final product prior to its full implementation. Therefore, formal mechanisms that will enable early validation of designer intent and design rule checking can vastly improve the quality and reliability of the building system prior to deployment.

Floorplan Design for a Small Apartment Building. Figure 1.6 shows, for example, how floor-level design for a small apartment building can be organized into layers of development, each corresponding to a region decomposition:

Level 0. Starting with a list of specific spaces (apartments, stairwell access) with planned area, a top-level design is defined by wall lines for the building exterior.

Level 1. Next, the building exterior is partitioned into regions for the individual apartments, plus space for the corridors and stairwells. Initially walls are defined by their centerlines and drawn as a single line.

Level 2. Regions for individual apartments are divided into rooms – kitchen, living room, bathroom, bedrooms. Spaces are connected with openings in walls. Doorways and windows are added to walls. Attributes are applied to walls (e.g., thickness). Walls are drawn as double parallel lines.

At each level of development a primary goal is to organize the spaces so that they satisfy orientation, topological and access constraints. For example, at level 1 each apartment needs to be provided access to the stairwell. Within the individual apartments bedrooms should face open air. High-level topological constraints that facilitate occupant functionality include factors like, (1) The kitchen should be close to the living room, (2) You should be able to go from the bedroom to the bathroom without having to pass through the kitchen, and (3) The bedrooms should not be

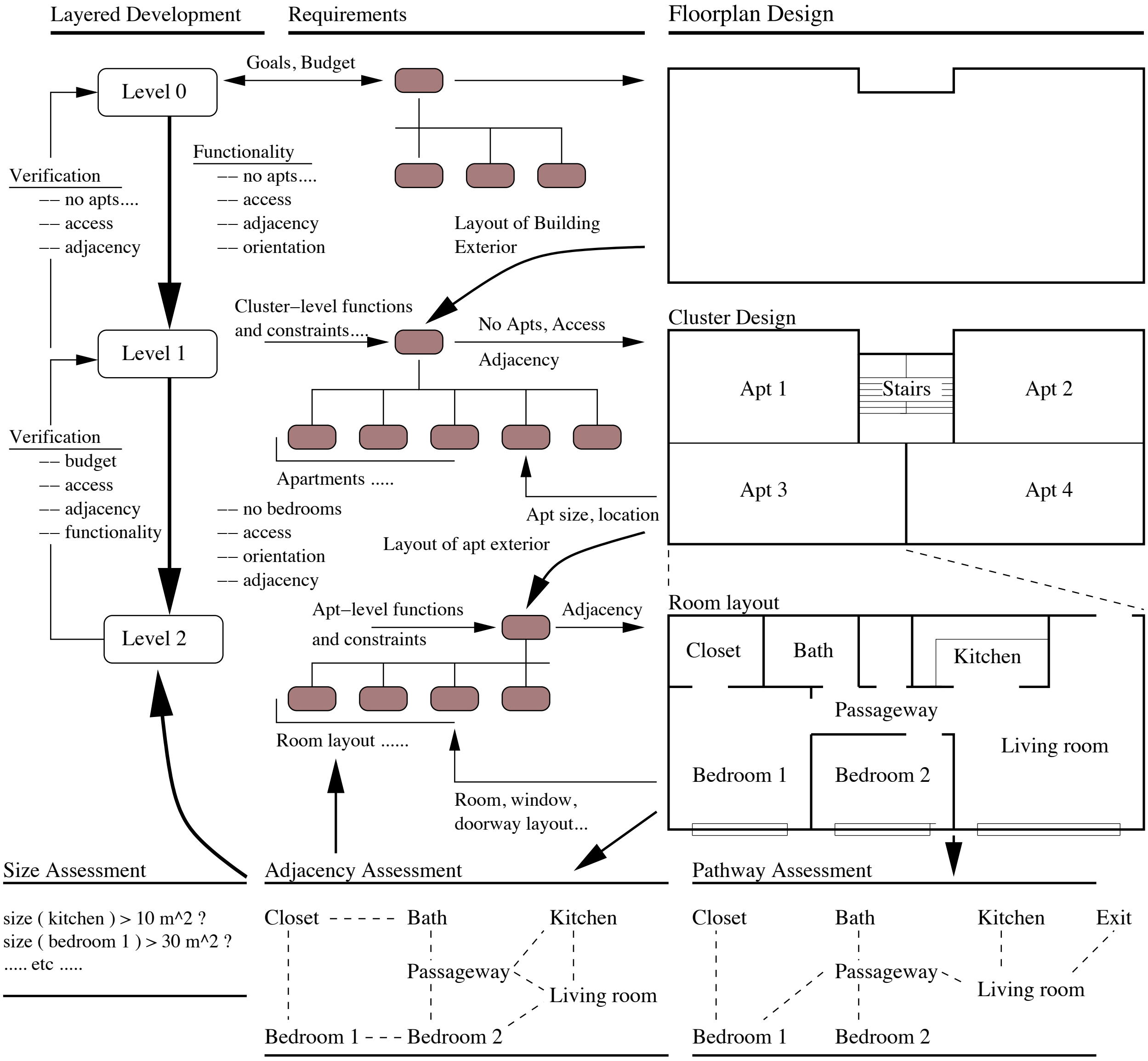


Figure 1.6: Progressive decomposition of architectural floorplans for a small apartment building. Adjacency assessment means spatial adjacency assessment – two spaces can be adjacent and separated by a wall. Pathway assessment is defined by the layout of the walls.

immediately adjacent to the entrance.

Beyond level 2 the tops of doorways and windows are defined by "header lines." The bottoms of windows are defined by "sill lines." Plumbing and electrical fixtures are added and connected to services with appropriate wiring/piping details. Dimensions and text are added to the floor plans. If the required functionality at lower levels of development cannot be satisfied (perhaps because the constraint values are too stringent), then the verification process will fail and the high-level developments will need to be adjusted to accommodate the demands of the lower level requirements (e.g., perhaps the overall size of an apartment would need to be increased).

1.4 Building Information Modeling (BIM)

During the past three decades the architectural design profession has benefited from the development of software to support building information modeling (BIM) and computer-aided architectural design (CAAD). In a departure from past practices, which have focused on the production of detailed design documents (drawings of the building plan, elevation and sections) to describe what a building should look like when complete, the general idea of BIM is that architects and engineers will create models of buildings, and drawings and support documents will be views of the models. If an element (e.g., a door or window) is repositioned in the model, then all of the views containing that element be automatically updated.

Commercial tools such as AutoCAD [48], 3D Home Architect and TurboCAD Professional [12, 50] focus on the editing and presentation of Architectural/Mechanical CAD models/plans as blueprint-like drawings, 2-D designer viewpoints, and 3-D photorealistc renderings. Medium-end versions include support for pre-defined domain-specific features (e.g., architectural symbols), dynamic dimensioning, basic solid and surface modeling (i.e., boolean operations and slicing), collision detection, and cost estimation. High-end versions are linked to external packages for performance-based assessment through the use of simulation (e.g., energy simulation; solar studies; egress analysis). They also support team development of projects [47].

Capabilities of Present-Day BIM. Building information models are compelling because they enable processes, facilitate communication across disciplines, reduce the likelihood of errors, and provide a pathway to the automatic enforcement of standards. BIM coverage includes representation of building components, data attributes, parametric rules, and support for consistent management and propagation of all non-redundant data to all views of the model [9, 25]. Simple drawing errors (e.g., drawing something upside down) can be eliminated. By incorporating information on the project cost and schedule into the models, BIMs can be of great benefit to project managers.

Parametric design systems model a design as a collection of mathematically constrained entities. Designers work with such systems on two levels: (1) definition of the entities and the relationships among them, and (2) search within a design space [3]. Parametric constraints can be used within a model, for example, to

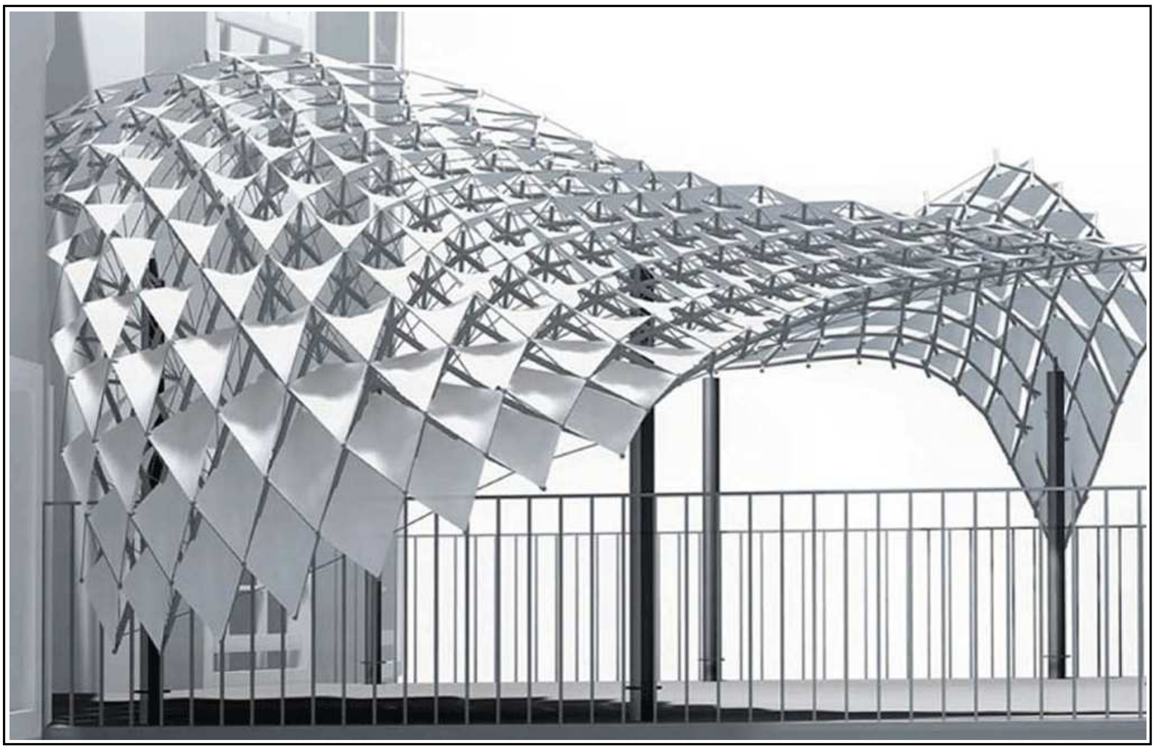


Figure 1.7: Parametric representation for a generated compound membrane [10].

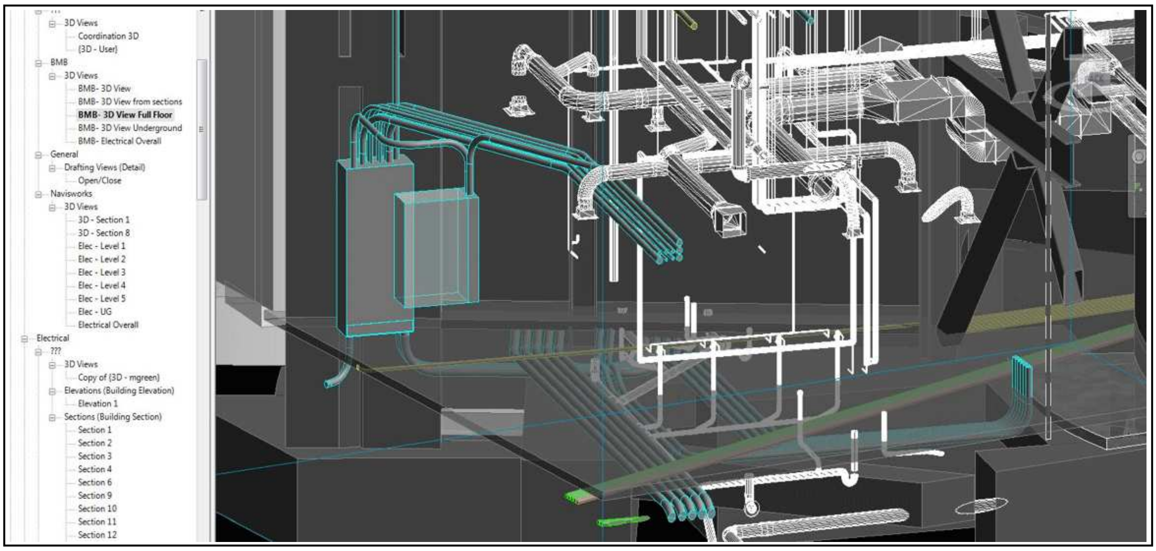


Figure 1.8: Tree hierarchy organization and three-dimensional visualization of HVAC systems layout [11].

enforce compatibilities among connected objects [53]. An emerging application of parametric modeling is to support the generation of free-form architectures. Figure 1.7 shows, for example, the results of a parametrically generated compound (shell) membrane. Propagation-based systems are the simplest type of constraint-based system [4]. At the building architecture level, propagation-based constraint systems comprise an acyclic directed graph and two algorithms: one for ordering the graph (i.e., a topological sort of dependency relationships in the graph), and a second for propagating values through the graph. Figure 1.8 shows how a three-dimensional visualization for a building and its contents can be organized into a tree hierarchy. The latter is a very convenient and efficient mechanisms for defining and searching the space of design options.

As of 2014, the BIM market leaders are: (1) Autodesk with Revit Architecture [47], and (2) Bentley Systems with a suite of complementary software (e.g., Bentley Architecture, Bentley Structural, Bentley Mechanical Building Systems). Both companies offer suites of discipline-specific BIM authoring software that is built on a common BIM platform and, thus, internally interoperable. A key feature of current mainstream BIM authoring software is the ability to define parametric constraints to enforce relationships on the geometry of objects. On one hand, providing support for parametric relationships increases the complexity of creating designs because both the entities and their mathematical relationship need to be represented. On the other hand, once such relationships are in place, parametric relationships facilitate exploration of the design space and possibly discovery of

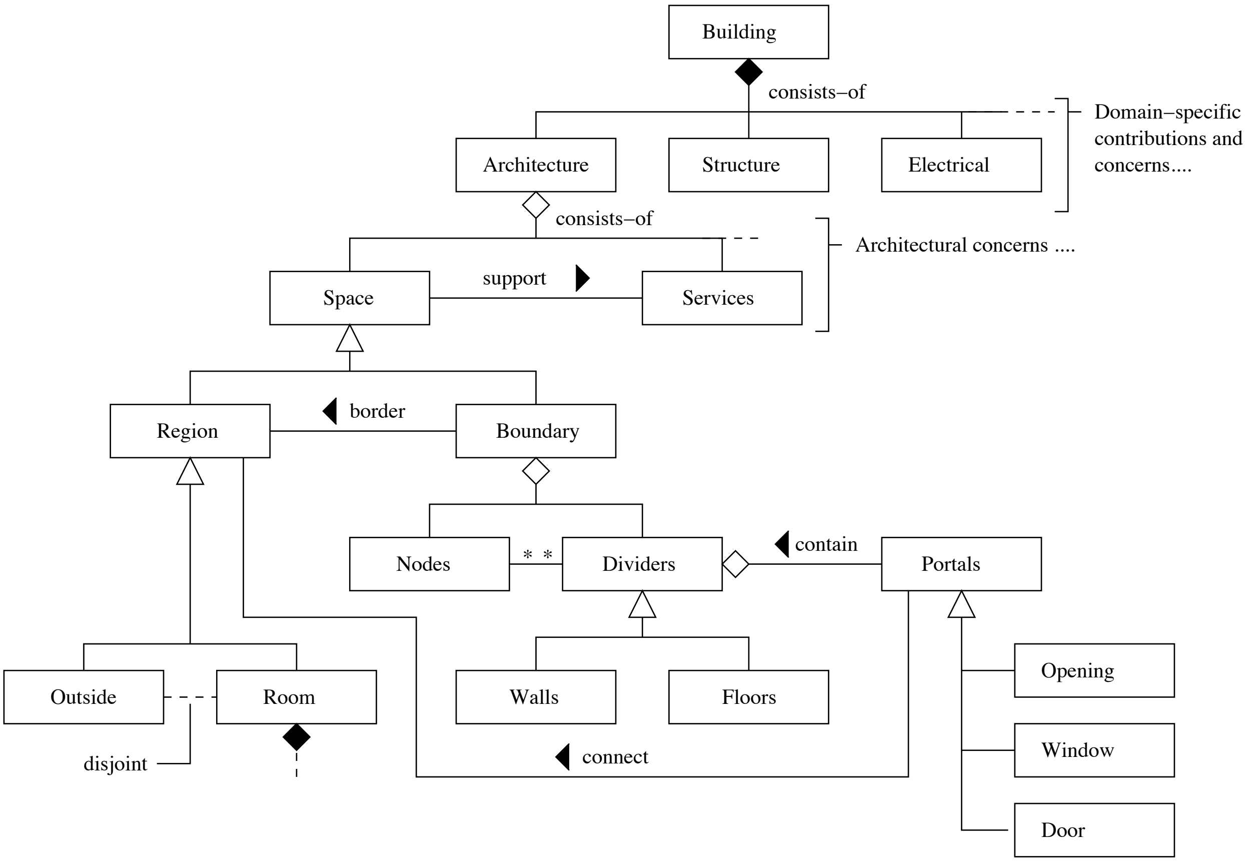


Figure 1.9: Class diagram for the decomposition, layout, and connectivity of spaces in a building.

new forms. If the participating disciplines can understand the nature of the graph dependency relationships connecting entities (possible across disciplines), then the graph provides a platform for communication among disciplines, and exploration and assessment of design options [39].

Weaknesses of Present-Day BIM. Despite these successes, present-day BIM is directed mainly at the design phase and is compartmentalized into discipline-specific software packages. In theory, communication among software packages is enabled by BIM interoperability standards, but in practice, competitive pressures prevent all vendors from adopting a single uniform standard. Present-day building design processes are weak in their support of models capable of linking fragments of behavior to system components, and at expressing dependencies and interdependencies among disciplines – so-called multi-aspect design – from a variety of viewpoints [18]. Comprehensive descriptions of the building system structure are complicated by mixtures of hierarchy, network, and association relationships, as illustrated in Figure 1.9. Propagation-constraint relationships occur due to hierarchies of abstractions (see Figures 1.4, 1.5 and 1.6) and the complementary roles served by solid-filled and void spaces. Furthermore, a single object (or subsystem) may be required to satisfy the functional concerns of multiple disciplines [52].

Most software implementations (and their underlying data representations) skirt these issues by assuming that one viewpoint takes priority. Support for the remaining viewpoint(s) may be much weaker (see, for example, references [23, 32, 45, 54]). With its focus on generation and visualization of surface models, even Google SketchUp favors the presentation of architectural forms (drawings and documents) over support for formal approaches to geometric representation and reasoning. Indeed, current practice for mechanical, plumbing and electrical systems coordination is for design consultants to design each system independently. The coordination process is slow and expensive, in part because only minor advances are made at each step, and because it is human intensive [8, 46].

1.5 Model-Based Systems Engineering for Building Systems Design

Model-based systems engineering (MBSE) development is an approach to systems-level development in which the focus and primary artifacts of development are models (as opposed to documents). Methodologies for the model-based design, management, and evolution of whole building systems need to cover system functionality (what will the system do?), evaluation of system performance (how well will it perform?), system validation and verification (how can we make sure the system will actually work?), and economics (how much will it cost?) [52].

In contrast to present-day BIM, which has a heavy focus on visualization and support for project management activities, MBSE aims to capture the multidisciplinary aspects of system structure and behavior in building systems design and operation. As such, MBSE procedures hold the promise of being ideally suited to the demands of future building design where superior levels of performance and comfort will cause energy and control systems to become much more integrated and complex than today [67].

Experience [30] tell us that good solutions are likely to employ a combination of semi-formal models (e.g., SysML), formal models, abstraction mechanisms, top-down decomposition and bottom-up composition, and deal with design concerns through strategies of: (1) separation of concerns, (2) breadth before depth, and (3) function-architecture co-design. When the elements of a design are easily separated, as shown in Figure 1.10, step-by-step procedures can be followed for the development of models for system behavior, structure, design, and evaluation and ranking of alternatives. We believe that as the complexity of a system increases (or becomes intertwined), high levels of productivity in system development can be achieved through the use of high-level visual abstractions coupled with lower-level (mathematical) abstractions suitable for formal systems analysis [6, 7, 29]. This multi-level framework is illustrated in Figure 1.11. Semi-formal abstractions provide efficiency in a "big picture" representation of the system under development. They highlight the major components, their organization (layout, decomposition), and connectivity to nearby components. The lower-level abstractions are suitable for formal systems analysis – for example, verification of component interface compatibilities and/or assessment of system performance through the use of simulation methods.

While these practices are in common use in the development of engineering products, for the architectural design of buildings, completion of these tasks is complicated by: (1) the multitude of organizational and propagation-constraint relationships that need to be supported, and (2) the necessity of performance-based design and real-time management of buildings, serving many stakeholders over extended periods of time. As a result, the number of design options that are generated may be very small, meaning that architects may consider on a very narrow range of candidate designs. This situation is less than ideal. The move toward performance-based design and management of buildings means that models of system behavior will need to be explicitly considered. Figure 1.12 shows, for example, energy exchange and material flow relationships in a simple one-room building. The activity diagram

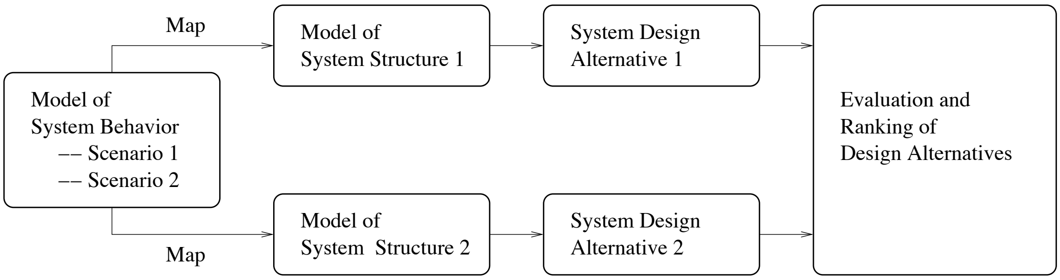


Figure 1.10: Mapping models of system behavior onto system structure alternatives.

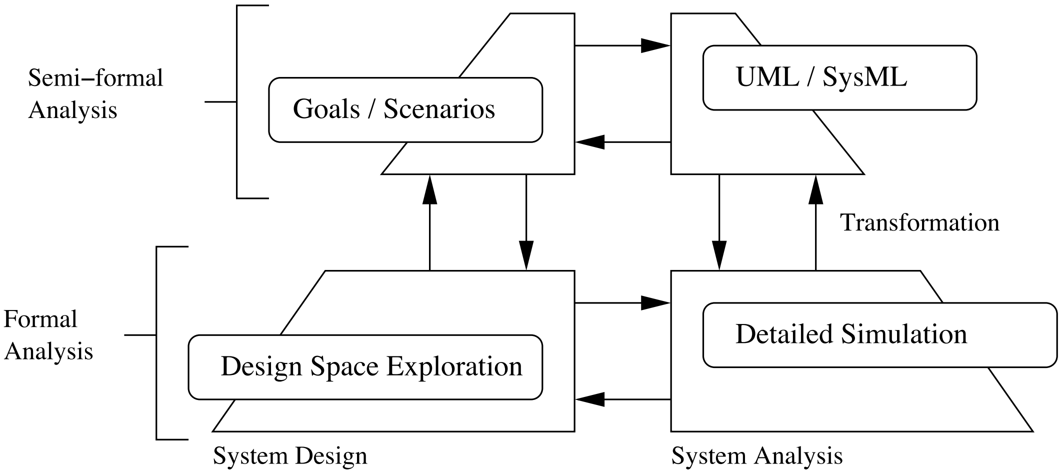


Figure 1.11: Multi-level approach to model-based systems engineering.

has five horizontal process boxes for radiation, thermal, electric, air, and materials and emissions. The vertical swim lanes represent an allocation (or mapping) of the processes to four subsystems, the environment, envelope, building technology and indoor space. Figuring out how to scale this capability up to buildings of a realistic size is a significant long-term challenge.

1.6 Objectives and Scope of this Thesis

The long-term objectives of this research are development of model-based systems engineering (MBSE) procedures and computer-aided tools for the paramet-

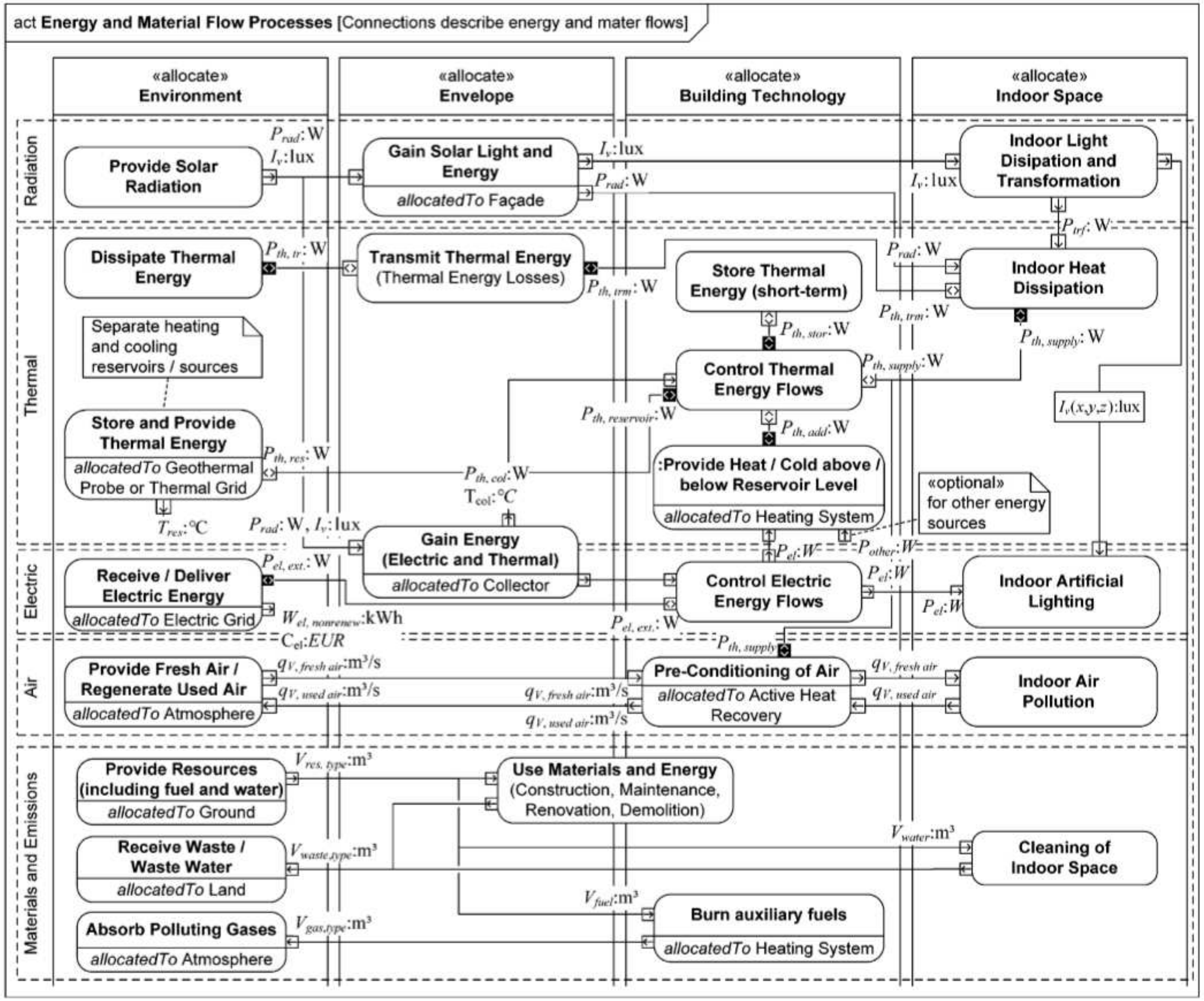

act Energy and Material Flow Processes [Connections describe energy and mater flows]
«allocate» Environment
«allocate» Envelope
«allocate» Building Technology
«allocate» Indoor Space
Radiation
Thermal
Electric
Air
Materials and Emissions
Provide Solar Radiation
Gain Solar Light and Energy
allocatedTo Façade
Indoor Light Disipation and Transformation
Dissipate Thermal Energy
Transmit Thermal Energy (Thermal Energy Losses)
Store Thermal Energy (short-term)
Indoor Heat Dissipation
Separate heating and cooling reservoirs / sources
Control Thermal Energy Flows
Store and Provide Thermal Energy
allocatedTo Geothermal Probe or Thermal Grid
:Provide Heat / Cold above / below Reservoir Level
allocatedTo Heating System
«optional» for other energy sources
Gain Energy (Electric and Thermal)
allocatedTo Collector
Receive / Deliver Electric Energy
allocatedTo Electric Grid
Control Electric Energy Flows
Indoor Artificial Lighting
Provide Fresh Air / Regenerate Used Air
allocatedTo Atmosphere
Pre-Conditioning of Air
allocatedTo Active Heat Recovery
Indoor Air Pollution
Provide Resources (including fuel and water)
allocatedTo Ground
Use Materials and Energy (Construction, Maintenance, Renovation, Demolition)
Receive Waste / Waste Water
allocatedTo Land
Cleaning of Indoor Space
Absorb Polluting Gases
allocatedTo Atmosphere
Burn auxiliary fuels
allocatedTo Heating System
P_{rad}: W
I_v: lux
I_v: lux
P_{rad}: W
I_v: lux
P_{trf}: W
$P_{th,\,tr}$: W
$P_{th,\,trm}$: W
P_{rad}: W
$P_{th,\,trm}$: W
$P_{th,\,stor}$: W
$P_{th,\,supply}$: W
$P_{th,\,supply}$: W
$P_{th,\,reservoir}$: W
$P_{th,\,add}$: W
$I_v(x,y,z)$: lux
$P_{th,\,res}$: W
$P_{th,\,col}$: W
T_{col}: °C
T_{res}: °C
P_{rad}: W, I_v: lux
P_{el}: W
P_{other}: W
$P_{el,\,ext.}$: W
P_{el}: W
P_{el}: W
$W_{el,\,nonrenew}$: kWh
C_{el}: EUR
$P_{el,\,ext.}$: W
$P_{th,\,supply}$
$q_{V,\,fresh\,air}$: m³/s
$q_{V,\,used\,air}$: m³/s
$q_{V,\,fresh\,air}$: m³/s
$q_{V,\,used\,air}$: m³/s
$q_{V,\,fresh\,air}$
$q_{V,\,used\,air}$
$V_{res,\,type}$: m³
V_{water}: m³
$V_{waste,type}$: m³
V_{fuel}: m³
$V_{gas,type}$: m³

Figure 1.12: Activity diagram for traces of energy and material flows within a building (Adapted from Geyer [34]).

ric modeling, system-level assessment, and trade-study analysis of buildings. The immediate goal of this research project is to take a first step toward providing this capability and, specifically, to understand how top-down parametric representations of building floorplans can be created, and then systematically adjusted to cover design spaces. To deal with large number of components and multiplicity of dependency relationships in representations of buildings, we are interested in understanding how software design patterns (specifically, composite hierarchies, model-view-controller, observer, mediator and visitor design patterns) can facilitate the organization, modularity, and visualization of building architecture representations. With a prototype of software having this capability in place, the second objective of this thesis is to develop a preliminary framework for architecture-energy assessment of building floorplans.

Chapter 2 focuses on models and software design patterns for the top-down parametric modeling of floorplans. Chapter 3 presents a scripting approach to floorplan specification. We will see that while the method works, the step-by-step procedure for creating the floorplan models and all of the dependency relationships is tedious. And it certainly isn't scalable – a relatively small two apartment floorplan requires approximately 2,000 lines of Java. In an effort the overcome this shortcoming, Chapter 4 presents a graphical approach to floorplan specification. Figure 1.13 shows the essential details of the system architecture and composite hierarchy framework for modeling and visualizing building floorplans. We employ the MVC software design pattern to links the models, views and controllers together. To sim-

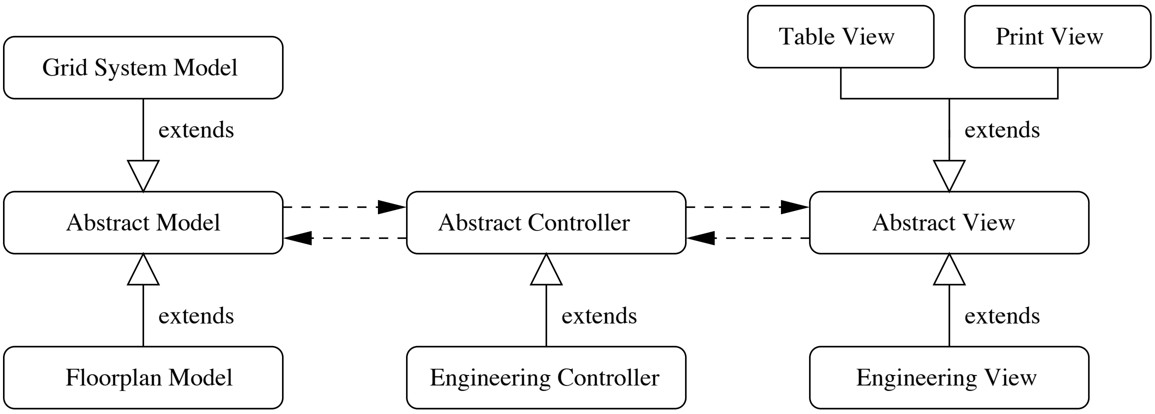


Figure 1.13: Implementation of the MVC software design pattern using mixtures of abstract and implementation-specific classes.

plify the details of implementation, the discipline-specific models, views and controllers are concrete extensions of abstract implementations. The primary purpose of the abstract-level specifications is to take care of the details of model-controller and view-controller communication. Chapter 5 presents case studies for a full system analysis of the “two apartment units” floorplan model introduced in Chapters 3 and 4. Simplified system analyses are provided for building code requirements verification and architecture-energy system assessment and tradeoff analysis. Chapter 6 presents the conclusions and makes suggestions for future research.

Chapter 2

Parametric Modeling of Building Floorplans

This chapter presents the formulation of parametric models for building floorplans. Parametric models are provided for individual elements, relationships among elements, and dependency relationships between layers of abstraction. Together, they work to allow for the evaluation of both the "spatial elements of a floor plan" as well as the walls defining the floorplan topology and geometry. The second purpose of this chapter is to introduce software design patterns that can be used in the implementation of software for parametric modeling of building floorplas. Finally, this chapter demonstrates how the Java Topology Suite can be employed for area computations of building floorplans.

2.1 Preliminary Concepts

From an architectural standpoint, parametric modeling is the process of putting together a geometric representation of a design using components and attributes that have been parameterized. While procedures for parametric modeling for mechanical CAD are quite mature, parametric models for building architecture presently are challenged by the scale and complexity of geometric and dependency relationships. For the conceptual design of buildings, these relationship can be naturally organized into a hierarchy of dependencies. Some of the parameters will act as independent variables (i.e., as system inputs) others will take values evaluated through the chain of dependency relationships. Parametric design creates the possibility of generating large sets of design alternatives that are both practical and appealing.

As already explained in Chapter 1, engineers are first concerned with the building as a whole, next with the pieces that make up that whole, the connections among them, and lastly the details of each individual object. When changes are made to the building at a certain level, all of the elements at the lower levels should adapt to the changes made, thereby maintaining the spatial integrity [53]. If it will work, a key benefit in this approach to building specification is that engineers can explore design spaces by playing with building layouts at a high level of abstraction. Therefore, using parametric modeling for components makes a real impact to productivity of the engineering project.

2.2 Parametric Modeling of Floorplans

2.2.1 Organization of Floorplans into Hierarchy of Models

Multi-layer hierarchy has been widely implemented in engineering systems design. For the parametric modeling of building floorplans, the performance of traditional floorplan design can be enhanced with models that are organized into hierarchies. This research employs the three-layer hierarchy shown in Figure 2.1:

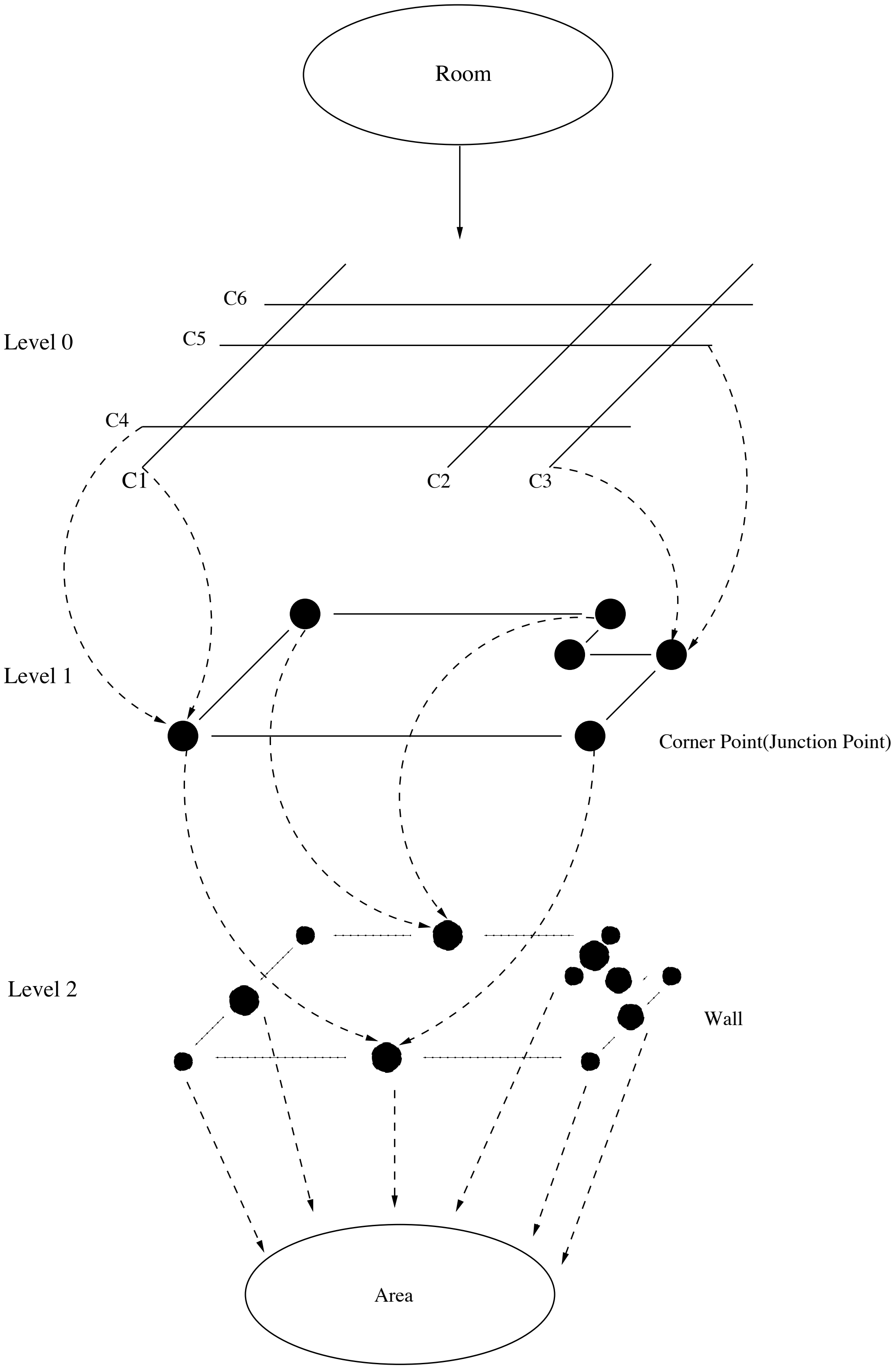


Figure 2.1: Propagation of dependency relationship representation for a simple building model.

- **Level 0. Centerline Layer.** In a two-dimensional floor plan, sets of centerlines representing the position of structural frames and bays are created in the horizontal and vertical directions.

- **Level 1. Junction Points Layer.** The junction point layer is defined by items positioned at the intersection of orthogonal centerlines. The junction points inherit the centerline coordinates and are defined by rectangle blocks having a predefined width.

- **Level 2. Wall Layer.** The wall layer represents the interior and exterior walls of a building system.

The propagation of dependency relationships between adjacent layers will serve as supplement for reductionism. Components in the high-level layers (e.g., layer 0) can effortlessly propogate data (e.g., locations, width, and height) to items in the lower-level layers. In Figure 2.1, the large dots represent junction points and columns that have been positioned at the intersection of centerlines. Once the walls are in place, the area of rooms can be computed.

2.2.2 Modeling Many-to-Many Association Relationships

The modeling of building floorplans is complicated by a multitude of many-to-many association relationships between entities. In software, many-to-many relationship between classes A and B exists when ...

... multiple objects of type A are associated with multiple objects of type B, and visa versa.

Many-to-many relationships exist between entities that participate in assembly relationships.

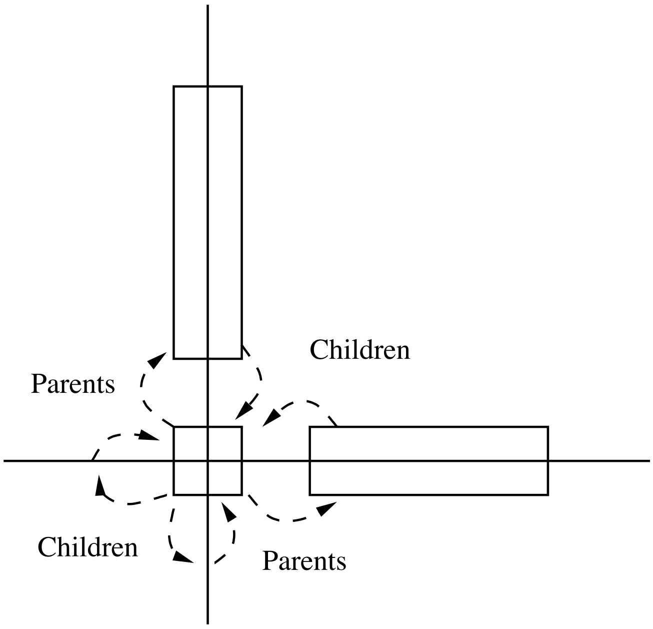


Figure 2.2: Many-to-many relationship representation for three different type of building object model.

Figure 2.2 shows, for example, the relationship between the three building object model. Notice that both vertical and horizontal centerline became the parent and generated the junction point on their intersection. Then, the junction point became the parent of it's upper wall element and right wall element. In this case, the location of the junction point was defined by those two centerlines, and the width and height of the walls were determined by two ends of the wall element, which is the junction points.

2.3 Area Computation with the Java Topology Suite

Area computations are an important part of building system analysis because construction costs and energy system costs are roughly correlated to areas. In this research, area computations for each building object or building system are accomplished with the Java Topology Suite (JTS) library [41]. The JTS Topology Suite is an API of 2D spatial predicates and functions that provides a complete, consistent, and robust implementation of fundamental 2D spatial algorithms. As such, the library provides a methodology to compute the area and to determine the relationship of geometries. For the purposes of this study, the Java Topology Suite is used to compute the area of rooms, which may have a complicated interior wall design or non-rectangular shape.

Basic Geometry Operations. Listed below are the basic geometry methods from the JTS that are used in this study to compute the usable area of building floorplans. They are:

- **union():** With an input of a Geometry object, this method will return a combined Geometry object. By using this method, the software can calculate the combined geometry of the unusable area in floorplan model, such as the combination of walls and columns.
- **difference():** This method returns the difference between two polygon objects/areas. In this study, the method is used to compute the difference of the whole floorplan area and the unusable area geometry.

- **getArea():** After getting the usable Geometry object from the difference method, this method will return a quantitative value for the usable area.

- **getCoordinates():** This method returns a list of sequential vertices which can be used to calculate the width of the geometry.

The next section will demonstrate some example room area calculations by implementing the Java Topology Suite library (JTS).

2.3.1 Example 1. A Simple Area Computation

The following snippets of source code show the step-by-step procedure for defining the polygon in Figure 2.3 and computing it's area with JTS.

Step 1. Create polygon vertices with a coordinate array.

code segment

```
Coordinate pt1 = new Coordinate(  0,  0, 0 );
Coordinate pt2 = new Coordinate(  0, 50, 0 );
Coordinate pt3 = new Coordinate( 50, 50, 0 );
Coordinate pt4 = new Coordinate( 50,  0, 0 );

List<Coordinate>  points = new ArrayList<Coordinate>();

points.add( pt1 ); points.add( pt2 );
points.add( pt3 ); points.add( pt4 );
points.add( pt1 );

Coordinate coordinates[] = points.toArray( new Coordinate[ points.size() ] );
```

In the first four lines of the script, objects of type Coordinate are created and instantiated with the (x,y) coordinate values for the test polygon. Next, an array list of coordinates is created. The coordinate points (i.e., pt1, pt2, pt3 and pt4)

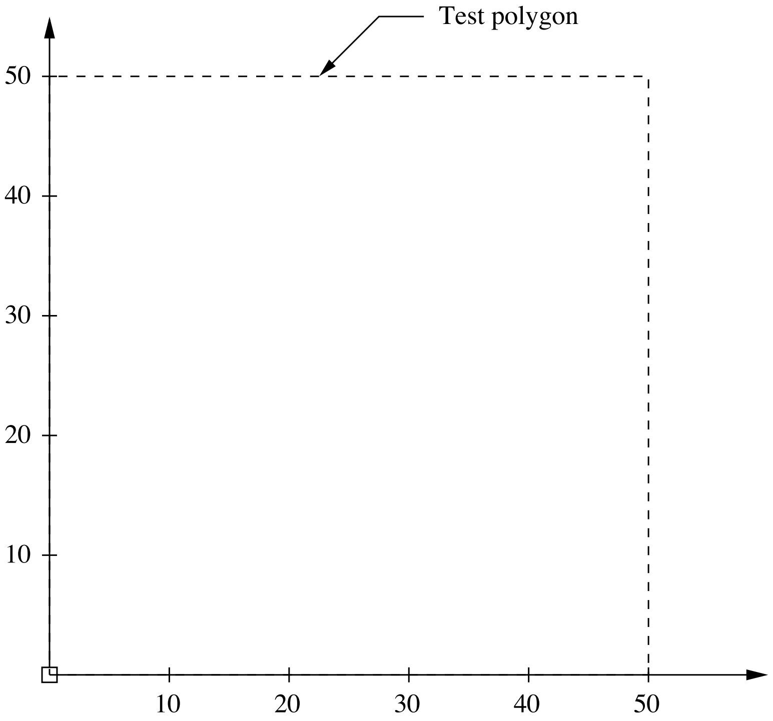


Figure 2.3: Simple area comutation with JTS.

are added to the array list. Finally, the array list is converted into an array data structure.

Step 2. Create the GeometryFactory object.

code segment

```
GeometryFactory fact = new GeometryFactory();
```

The GeometryFactory provides users with numerous methods to generate Geometry objects.

Step 3. Use the LinearRing method and the array of coordinate values to create an outer boundary for the polygon geometry.

code segment

```
Polygon polygon = new Polygon(
                         fact.createLinearRing( coordinates ), null, fact
```

```
                        );
```

While orientation of the polygon exterior may be clockwise or anticlockwise, the line segments must not intersection (i.e., in other words, the polygon needs to be a simple polygon).

Step 4. Utilize the getArea() method that provided by JTS library and print out the area size which is calculated by the method.

code segment

```
System.out.println( polygon );
double area = polygon.getArea();
System.out.println( "Area = " + area );
```

Since each side of the rectangle is 50, one can calculate the area by hand with the result of 2500. By comparing the JTS result below with the hand calculated result, one can say that JTS library has the capability to do area calculation.

result scripts

```
POLYGON ((0 0, 0 50, 50 50, 50 0, 0 0))
Area = 2500.0
```

2.3.2 Example 2. Set Operations + Area Computation

The following description and source code shows the step-by-step procedure for defining the collection of polygons shown in Figure 2.4, and then computing set operations for area computation. Other than the different location for all vertiics of the polygon, the source code creation for each polygon in Example 2 is almost

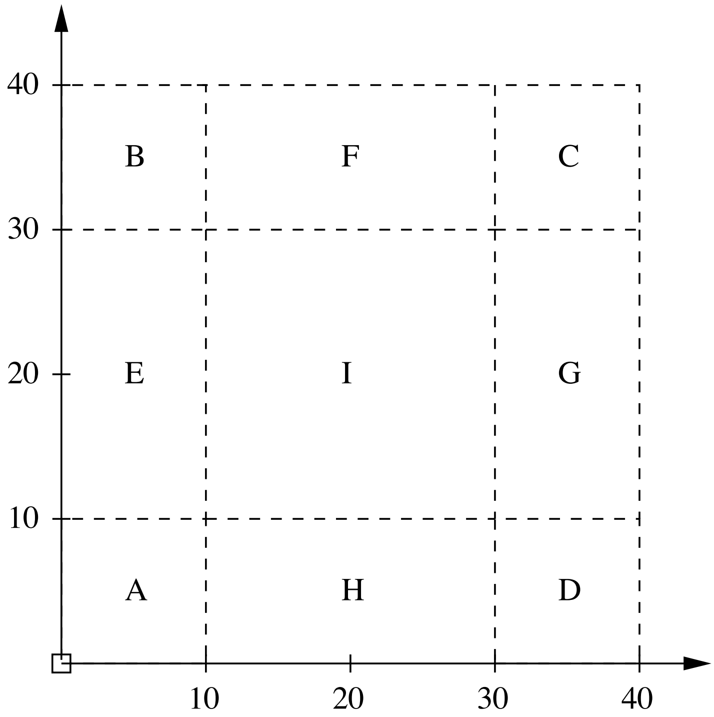


Figure 2.4: Set operations and area computations with JTS.

identical as the source code shown in Example 1, Steps 1 through 4. Therefore, the fragements of source code shown below will focus on outputs of the set operation computations.

Step 1. Create 9 polygons as shown in Figure 2.4 by using the same procedure as in Section 2.4.1. The polygon coordinates and area computations are as follows:

result scripts

```
Polygon A : POLYGON ((0 0, 0 10, 10 10, 10 0, 0 0))
Polygon B : POLYGON ((0 30, 0 40, 10 40, 10 30, 0 30))
Polygon C : POLYGON ((30 30, 30 40, 40 40, 40 30, 30 30))
Polygon D : POLYGON ((30 0, 30 10, 40 10, 40 0, 30 0))
Polygon E : POLYGON ((0 10, 0 30, 10 30, 10 10, 0 10))
Polygon F : POLYGON ((10 30, 10 40, 30 40, 30 30, 10 30))
Polygon G : POLYGON ((30 10, 30 30, 40 30, 40 10, 30 10))
Polygon H : POLYGON ((10 0, 10 10, 30 10, 30 0, 10 0))
Polygon I : POLYGON ((10 10, 10 30, 30 30, 30 10, 10 10))

Area A = 100.0, Area B = 100.0, Area C = 100.0
Area D = 100.0, Area E = 200.0, Area F = 200.0
Area G = 200.0, Area H = 200.0, Area I = 400.0
```

Step 2. Add up the geometry of polygon A and E. This step allows the user to have a better idea of how simple union operation works by using JTS library.

code segment

```
   Geometry aUnionE = polygonA.union( polygonE );
   System.out.println( aUnionE );
   double areaAUnionE = aUnionE.getArea();
   System.out.println( "Area A union E = " + areaAUnionE );
```

As one can observe in the output,

result scripts

```
POLYGON ((0 0, 0 10, 0 30, 10 30, 10 10, 10 0, 0 0))
Area A union E = 300.0
```

A union E geometry and the area set operation result provides a strong evidence that JTS is capable of computing union set operations.

Step 3. Compute the union all outer polygons and calculate the area. The result is as follows:

result scripts

```
POLYGON ((0 0, 0 10, 0 30, 0 40, 10 40, 30 40, 40 40,
40 30, 40 10, 40 0, 30 0, 10 0, 0 0),
(30 10, 30 30, 10 30, 10 10, 30 10))
The union of all outer polygons = 1200.0
```

Since the union of the outer polygons is a polygon with an internal ring, the printed result displays two lists of verticies. One is for the outer polygon, and the other is for the inner polygon.

Step 4. Compute the union of all polygons and calculate the total area. The results are as follows:

result scripts

```
POLYGON ((0 0, 0 10, 0 30, 0 40, 10 40, 30 40, 40 40,
40 30, 40 10, 40 0, 30 0, 10 0, 0 0))
Whole area = 1600.0
```

Step 5. Compute the difference between the whole geometry and the union of outer polygons.

code segment

```
   Geometry diffrence = all.difference( outerUnion );
   System.out.println( diffrence );
   double centerArea = diffrence.getArea();
   System.out.println( "Center area = " + centerArea );
```

The output:

result scripts

```
POLYGON ((30 10, 10 10, 10 30, 30 30, 30 10))
Center area = 400.0
Area I = 400.0
```

is the area of polygon I, thereby demonstrating that JTS can be used for the computation of simple geometry operations. Later on, we will use exactly the same procedure to compute the usable area of more complicated building floorplans.

2.4 Software Design Patterns

Software design patterns are defined as general repeatable solutions to common software design problems; designers customize these templates to suit the design requirements.

Behavior	**Structure**	**System**
Command	Adapter	**Model-View-Controller**
Interpreter	Bridge	Session
Mediator	**Composite**	Router
Observer	Decorator	Transaction
Visitor		

Table 2.1: Taxonomy of commonly used software design patterns. Those highlighted in bold are relevant to building floorplan modeling.

Table 2.1 contains a summary of the software design patterns – a novel mix of model-view-controller, mediator, composite hierarchy, observer and visitor design patterns – employed in this project.

Model-View-Controller Design Pattern. The model-view-controller (MVC) is an architectural design pattern which can be divided into three logical parts (the model, view, and controller), and as pointed out by Fowler [33], was one of the first attempts at addressing large-scale user-interface development in a systematic way. Fundamentally, the MVC provides a separation between domain objects and presentation objects. The domain objects is the model that contains all the business logic but no visualization, and the presentation objects contains the logic which represents the GUI elements. Controllers receive updates from models and forward the appropriate data to the presentation views.

The model, view, and controller specifications work together to ensure the data and information that is stored in the models and displayed in various views is consistent and fully synchronized. Figure 2.5 elaborates two approaches for MVC implementation. In the most common implementation of the MVC design pattern (see, for example, the Java patterns in Stelting and Maasson [56]), views register for their intent to be notified when changes to a model occur. Controllers register their interest in being notified of changes to a view. When a change occurs in the view, the view (graphical user interface) will query the model state and call the controller if the model needs to be modified. The controller then makes the modification. Finally the model notifies the view that an update is required, based on a on change in the model.

In the second approach to implementation, the controller is positioned at the center of the pattern and the models and views communicate through the controller channels. For example, after a view has notified the controller of a user action, the controller will update the property in the model based upon that action. From the other direction, the controller registers for the changes in the model and updates the view based on the notification triggered from the model. This approach is combined with the mediator where the controller plays the mediator role for model and view communications.

Composite Design Pattern. The composite hierarchy design pattern provides a flexible and extensible solution to the specification of hierarchical tree structures

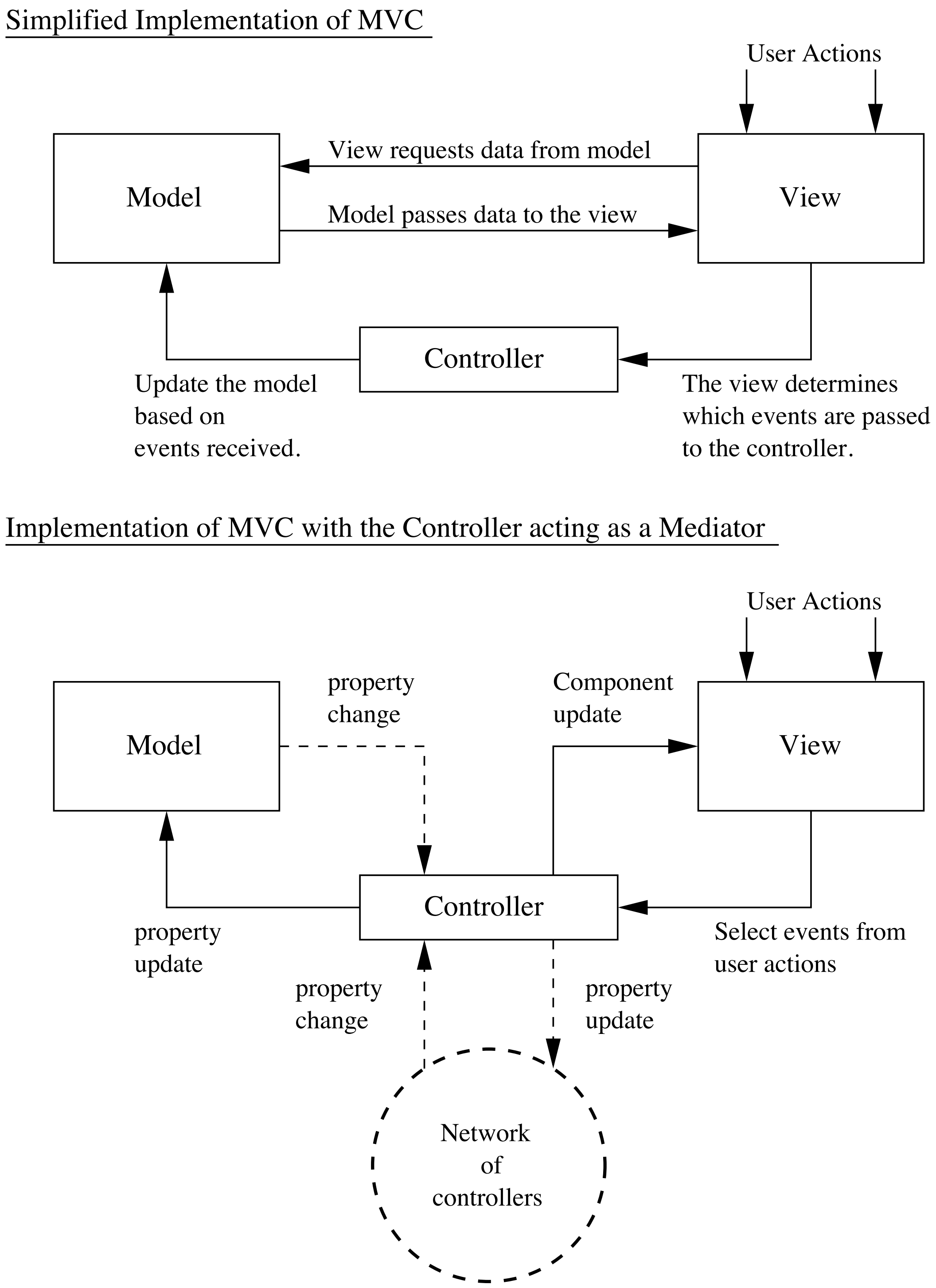

Simplified Implementation of MVC
User Actions
Model
View requests data from model
Model passes data to the view
View
Controller
Update the model based on events received.
The view determines which events are passed to the controller.
Implementation of MVC with the Controller acting as a Mediator
User Actions
Model
property change
Component update
View
Controller
property update
Select events from user actions
property change
property update
Network of controllers

Figure 2.5: Styles of model-view-controller implementation.

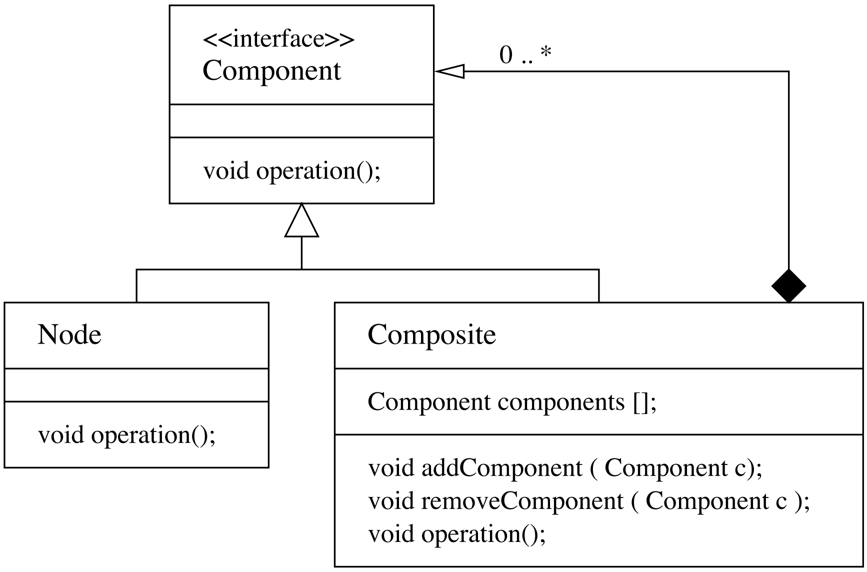


Figure 2.6: Composite class diagram.

(i.e., part-whole hierarchies) of arbitrary complexity. As illustrated in Figure 2.6, implementations of this pattern employ component, node, and composite classes:

1. **Component.** The component class is a common interface which defines methods that must be implemented by the objects and propagated down to all parts of the tree structure.

2. **Composite.** The Composite class serves as a container to store the components. It supports a dynamic set of Component references, and so it has methods to add and remove components to/from the container, as well as retrieve Component objects from its collection. All operational methods which are defined in the Component interface must also be implemented.

3. **Node.** The node (or leaf) classes represent terminal behavior of single objects (i.e., parts of the composite that will not have other components). They also implement the Component interface and provide an implementation for each

of the component's operational methods.

It is important to note that composite objects can be assembled recursively into a multi-layer tree structure. The structure grows until the lowest-level node (leaf) is reached.

Observer Design Pattern. The observer design pattern defines a one-to-many relationship between objects. The pattern defines the dependency between a "subject object" and an "observer object." An observer component registers itself to a subject of interest and will be notified when an event occurs. The registration process can be very flexible: observers can register with a multitude of subject (observable) components and be removed when an interest in the subject no longer exists.

Figure 2.7 shows the relationship among classes and interfaces in a typical implementation of the observer design pattern. Observable subjects are extensions to a high-level Observable class. The latter uses a collection of references to Observer interfaces. All that concrete observers need to do is implement the observer interface. Since concrete observers can register with multiple subjects, software architectures can be easily assembled into networks of loosely coupled entities.

To see how this observer design pattern works in practice, Figure 2.8 shows a problem setup where two intervals on a line segment are defined by the position of three nodes. As drawn, intervals 1 and 2 are each five units of length. Now suppose that node 2 is moved one interval to the right. Intervals 1 and 2 will now have lengths six and four respectively. We can model the node-interval dependency

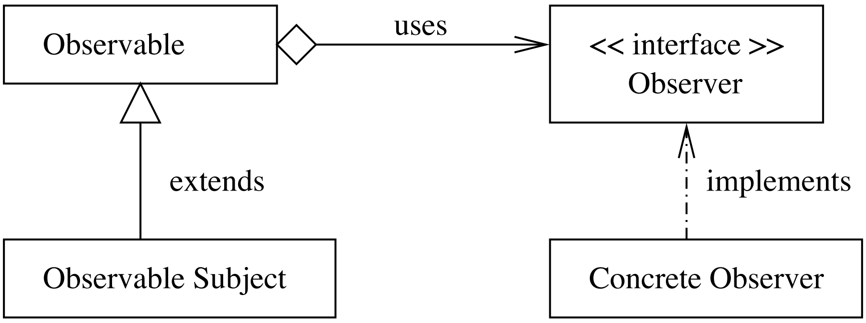


Figure 2.7: Relationship of classes and interfaces in the observer design pattern.

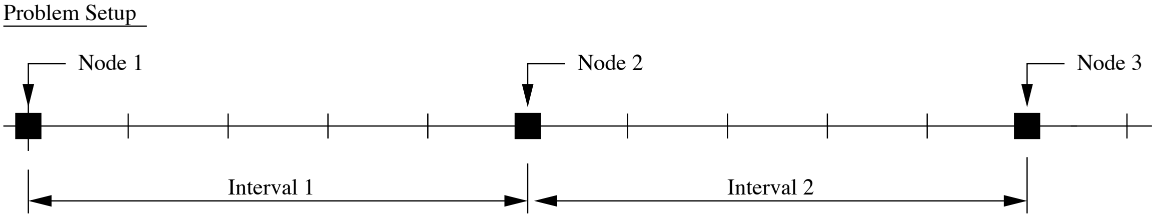


Figure 2.8: Problem setup: three nodes and two intervals.

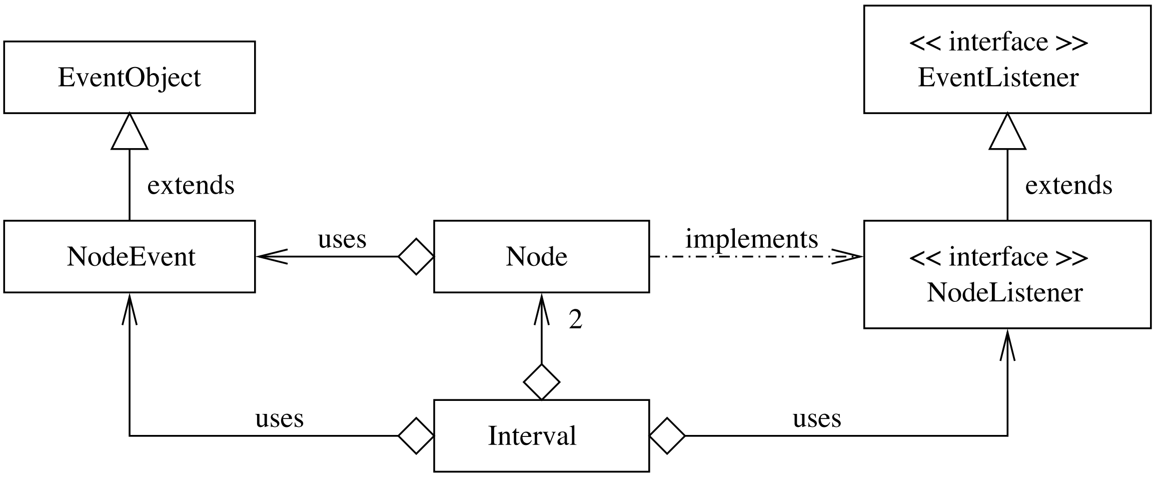


Figure 2.9: Relationship among classes for the node and interval dependency model.

relationships with the network of class relationships shown in Figure 2.9. Each interval is defined by the position of two nodes. Rather than embed references to the nodes inside the interval (certainly this would work), the nodes are observable subjects and the intervals are the observers. For this specific application, interval 1 will observe the positions of nodes 1 and 2. Interval 2 will observe the positions of nodes 2 and 3. The act of repositioning a node will generate a node event, which in turn, will trigger notifications to the registered observers.

Visitor Design Pattern. The visitor software design pattern allows for the separation of an algorithm (system functionality) from an object structure on which it operates. Use of this pattern makes sense when you have distinct and unrelated operations (e.g., print, display, modify) to perform across an ensemble or structure of objects.

Figure 2.10 shows the arrangement of classes in a typical implementation of the visitor design pattern. Visitors A and B each implement a visitor interface. The objects to be visited (i.e., Elements A and B) implement an accept() method having an argument of type interface visitor. It is important to note that visitors need not know the details of the organizational structure beforehand – they just visit collections of objects, perform their function, and then leave. A key benefit in using the visitor design pattern is that new operations (e.g., print, display, export) may be added to existing object structures without the need to actually modify those structures.

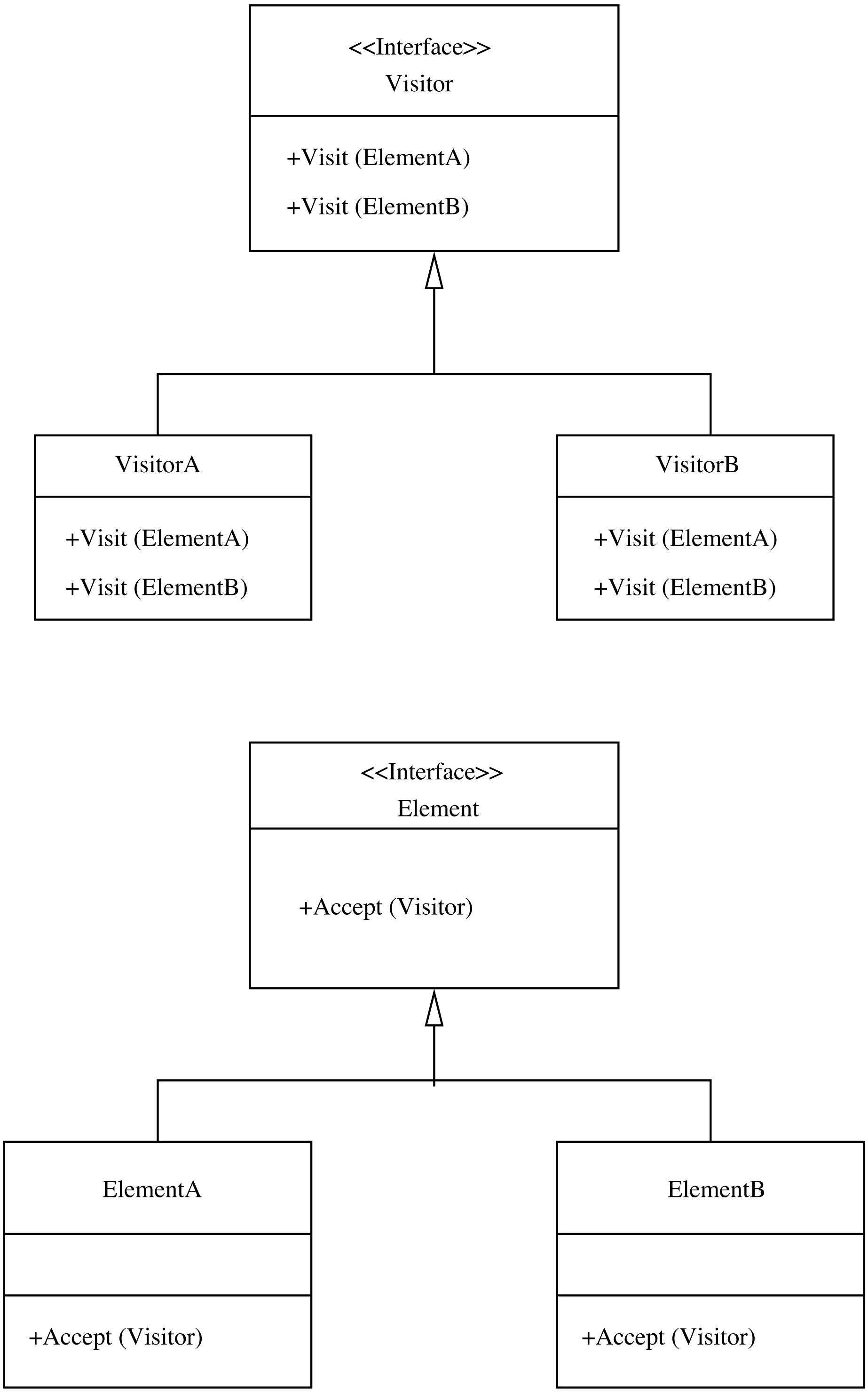


Figure 2.10: Schematic for visitor class diagram.

2.5 Composite Hierarchies of Features

An important aspect of building floorplans (and buildings in general) is that they contain a large number of different types of components (e.g., floors, rooms, walls, columns, doors, windows, etc). While it is certainly possible to model each of these component types explicitly, the long-term management of these components would be very tedious. To circumvent this problem, an alternative and more elegant approach is adopted. Instead of modeling the building floorplan as collections of specific types of components (e.g., walls, doors, windows), the general idea is that all components will be modeled as features. Features will be organized into composite hierarchies.

Figure 2.11 is a class diagram for the modeling of building floorplan as composite hierarchies of features. A feature is a generic interface to a component, e.g.,

```
public interface Feature extends Cloneable {
   public void    setName( String sName );
   public String  getName();
   public void    setX( double dX );
   public double  getX();
   public void    setY( double dY );
   public double  getY();
   public void    setColor( Color c );
   public Color   getColor();
   public void    setSelection( boolean b );
   public boolean getSelection();
   public void    accept( FeatureElementVisitor visitor );
   public void    search( AffineTransform at, int dx, int dy );
}
```

The feature interface extends clonable, which means that it supports the copying of features. It provides methods for naming and positioning of features, selection of

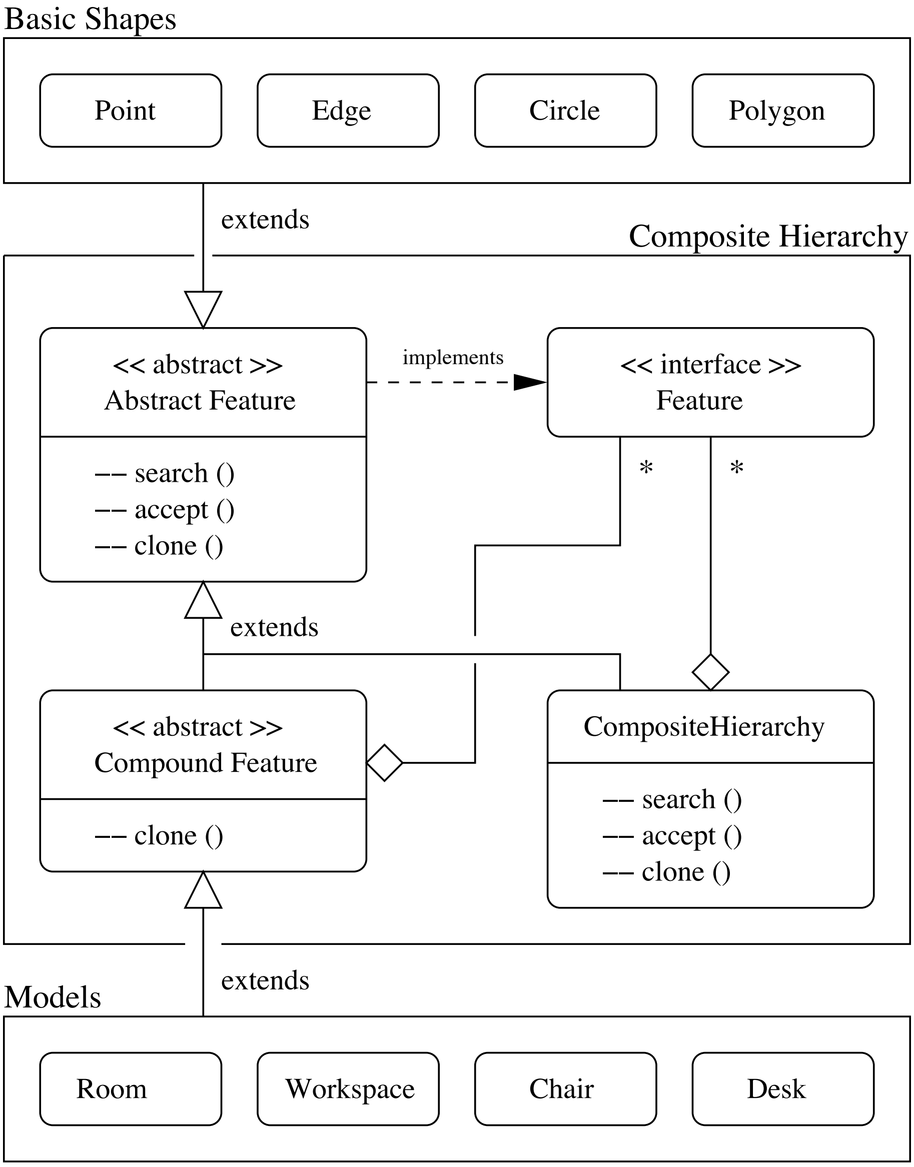


Figure 2.11: Composite hierarchy class diagram for the modeling of floorplans.

objects in the composite hiearchy, the acceptance of visitors (e.g., to print or display objects in the hierarchy) and search the hierarchy for an object having coordinates (x,y).

The class AbstractFeature implements the interface Feature and, as such, it is required to provide implementations for all of the methods declared in the interface specification. Parameters are provided for the features (x,y) coordinates, color, height, width, and rectangular bounding box. The search() method provides for systematic traversal of the composite hierarchy. The accept() method provides for traversal of the composite hierarchy by methods that implement the FeatureElementVisitor interface. The clone() method makes a deep copy of the abstract feature contents. Basic shapes, such as point, edge, circle and polygon are extensions of AbstractFeature.

The class AbstractCompoundFeature supports the representation of components that are an assembly of simple feature primitives (e.g., lines and filled circles). The extension relationship between the class AbstractCompoundFeature and class AbstractFeature is indicated by the triangle notation (△). Similarly, the graphical notation (◇ and $*$) indicates that a compound feature will contain zero or more features. Our current implementation stores these items as a hashmap, i.e.,

```
public HashMap<String,Feature> items = new HashMap<String,Feature>();
```

Simplified models of real-world components (e.g., chairs, desks, a very simple car) can be represented as abstract compound features.

The class `CompositeHierarchy` provides support for the specification of composite hierarchies of features. Each layer of the composite hierarchy is an array list of features, i.e.,

```
  private ArrayList<Feature> children = new ArrayList<Feature>();
```

The class parameters keep track of the current level number, as well as the global and local offsets in the x- and y- directions, and rotation. Traversal of the composite hierarchy corresponds to a recursive search. During a traversal of the composite hierarchy model, updates in the cooordinate offsets are managed by a family of affine transformation matrices. Methods are provided to add and remove a feature, and find a feature having a specific name. The methods `accept()` and `clone()` are redefined so that array list of features can be visited and copied, respectively.

Chapter 3

Approach 1: Scripting Floorplan Specifications

This chapter presents a script-based approach to the step-by-step assembly of floorplan models. This approach is the first of two approaches to the development of floorplan specifications presented in this thesis.

3.1 Step-by-Step Assembly Procedure

The step-by-step assembly procedures for floorplan examples are as follows:

1. Establish the base composite hierarchy workspace for the room model.
2. Specify fixed entities, such as column in the model.
3. Define positions of the centerlines along the x- and y- axes.
4. Generate junction points on the intersection of correlated centerline for the model.
5. Build up wall components by the demarcated relationship of junction points.
6. Create composite hierarchy workspaces for walls that contain doors and windows.
7. Add door and window components to the corresponding composite hierarchy workspaces.

8. Analyze the sequence of design refinement by using JGraphT.

9. Determines building components and other subsystems to form a room system.

10. Calculate the usable area by using Java Topology Suite.

The underlying dependency relationships are modeled with JGraphT, a graph package. Area calculations are handled by the Java Topology Suite.

3.2 Example 1. Scripting Specification for a Simple Room.

We begin with a step-by-step assembly of a model for a simple room, as illustrated in Figure 3.1. Despite its small size, it uses all of the component types and is assembled into a small composite hierarchy. We also demonstrate how the model reacts to the re-positioning of a single centerline.

3.2.1 Step-by-Step Assembly of the Floorplan Layout

Part 1. Definition of the composition hierarchy workspace, centerlines, columns, junction points, walls, doors and windows.

Step 1. Create a base composite hierarchy workspace. The output is as follows:

result scripts

```
Create Composite Hierarchy workspace:
       Location: (0.0, 0.0)
       Rotation: 0.0
```

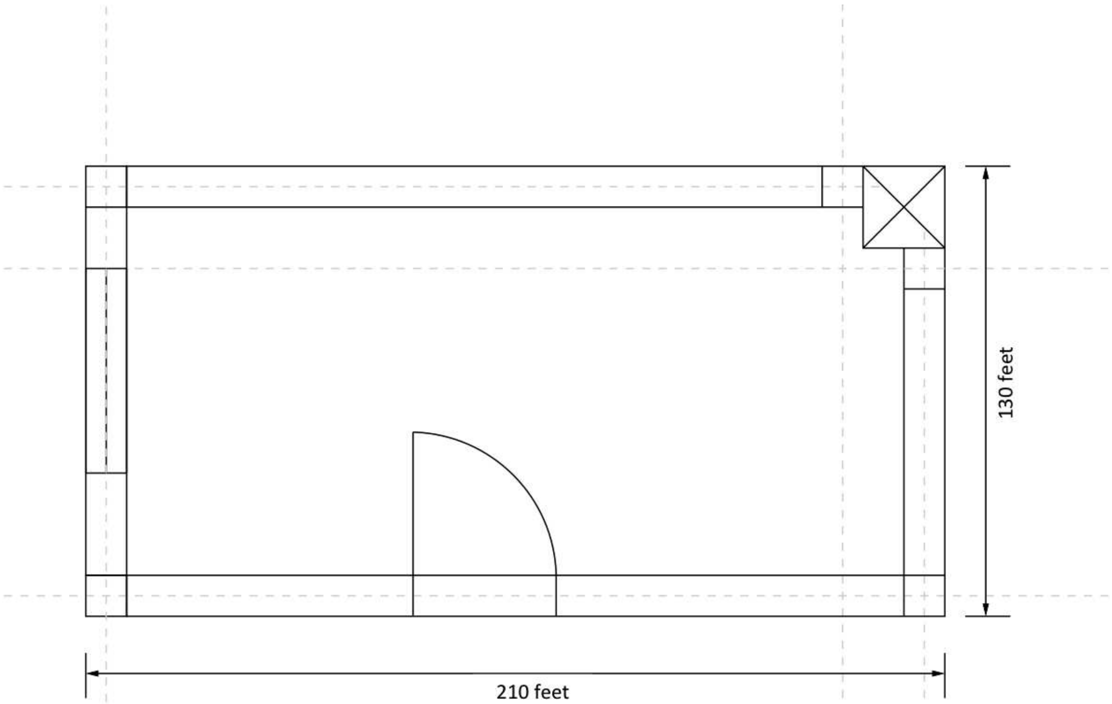


Figure 3.1: Floorplan and dimensions for a simple room.

Step 2. Add entities to the model that are fixed in place (i.e., they cannot be re-positioned). For our purposes, columns are fixed in place.

result scripts

```
Create a unmovable Column:
       Location: (185.0, 105.0)
       Width: 20.0  Height: 20.0
```

Step 3. Now that the fully-fixed components are in place, the next step is to define the layout of designable component pieces. For our example, establishing centerlines might be a great start to depict a room system.

result scripts

```
Create vertical Centerline  : x = 0.0
Create horizontal Centerline: y = 0.0
Create vertical Centerline  : x = 200.0
Create horizontal Centerline: y = 120.0
Create vertical Centerline  : x = 180.0
Create horizontal Centerline: y = 100.0
```

Notice that the centerline positions are consistent with the overall room dimensions shown in Figure 3.1.

Step 4. A grid of centerline positions provides a very high-level blueprint for the definition of junction points and positioning of rooms. Junction points were generated as follows:

result scripts

```
Create a Junction Point
       Location: (-5.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c1, Centerline c2
Create a Junction Point
       Location: (-5.0, 115.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c1, Centerline c4
Create a Junction Point
       Location: (195.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c3, Centerline c2
Create a Junction Point
       Location: (175.0, 115.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c5, Centerline c4
Create a Junction Point
       Location: (195.0, 95.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c3, Centerline c6
```

Step 5. As already explained in the component library section, wall components are generated between junction points. If the wall contains either a window or a door, then composite hierarchy workspace for the corresponding wall element will be created to form a subsystem inside the system. The following script illustrates this process:

result scripts

```
Create a wall
       Location: (-5.0, 5.0)
       Width: 10.0  Height: 110.0
       Dependency: jpt1, jpt2
Create a wall
       Location: (5.0, -5.0)
       Width: 190.0  Height: 10.0
       Dependency: jpt1, jpt3
Create a wall
       Location: (5.0, 115.0)
       Width: 170.0  Height: 10.0
       Dependency: jpt2, jpt4
Create a wall
       Location: (195.0, 5.0)
       Width: 10.0  Height: 90.0
       Dependency: jpt3, jpt5
```

Step 6 and 7. Composite hierarchy models are serve as a container inside the systems. As the definition in this research, doors, windows or other components have to set up inside the composite hierarchy of that wall if it is on the wall or inside the wall. In this case, there are two components, door and window, in this example, and those components are contained inside the corresponding composite hierarchy; moreover, the coordinates of the components are set from the base composite hierarchy to the wall composite hierarchy. The scripts for this example are shown below.

result scripts

```
Create Composite Hierarchy workspace:
       Location: (5.0, -5.0)
       Rotation: 0.0
========================================================
Create a Door:
       Location: (60.0, 0.0)
       Width: 40.0  Height: 10.0
========================================================
Create Composite Hierarchy workspace:
       Location: (5.0, 5.0)
       Rotation: 1.5707963267948966
========================================================
Create a Window:
       Location: (20.0, 0.0)
       Width: 50.0  Height: 10.0
```

Part 2. Design refinement and dependency analysis.

Step 8. Dependency analysis provides design engineers with an understanding of the cause-and-effect relationships between changes to the high-level parameters settings (e.g., the position of a center line), and its affect on lower level entities, such as the position of junction points and the dimensions of a wall component. The script belows shows the sequence of design refinement for our simple one room model.

result scripts

```
Sequence of Design Refinement:
=====================
Base Composite Hierarchy
Column
Centerlines:        c1,    c2,    c3,    c4,   c5,  c6
Junction Points:  jpt1,  jpt3,  jpt2,  jpt4, jpt5
Wall Elements:   wall2, wall1, wall3, wall4

Composite Hierarchy wall2
   Leftside window
Composite Hierarchy wall1
   Front door
```

Step 9 and 10. With the room model in place, we can use the Java Topology Suite to calculate the room model area. Notice that each object inside the room model has its own area, and that all objects are circle the usable space of the room. As indicated in the script indicates below, the union of the whole surrounding components area and the abstract room area difference with the whole area will be the usable spaces inside the room. The abstract room area was defined by the surrounded room objects, and it will generate a room polygon by connecting all the centers of junction points that surrounded the room.

result scripts

```
Room components = POLYGON ( ( -5  -5,  -5   5,   -5 115,
                              -5 125,   5 125,  175 125,
                             185 125, 205 125,  205 105,
                             205  95,   205 5,   205 -5,
                             195  -5,    5 -5,    -5 -5 ),
                          (  195  95, 195 105,  185 105,
                             185 115, 175 115,    5 115,
                               5   5,   195 5,  195  95 ) )

Columns, Junction Points & Walls Area = 65.00 square foot
===============================================================

Whole Geometry = POLYGON ( (  -5  -5,  -5   5,  -5 115,
                              -5 125,   5 125, 175 125,
                             185 125, 205 125, 205 105,
                              205 95,   205 5,  205 -5,
                              195 -5,    5 -5,   -5 -5 ) )

Total Covered Area = 273.00 square foot
===============================================================

Usable Area Geometry = POLYGON ( (195 95,   195 5,     5 5,
                                    5 115, 175 115, 185 115,
                                  185 105, 195 105,  195 95 ) )

Usable Area = 208.00 square foot
===============================================================
```

In these calculations, usable area is defined as the area surrounding components, such as column, walls, and junction points. Figure 3.2 shows the corresponding dependency for each component. The arrow is pointing to the area or component

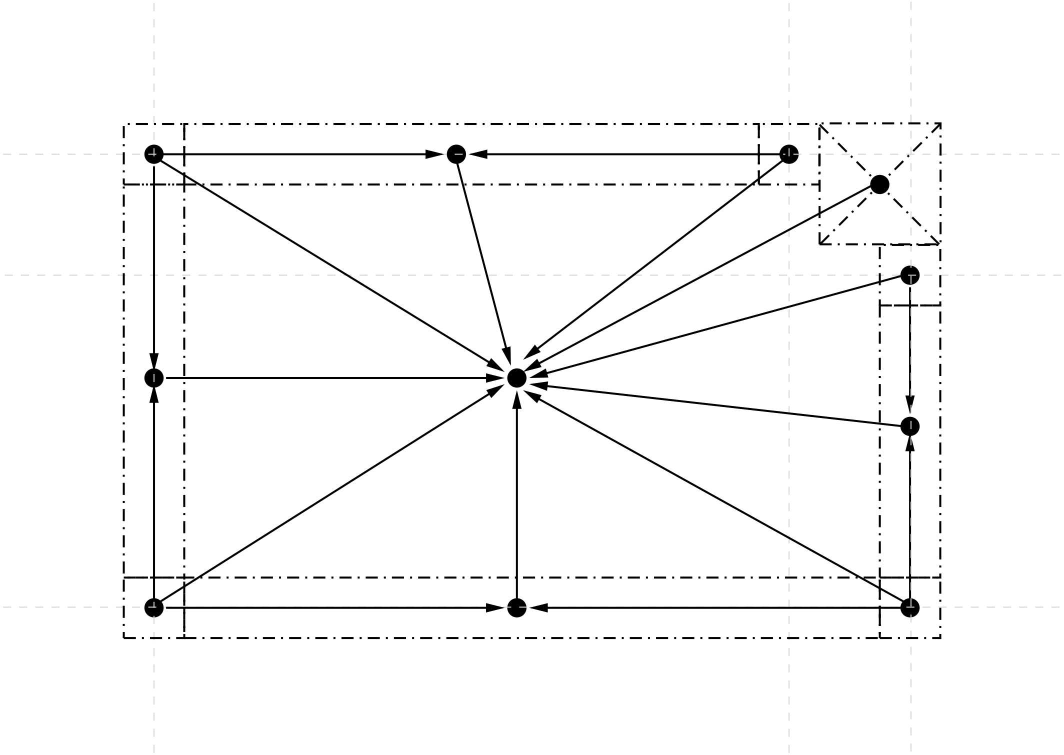


Figure 3.2: Graph of dependency relationships in the room. Only the parent dependencies are shown.

from the parents. Therefore, all the dependency of components will convergence to the usable area in the center.

3.2.2 System Redesign

The system is redesigned by adjusting the positions of centerlines c3 and c4 to shrink the room area. When centerline c3 is moved to left, both junction point 3 and junction point 5 that are related to centerline c3 will move as well. Further, junction point 2 and junction point 4 will alter their location due to the shift of centerline c3. The wall objects that depend on these relative junction points will be adjusted. The script below shows the chain of events that is triggered by the system redesign:

result scripts

```
Create vertical Centerline  : x = 0.0
Create horizontal Centerline: y = 0.0
Create vertical Centerline  : x = 190.0
Create horizontal Centerline: y = 110.0
Create vertical Centerline  : x = 180.0
Create horizontal Centerline: y = 100.0
========================================================
Create a Junction Point
       Location: (-5.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c1, Centerline c2
Create a Junction Point
       Location: (-5.0, 105.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c1, Centerline c4
Create a Junction Point
       Location: (185.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c3, Centerline c2
Create a Junction Point
       Location: (175.0, 105.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c5, Centerline c4
Create a Junction Point
       Location: (185.0, 95.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c3, Centerline c6
========================================================
Create a wall
       Location: (-5.0, 5.0)
       Width: 10.0  Height: 100.0
       Dependency: jpt1, jpt2
Create a wall
       Location: (5.0, -5.0)
       Width: 180.0  Height: 10.0
       Dependency: jpt1, jpt3
Create a wall
       Location: (5.0, 105.0)
       Width: 170.0  Height: 10.0
       Dependency: jpt2, jpt4
Create a wall
       Location: (185.0, 5.0)
       Width: 10.0  Height: 90.0
       Dependency: jpt3, jpt5
```

After the new surrounding objects model have been built up, the new center area of the room can be calculated by the Java Topology Suite. Notice that the number of objects and the dependency relationships among the objects remained unchanged by the redesign. As such, the sequence of design refinement for this model remains the same as for the initial model. The essential details of the area calculations (and program output) are as follows:

result scripts

```
Room components = POLYGON ( (  -5 -5,    -5 5,  -5 105,
                              -5 115,   5 115, 175 115,
                             185 115, 185 125, 205 125,
                             205 105, 195 105,  195 95,
                               195 5,  195 -5,  185 -5,
                                5 -5,  -5 -5), (185 95,
                             185 105, 175 105,   5 105,
                                 5 5,   185 5,  185 95) )

Columns, Junction Points & Walls Area = 63.00 square foot
=================================================================

Whole Geometry = POLYGON ( (  -5  -5,  -5   5,  -5 105,  -5 115,
                               5 115, 175 115, 185 115, 185 125,
                             205 125, 205 105, 195 105, 195  95,
                             195   5, 195  -5, 185  -5,   5  -5,
                              -5  -5 ) )

Total Covered Area = 243.00 square foot
=================================================================

Usable Area Geometry = POLYGON ( (185 95, 185   5,   5  5, 5 105,
                                 175 105, 185 105, 185 95) )

Usable Area = 180.00 square foot
=================================================================
```

Figures 3.3 and 3.4 are a side-by-side comparison of the original and redesigned room models.

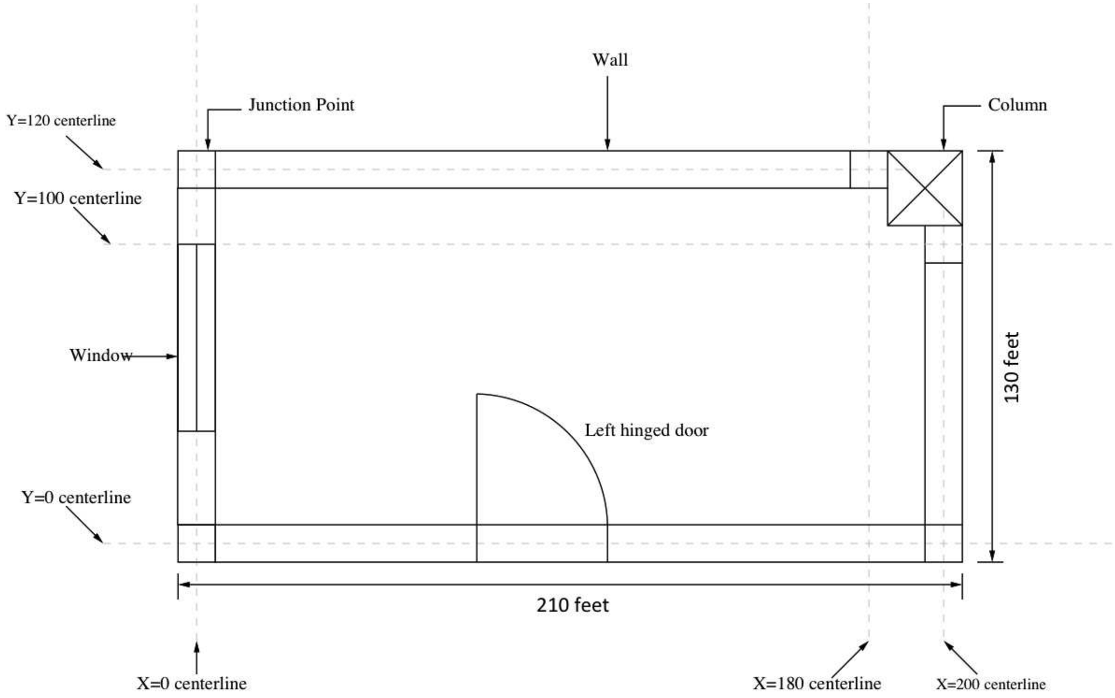


Figure 3.3: The original design of the simple room example.

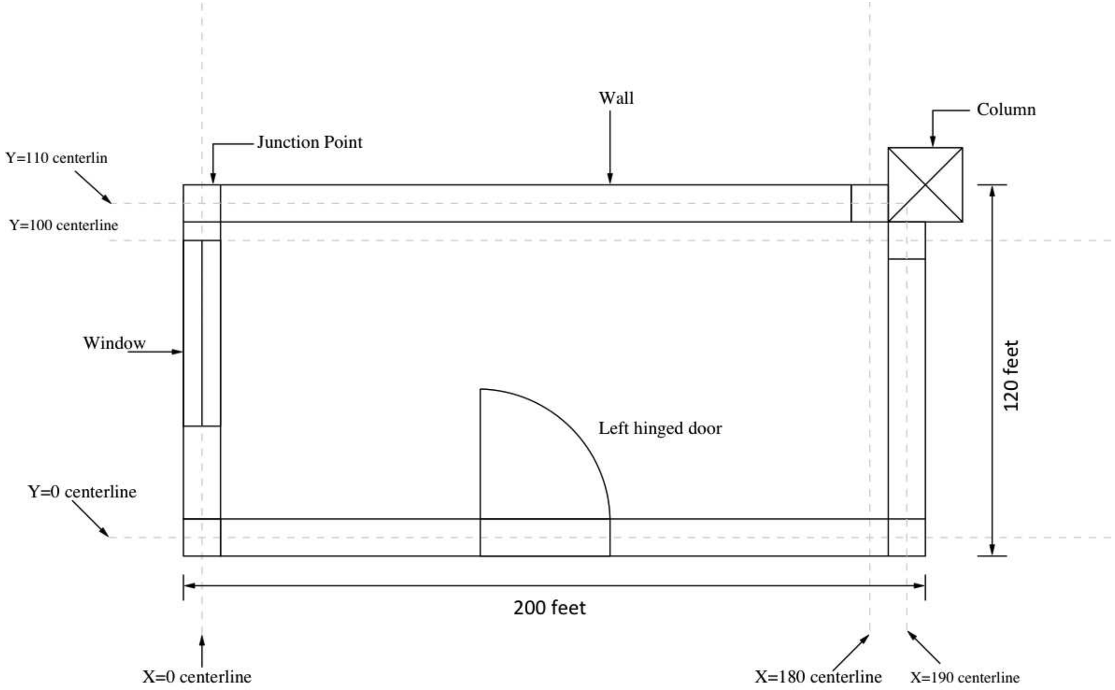


Figure 3.4: The redesign of the simple room example.

3.3 Example 2. Scripting Specification for a Simple House

The simple room model developed in the previous section is a subsystem system that could be placed in any house or building. We now demonstrate the floorplan modeling capabilities by repeating the step-by-step assembly procedure for a floorplan containing two apartment units. Figure 3.5 shows details of the architectural floorplan. Figure 3.6 shows a graph of room adjacency relationships.

3.3.1 Step-by-Step Assembly of the House Floorplan

Step 1. We begin by defining a composite hierarchy structure for the base workspace of the whole building system.

result scripts

```
Create Composite Hierarchy workspace:
       Location: (0.0, 0.0)
       Rotation: 0.0
```

Step 2. Because the simple house example is not a room subsystem inside a building, design regulations indicated that there will be no fixed entities inside the building system model.

Step 3. Twenty seven centerlines are strategically positioned along the x- and y-axes to create the floorplan design layout. The intersection of these centerlines can act as the parents of junction points, which, in turn, may connect to wall elements.

result scripts

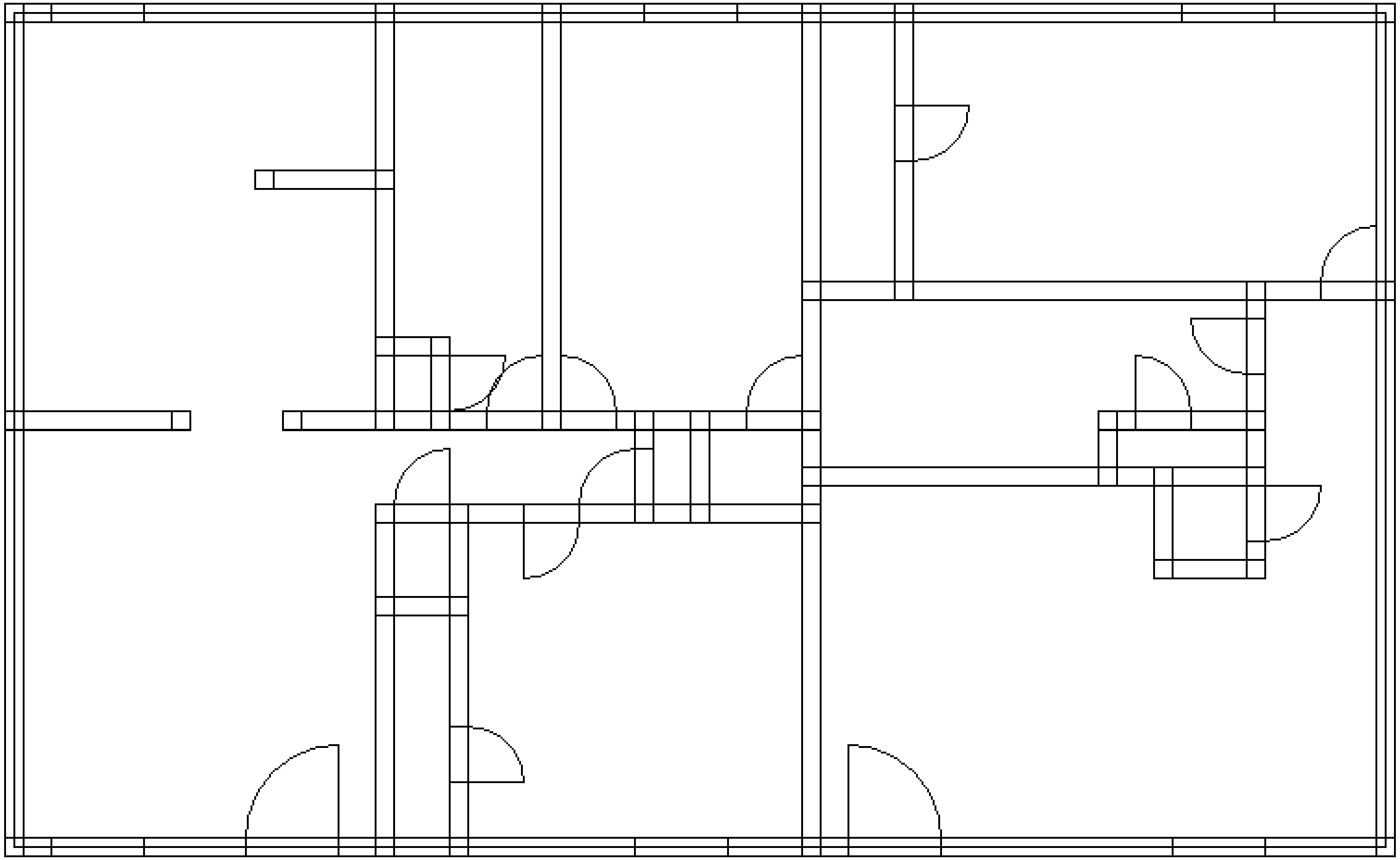


Figure 3.5: Original design of the simple house.

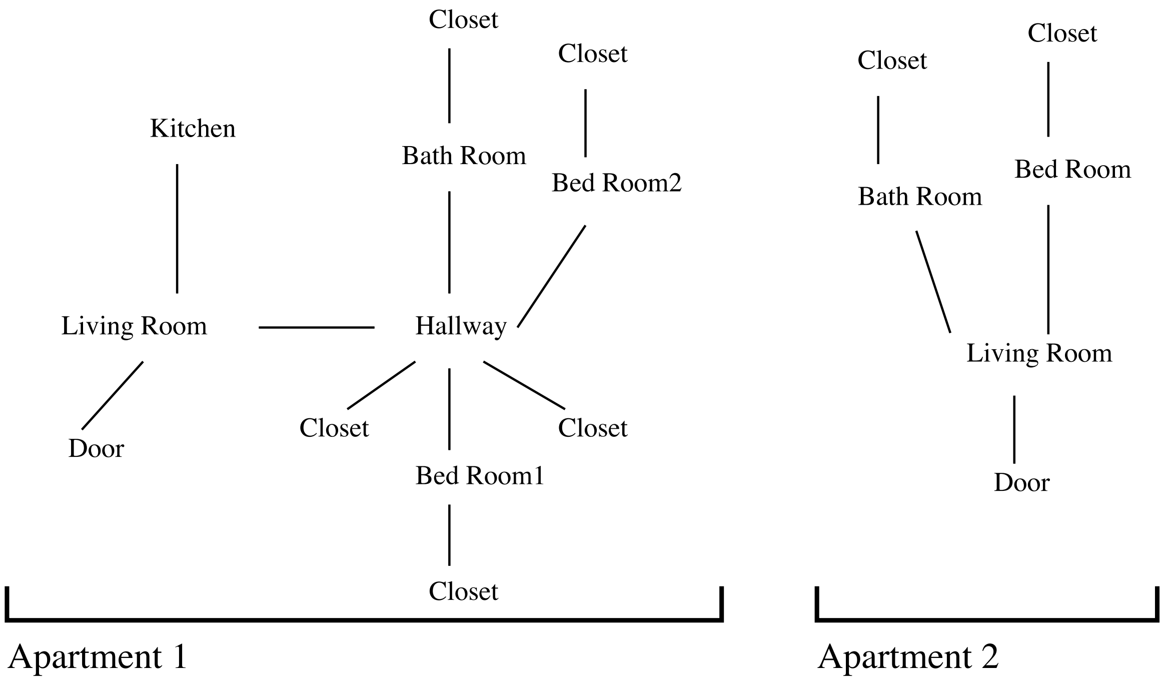


Figure 3.6: Room adjacency relationships in the simple house.

```
Create vertical Centerline  : x = 0.0
Create vertical Centerline  : x = 90.0
Create vertical Centerline  : x = 135.0
Create vertical Centerline  : x = 150.0
Create vertical Centerline  : x = 200.0
Create vertical Centerline  : x = 230.0
Create vertical Centerline  : x = 240.0
Create vertical Centerline  : x = 290.0
Create vertical Centerline  : x = 340.0
Create vertical Centerline  : x = 370.0
Create vertical Centerline  : x = 430.0
Create vertical Centerline  : x = 480.0
Create vertical Centerline  : x = 590.0
Create vertical Centerline  : x = 620.0
Create vertical Centerline  : x = 670.0
Create vertical Centerline  : x = 740.0
Create horizontal Centerline: y = 0.0
Create horizontal Centerline: y = 130.0
Create horizontal Centerline: y = 150.0
Create horizontal Centerline: y = 180.0
Create horizontal Centerline: y = 200.0
Create horizontal Centerline: y = 230.0
Create horizontal Centerline: y = 230.0
Create horizontal Centerline: y = 270.0
Create horizontal Centerline: y = 300.0
Create horizontal Centerline: y = 360.0
Create horizontal Centerline: y = 450.0
```

Step 4. Next, we identify centerline intersection points that define profiles for the exterior and interior walls. From a modeling standpoint, this step is complicated by many-to-many relationships between the building elements. junction points are the parents of the wall element between. For this particular example, the walls that divided the entire floor plan into two apartment units were relied on the junction points which has the same vertical parent centerline (x = 435), but different horizontal parent centerlines.

The script below is a summary of the 43 junction points:

result scripts

```
Create a Junction Point
       Location: (-5.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c1, Centerline c17
Create a Junction Point
       Location: (195.0, -5.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c5, Centerline c17

... details of output removed ...

Create a Junction Point
       Location: (475.0, 445.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c12, Centerline c26
Create a Junction Point
       Location: (735.0, 445.0)
       Width: 10.0  Height: 10.0
       Dependency: Centerline c16, Centerline c26
```

Step 5. Closets are placed inside rooms and bathrooms. Since rooms are no longer simple rectangles, this complicates the layout of wall elements. The simple house has 55 wall elements whose locations are dependent on the adjoining junction points.

The abbreviated details are as follows:

result scripts

```
Create a wall
       Location: (5.0, -5.0)
       Width: 190.0  Height: 10.0
       Dependency: jpt1, jpt2
Create a wall
       Location: (205.0, -5.0)
       Width: 30.0  Height: 10.0
       Dependency: jpt2, jpt3

... details of wall elements removed ...

Create a wall
       Location: (665.0, 235.0)
       Width: 10.0  Height: 50.0
       Dependency: jpt29, jpt34
Create a wall
       Location: (735.0, 5.0)
       Width: 10.0  Height: 280.0
       Dependency: jpt5, jpt35
Create a wall
       Location: (735.0, 295.0)
       Width: 10.0  Height: 150.0
       Dependency: jpt35, jpt43
```

Step 6 and 7. Door, window and portal elements are added to the floorplan. Notice, in particular, the use of portal elements: one between living room and kitchen, and a second placed between the living room and hallway.

result scripts

```
Create Composite Hierarchy workspace:
       Location: (5.0, -5.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (245.0, -5.0)
       Rotation: 0.0

... details of output removed ...

Create Composite Hierarchy workspace:
       Location: (665.0, 195.0)
       Rotation: -1.5707963267948966
========================================================
Create Composite Hierarchy workspace:
       Location: (675.0, 235.0)
       Rotation: 1.5707963267948966
========================================================
Create a Door:
       Location: (120.0, 0.0)
       Width: 50.0  Height: 10.0
Create a Door:
       Location: (15.0, 0.0)
       Width: 50.0  Height: 10.0

... details of output removed ...

Create a Door:
       Location: (45.0, 0.0)
       Width: 30.0  Height: 10.0
Create a Door:
       Location: (0.0, 0.0)
       Width: 30.0  Height: 10.0
Create a Window:
       Location: (15.0, 0.0)
       Width: 50.0  Height: 10.0
Create a Window:
       Location: (90.0, 0.0)
       Width: 50.0  Height: 10.0

... details of output removed ...

Create a Window:
       Location: (45.0, 0.0)
       Width: 50.0  Height: 10.0
Create a Window:
       Location: (145.0, 0.0)
       Width: 50.0  Height: 10.0
Create a portal
       Location: (95.0, 225.0)
       Width: 50.0  Height: 10.0
       Dependency: jpt20, jpt21
Create a portal
       Location: (195.0, 185.0)
       Width: 10.0  Height: 40.0
       Dependency: jpt10, jpt22
```

3.3.2 Systems Analysis

Step 8. The following (abbreviated) script of code shows the sequence of design refinement employed in systems analysis. The latter is supported by JGraphT and the Java Topology Suite.

result scripts

```
Sequence of Design Refinement:
=====================
Base Composite Hierarchy
Centerline c1
Centerline c2

... details of output removed ...

Centerline c25
Centerline c26
jpt1
jpt2
```

```
... details of output removed ...

jpt42
jpt43
wall1
wall2

... details of output removed ...

wall31
wall57
wall1 CH
wall3 CH

... details of output removed ...

wall50 CH
wall31 CH
Front door A
Front window A
Bedroom1 window
Front door B
Front window B
Closet door 5
Closet door 1
Bedroom door 1
Closet door 8
Bathroom door 1
Bedroom door 2
Closet door 2
Closet door 6
Closet door 3
Closet door 4
Bathroom door 2
Bedroom door 3
Closet door 7
Leftside window
```

Step 9 and 10. The building floor plan consists of two apartment unit subsystems. The first apartment has a living room, kitchen, hallway, bathroom, bedroom 1, and bedroom 2, with adjacency relationships as shown in the previous figure. Each of the room subsystems can be defined by identifying the components inside each room. Then, the Java Topology Suite can calculate the area of each room. Similarly, the second apartment can also divided into living room, bathroom, and bedroom.

The following script of code summarizes these room subsystems:

result scripts

```
Apartment 1 Living Room = POLYGON ((-5 -5, -5 5, -5 225, -5 235, 5 235, 85 235,
 95 235, 145 235, 155 235, 195 235, 205 235, 205 225, 205 185, 205 175, 205 135,
 205 125, 205 5, 205 -5, 195 -5, 5 -5, -5 -5), (195 5, 195 125, 195 135, 195 175,
 195 185, 195 225, 155 225, 145 225, 95 225, 85 225, 5 225, 5 5, 195 5))
Columns, Junction Points & Walls Area = 86.00 square foot

Whole Geometry = POLYGON ((-5 -5, -5 5, -5 225, -5 235, 5 235, 85 235, 95 235,
 145 235, 155 235, 195 235, 205 235, 205 225, 205 185, 205 175, 205 135, 205 125,
 205 5, 205 -5, 195 -5, 5 -5, -5 -5))
Apartment 1 Living Room Covered Area = 504.00 square foot

Usable Area Geometry = POLYGON ((195 5, 5 5, 5 225, 85 225, 95 225, 145 225,
 155 225, 195 225, 195 185, 195 175, 195 135, 195 125, 195 5))
Apartment 1 Living Room Usable Area = 418.00 square foot

========================================================
Apartment 1 Kitchen = POLYGON ((-5 225, -5 235, -5 445, -5 455, 5 455, 195 455,
 205 455, 205 445, 205 365, 205 355, 205 275, 205 265, 205 235, 205 225, 195 225,
 155 225, 145 225, 95 225, 85 225, 5 225, -5 225), (85 235, 95 235, 145 235,
 155 235, 195 235, 195 265, 195 275, 195 355, 140 355, 130 355, 130 365, 140 365,
 195 365, 195 445, 5 445, 5 235, 85 235))
Columns, Junction Points & Walls Area = 90.50 square foot

Whole Geometry = POLYGON ((-5 225, -5 235, -5 445, -5 455, 5 455, 195 455, 205 455,
 205 445, 205 365, 205 355, 205 275, 205 265, 205 235, 205 225, 195 225, 155 225,
 145 225, 95 225, 85 225, 5 225, -5 225))
Apartment 1 Kitchen Covered Area = 483.00 square foot

Usable Area Geometry = POLYGON ((85 235, 5 235, 5 445, 195 445, 195 365, 140 365,
 130 365, 130 355, 140 355, 195 355, 195 275, 195 265, 195 235, 155 235, 145 235,
 95 235, 85 235))
Apartment 1 Kitchen Usable Area = 392.50 square foot

========================================================
Apartment 1 Hallway = POLYGON ((195 125, 195 135, 195 175, 195 185, 195 225,
 195 235, 205 235, 225 235, 235 235, 285 235, 295 235, 335 235, 345 235, 365 235,
 375 235, 375 225, 375 185, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135,
 245 125, 235 125, 205 125, 195 125), (235 135, 235 175, 205 175, 205 135,
 235 135), (235 185, 245 185, 335 185, 335 225, 295 225, 285 225, 235 225, 225 225,
 205 225, 205 185, 235 185), (365 185, 365 225, 345 225, 345 185, 365 185))
Columns, Junction Points & Walls Area = 61.00 square foot

Whole Geometry = POLYGON ((195 125, 195 135, 195 175, 195 185, 195 225, 195 235,
 205 235, 225 235, 235 235, 285 235, 295 235, 335 235, 345 235, 365 235, 375 235,
 375 225, 375 185, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135, 245 125,
 235 125, 205 125, 195 125))
```

```
Apartment 1 Hallway Covered Area = 133.00 square foot

Usable Area Geometry = MULTIPOLYGON (((235 135, 205 135, 205 175, 235 175,
 235 135)), ((235 185, 205 185, 205 225, 225 225, 235 225, 285 225, 295 225,
 335 225, 335 185, 245 185, 235 185)), ((365 185, 345 185, 345 225, 365 225,
 365 185)))
Apartment 1 Hallway Usable Area = 72.00 square foot

========================================================
Apartment 1 Bath = POLYGON ((195 225, 195 235, 195 265, 195 275, 195 355, 195 365,
 195 445, 195 455, 205 455, 285 455, 295 455, 295 445, 295 235, 295 225, 285 225,
 235 225, 225 225, 205 225, 195 225), (235 235, 285 235, 285 445, 205 445, 205 365,
 205 355, 205 275, 225 275, 235 275, 235 265, 235 235), (225 265, 205 265, 205 235,
 225 235, 225 265))
Columns, Junction Points & Walls Area = 68.00 square foot

Whole Geometry = POLYGON ((195 225, 195 235, 195 265, 195 275, 195 355, 195 365,
 195 445, 195 455, 205 455, 285 455, 295 455, 295 445, 295 235, 295 225, 285 225,
 235 225, 225 225, 205 225, 195 225))
Apartment 1 Bath Covered Area = 230.00 square foot

Usable Area Geometry = MULTIPOLYGON (((235 235, 235 265, 235 275, 225 275, 205 275,
 205 355, 205 365, 205 445, 285 445, 285 235, 235 235)), ((225 265, 225 235,
 205 235, 205 265, 225 265)))
Apartment 1 Bath Usable Area = 162.00 square foot

========================================================
Apartment 1 Bedroom 1 = POLYGON ((285 225, 285 235, 285 445, 285 455, 295 455,
 425 455, 435 455, 435 445, 435 295, 435 285, 435 235, 435 225, 435 205, 435 195,
 435 185, 435 175, 425 175, 375 175, 365 175, 365 185, 365 225, 345 225, 335 225,
 295 225, 285 225), (335 235, 345 235, 365 235, 375 235, 425 235, 425 285, 425 295,
 425 445, 295 445, 295 235, 335 235), (375 225, 375 185, 425 185, 425 195, 425 205,
 425 225, 375 225))
Columns, Junction Points & Walls Area = 87.00 square foot

Whole Geometry = POLYGON ((285 225, 285 235, 285 445, 285 455, 295 455, 425 455,
 435 455, 435 445, 435 295, 435 285, 435 235, 435 225, 435 205, 435 195, 435 185,
 435 175, 425 175, 375 175, 365 175, 365 185, 365 225, 345 225, 335 225, 295 225,
 285 225))
Apartment 1 Bedroom 1 Covered Area = 380.00 square foot

Usable Area Geometry = MULTIPOLYGON (((335 235, 295 235, 295 445, 425 445, 425 295,
 425 285, 425 235, 375 235, 365 235, 345 235, 335 235)), ((375 225, 425 225,
 425 205, 425 195, 425 185, 375 185, 375 225)))
Apartment 1 Bedroom 1 Usable Area = 293.00 square foot

========================================================
Apartment 1 Bedroom 2 = POLYGON ((195 -5, 195 5, 195 125, 195 135, 205 135,
 235 135, 235 175, 235 185, 245 185, 335 185, 345 185, 365 185, 375 185, 425 185,
 435 185, 435 175, 435 5, 435 -5, 425 -5, 245 -5, 235 -5, 205 -5, 195 -5), (245 5,
 425 5, 425 175, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135, 245 125,
 245 5), (235 125, 205 125, 205 5, 235 5, 235 125))
Columns, Junction Points & Walls Area = 94.00 square foot
```

```
Whole Geometry = POLYGON ((195 -5, 195 5, 195 125, 195 135, 205 135, 235 135,
 235 175, 235 185, 245 185, 335 185, 345 185, 365 185, 375 185, 425 185, 435 185,
 435 175, 435 5, 435 -5, 425 -5, 245 -5, 235 -5, 205 -5, 195 -5))
Apartment 1 Bedroom 2 Covered Area = 436.00 square foot

Usable Area Geometry = MULTIPOLYGON (((245 5, 245 125, 245 135, 245 175, 335 175,
 345 175, 365 175, 375 175, 425 175, 425 5, 245 5)), ((235 125, 235 5, 205 5,
 205 125, 235 125)))
Apartment 1 Bedroom 2 Usable Area = 342.00 square foot

========================================================
Apartment 2 Living Room = POLYGON ((425 -5, 425 5, 425 175, 425 185, 425 195,
 425 205, 435 205, 585 205, 595 205, 615 205, 625 205, 665 205, 665 225, 665 235,
 665 295, 665 305, 675 305, 735 305, 745 305, 745 295, 745 5, 745 -5, 735 -5,
 435 -5, 425 -5), (735 5, 735 295, 675 295, 675 235, 675 225, 675 205, 675 195,
 675 155, 675 145, 665 145, 625 145, 615 145, 615 155, 615 195, 595 195, 585 195,
 435 195, 435 185, 435 175, 435 5, 735 5), (665 155, 665 195, 625 195, 625 155,
 665 155))
Columns, Junction Points & Walls Area = 136.00 square foot

Whole Geometry = POLYGON ((425 -5, 425 5, 425 175, 425 185, 425 195, 425 205,
 435 205, 585 205, 595 205, 615 205, 625 205, 665 205, 665 225, 665 235, 665 295,
 665 305, 675 305, 735 305, 745 305, 745 295, 745 5, 745 -5, 735 -5, 435 -5,
 425 -5))
Apartment 2 Living Room Covered Area = 752.00 square foot

Usable Area Geometry = MULTIPOLYGON (((735 5, 435 5, 435 175, 435 185, 435 195,
 585 195, 595 195, 615 195, 615 155, 615 145, 625 145, 665 145, 675 145, 675 155,
 675 195, 675 205, 675 225, 675 235, 675 295, 735 295, 735 5)), ((665 155,
 625 155, 625 195, 665 195, 665 155)))
Apartment 2 Living Room Usable Area = 616.00 square foot

========================================================
Apartment 2 Bath Room = POLYGON ((425 195, 425 205, 425 225, 425 235, 425 295,
 425 305, 435 305, 475 305, 485 305, 665 305, 675 305, 675 295, 675 235, 675 225,
 675 205, 675 195, 665 195, 625 195, 615 195, 595 195, 585 195, 435 195, 425 195),
 (595 205, 615 205, 625 205, 665 205, 665 225, 595 225, 595 205), (595 235,
 665 235, 665 295, 485 295, 475 295, 435 295, 435 235, 435 225, 435 205, 585 205,
 585 225, 585 235, 595 235))
Columns, Junction Points & Walls Area = 78.00 square foot

Whole Geometry = POLYGON ((425 195, 425 205, 425 225, 425 235, 425 295, 425 305,
 435 305, 475 305, 485 305, 665 305, 675 305, 675 295, 675 235, 675 225, 675 205,
 675 195, 665 195, 625 195, 615 195, 595 195, 585 195, 435 195, 425 195))
Apartment 2 Bath Room Covered Area = 275.00 square foot

Usable Area Geometry = MULTIPOLYGON (((595 205, 595 225, 665 225, 665 205, 625 205,
 615 205, 595 205)), ((595 235, 585 235, 585 225, 585 205, 435 205, 435 225,
 435 235, 435 295, 475 295, 485 295, 665 295, 665 235, 595 235)))
Apartment 2 Bath Room Usable Area = 197.00 square foot

========================================================
Apartment 2 Bedroom = POLYGON ((425 295, 425 305, 425 445, 425 455, 435 455,
 475 455, 485 455, 735 455, 745 455, 745 445, 745 305, 745 295, 735 295, 675 295,
 665 295, 485 295, 475 295, 435 295, 425 295), (485 305, 665 305, 675 305, 735 305,
 735 445, 485 445, 485 305), (475 445, 435 445, 435 305, 475 305, 475 445))
Columns, Junction Points & Walls Area = 106.00 square foot

Whole Geometry = POLYGON ((425 295, 425 305, 425 445, 425 455, 435 455, 475 455,
 485 455, 735 455, 745 455, 745 445, 745 305, 745 295, 735 295, 675 295, 665 295,
 485 295, 475 295, 435 295, 425 295))
Apartment 2 Bedroom Covered Area = 512.00 square foot

Usable Area Geometry = MULTIPOLYGON (((485 305, 485 445, 735 445, 735 305, 675 305,
 665 305, 485 305)), ((475 445, 475 305, 435 305, 435 445, 475 445)))
Apartment 2 Bedroom Usable Area = 406.00 square foot
```

After the usable area for each room has been calculated, the whole building area can be calculated as a whole, which can further be the reference of the total room area. The union of all building objects area calculation can be present by Java Topology Suite as well. Moreover, the portal which is between the kitchen and the living room and the one between the hallway and the living room should also be added when doing the whole area calculation.

The script below demonstrates the reference building system area.

result scripts

```
All objects = POLYGON ((-5 -5, -5 5, -5 225, -5 235, -5 445, -5 455, 5 455,
 195 455, 205 455, 285 455, 295 455, 425 455, 435 455, 475 455, 485 455, 735 455,
 745 455, 745 445, 745 305, 745 295, 745 5, 745 -5, 735 -5, 435 -5, 425 -5, 245 -5,
 235 -5, 205 -5, 195 -5, 5 -5, -5 -5), (735 445, 485 445, 485 305, 665 305,
 675 305, 735 305, 735 445), (735 295, 675 295, 675 235, 675 225, 675 205, 675 195,
 675 155, 675 145, 665 145, 625 145, 615 145, 615 155, 615 195, 595 195, 585 195,
 435 195, 435 185, 435 175, 435 5, 735 5, 735 295), (665 295, 485 295, 475 295,
 435 295, 435 235, 435 225, 435 205, 585 205, 585 225, 585 235, 595 235, 665 235,
 665 295), (665 225, 595 225, 595 205, 615 205, 625 205, 665 205, 665 225),
 (665 195, 625 195, 625 155, 665 155, 665 195), (475 445, 435 445, 435 305,
 475 305, 475 445), (425 445, 295 445, 295 235, 335 235, 345 235, 365 235, 375 235,
 425 235, 425 295, 425 305, 425 445), (425 225, 375 225, 375 185, 425 185, 425 195,
 425 205, 425 225), (425 175, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135,
 245 125, 245 5, 425 5, 425 175), (365 225, 345 225, 345 185, 365 185, 365 225),
 (335 225, 295 225, 285 225, 235 225, 225 225, 205 225, 195 225, 155 225, 145 225,
 145 235, 155 235, 195 235, 195 265, 195 275, 195 355, 140 355, 130 355, 130 365,
 140 365, 195 365, 195 445, 5 445, 5 235, 85 235, 95 235, 95 225, 85 225, 5 225,
 5 5, 195 5, 195 125, 195 135, 195 175, 195 185, 205 185, 235 185, 245 185,
 335 185, 335 225)), (285 445, 205 445, 205 365, 205 355, 205 275, 225 275, 235 275,
 235 265, 235 235, 285 235, 285 445), (235 175, 205 175, 205 135, 235 135,
 235 175), (235 125, 205 125, 205 5, 235 5, 235 125), (225 265, 205 265, 205 235,
 225 235, 225 265))
Columns, Junction Points & Walls Area = 542.50 square foot

Whole Apartment = POLYGON ((-5 -5, -5 5, -5 225, -5 235, -5 445, -5 455, 5 455,
 195 455, 205 455, 285 455, 295 455, 425 455, 435 455, 475 455, 485 455, 735 455,
 745 455, 745 445, 745 305, 745 295, 745 5, 745 -5, 735 -5, 435 -5, 425 -5, 245 -5,
 235 -5, 205 -5, 195 -5, 5 -5, -5 -5))
Whole Apartment Covered Area = 3450.00 square foot

Usable Area Geometry = MULTIPOLYGON (((735 445, 735 305, 675 305, 665 305, 485 305,
 485 445, 735 445)), ((735 295, 735 5, 435 5, 435 175, 435 185, 435 195, 585 195,
 595 195, 615 195, 615 155, 615 145, 625 145, 665 145, 675 145, 675 155, 675 195,
 675 205, 675 225, 675 235, 675 295, 735 295)), ((665 295, 665 235, 595 235,
 585 235, 585 225, 585 205, 435 205, 435 225, 435 235, 435 295, 475 295, 485 295,
 665 295)), ((665 225, 665 205, 625 205, 615 205, 595 205, 595 225, 665 225)),
 ((665 195, 665 155, 625 155, 625 195, 665 195)), ((475 445, 475 305, 435 305,
 435 445, 475 445)), ((425 445, 425 305, 425 295, 425 235, 375 235, 365 235,
 345 235, 335 235, 295 235, 295 445, 425 445)), ((425 225, 425 205, 425 195,
 425 185, 375 185, 375 225, 425 225)), ((425 175, 425 5, 245 5, 245 125, 245 135,
 245 175, 335 175, 345 175, 365 175, 375 175, 425 175)), ((365 225, 365 185,
 345 185, 345 225, 365 225)), ((335 225, 335 185, 245 185, 235 185, 205 185,
 195 185, 195 175, 195 135, 195 125, 195 5, 5 5, 5 225, 85 225, 95 225, 95 235,
 85 235, 5 235, 5 445, 195 445, 195 365, 140 365, 130 365, 130 355, 140 355,
 195 355, 195 275, 195 265, 195 235, 155 235, 145 235, 145 225, 155 225, 195 225,
 205 225, 225 225, 235 225, 285 225, 295 225, 335 225)), ((285 445, 285 235,
 235 235, 235 265, 235 275, 225 275, 205 275, 205 355, 205 365, 205 445, 285 445)),
 ((235 175, 235 135, 205 135, 205 175, 235 175)), ((235 125, 235 5, 205 5, 205 125,
 235 125)), ((225 265, 225 235, 205 235, 205 265, 225 265)))
Whole Apartment Usable Area = 2907.50 square foot
```

3.3.3 Floorplan Design Area Validation

Geometric considerations require that the sum of the room areas within the apartments balance the whole area calculation. In other words, we require the area for Apartment 1, plus the area for Apartment 2 to balance the usable area for the whole apartment building. We employ a tabular approach to validate that the areas balance, as summarized in Table 3.2.

```
Area of Apartment 1
--------------------------------------------------------

   Living Room : 418.00 square foot.
   Kitchen:      392.50
   Hallway:       72.00
   Bathroom:     162.00
   Bedroom 1:    293.00
   Bedroom 2:    342.00
   Portal 1:       5.00
   Portal 2:       4.00
   ------------------------------------
   Subtotal:    1688.50 square foot.

Area of Apartment 2
--------------------------------------------------------

   Living Room : 616.00 square foot.
   Bathroom:     197.00
   Bedroom:      406.00
   ------------------------------------
   Subtotal:    1219.00 square foot.

Area of Whole Apartment
--------------------------------------------------------

   Area of Apartment 1: 1688.50 square foot.
   Area of Apartment 2: 1219.00
   ------------------------------------
   Subtotal:    2907.50 Square foot
```

Table 3.1: Summary of usable room areas.

3.3.4 System Redesign

Usability of a system is very important from the designer's stand point. Therefore, keeping the functionality and increase the usability under the established regulations became an issue that needs to find an adequate solution. In this section, the exterior wall of the apartment model will be extend to show the capability of increasing the usability of floor plan model design.

In order to extend the usable area of those two apartment unit, we could simply adjust the centerlines to expend exterior walls and some interior ones. Expand centerline 16 right and 26 up will make the whole apartment exterior boundary bigger. By the propagation of dependency, the junction points whose parent is centerline 16 and 26 will shift as well. After the exterior wall modification, we can adjust the interior walls to make the floor plan model more ideal. The result script below shows the new components model of redesign apartment.

result scripts

```
Create vertical Centerline  : x = 460.0
Create vertical Centerline  : x = 510.0
Create vertical Centerline  : x = 820.0
Create horizontal Centerline: y = 250.0
Create horizontal Centerline: y = 290.0
Create horizontal Centerline: y = 500.0
========================================================
Create Composite Hierarchy workspace:
       Location: (465.0, -5.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (235.0, 245.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (295.0, 245.0)
       Rotation: 0.0
```

```
========================================================
Create Composite Hierarchy workspace:
       Location: (375.0, 245.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (5.0, 495.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (295.0, 495.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (485.0, 495.0)
       Rotation: 0.0
========================================================
Create Composite Hierarchy workspace:
       Location: (225.0, 285.0)
       Rotation: -1.5707963267948966
========================================================
Create Composite Hierarchy workspace:
       Location: (505.0, 495.0)
       Rotation: -1.5707963267948966
==================================================================================
Apartment 1 Living Room = POLYGON ((-5 -5, -5 5, -5 245, -5 255, 5 255, 85 255,
 95 255, 145 255, 155 255, 195 255, 205 255, 205 245, 205 185, 205 175, 205 135,
 205 125, 205 5, 205 -5, 195 -5, 5 -5, -5 -5), (195 5, 195 125, 195 135, 195 175,
 195 185, 195 245, 155 245, 145 245, 95 245, 85 245, 5 245, 5 5, 195 5))
Columns, Junction Points & Walls Area = 90.00 Square foot

Whole Geometry = POLYGON ((-5 -5, -5 5, -5 245, -5 255, 5 255, 85 255, 95 255,
 145 255, 155 255, 195 255, 205 255, 205 245, 205 185, 205 175, 205 135, 205 125,
 205 5, 205 -5, 195 -5, 5 -5, -5 -5))
Apartment 1 Living Room Covered Area = 546.00 Square foot

Usable Area Geometry = POLYGON ((195 5, 5 5, 5 245, 85 245, 95 245, 145 245,
 155 245, 195 245, 195 185, 195 175, 195 135, 195 125, 195 5))
Apartment 1 Living Room Usable Area = 456.00 Square foot
==================================================================================
Apartment 1 Kitchen = POLYGON ((-5 245, -5 255, -5 495, -5 505, 5 505, 195 505,
 205 505, 205 495, 205 365, 205 355, 205 295, 205 285, 205 255, 205 245, 195 245,
 155 245, 145 245, 95 245, 85 245, 5 245, -5 245), (85 255, 95 255, 145 255,
 155 255, 195 255, 195 285, 195 295, 195 355, 140 355, 130 355, 130 365, 140 365,
 195 365, 195 495, 5 495, 5 255, 85 255))
Columns, Junction Points & Walls Area = 96.50 Square foot

Whole Geometry = POLYGON ((-5 245, -5 255, -5 495, -5 505, 5 505, 195 505, 205 505,
 205 495, 205 365, 205 355, 205 295, 205 285, 205 255, 205 245, 195 245, 155 245,
 145 245, 95 245, 85 245, 5 245, -5 245))
Apartment 1 Kitchen Covered Area = 546.00 Square foot

Usable Area Geometry = POLYGON ((85 255, 5 255, 5 495, 195 495, 195 365, 140 365,
 130 365, 130 355, 140 355, 195 355, 195 295, 195 285, 195 255, 155 255, 145 255,
 95 255, 85 255))
Apartment 1 Kitchen Usable Area = 449.50 Square foot
==================================================================================
Apartment 1 Hallway = POLYGON ((195 125, 195 135, 195 175, 195 185, 195 245,
 195 255, 205 255, 225 255, 235 255, 285 255, 295 255, 335 255, 345 255, 365 255,
 375 255, 375 245, 375 185, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135,
 245 125, 235 125, 205 125, 195 125), (235 135, 235 175, 205 175, 205 135,
 235 135), (235 185, 245 185, 335 185, 335 245, 295 245, 285 245, 235 245, 225 245,
 205 245, 205 185, 235 185), (365 185, 365 245, 345 245, 345 185, 365 185))
Columns, Junction Points & Walls Area = 67.00 Square foot

Whole Geometry = POLYGON ((195 125, 195 135, 195 175, 195 185, 195 245, 195 255,
 205 255, 225 255, 235 255, 285 255, 295 255, 335 255, 345 255, 365 255, 375 255,
 375 245, 375 185, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135, 245 125,
 235 125, 205 125, 195 125))
Apartment 1 Hallway Covered Area = 169.00 Square foot

Usable Area Geometry = MULTIPOLYGON (((235 135, 205 135, 205 175, 235 175,
 235 135)), ((235 185, 205 185, 205 245, 225 245, 235 245, 285 245, 295 245,
 335 245, 335 185, 245 185, 235 185)), ((365 185, 345 185, 345 245, 365 245,
 365 185)))
Apartment 1 Hallway Usable Area = 102.00 Square foot
========================================================
Apartment 1 Bath = POLYGON ((195 245, 195 255, 195 285, 195 295, 195 355, 195 365,
 195 495, 195 505, 205 505, 285 505, 295 505, 295 495, 295 255, 295 245, 285 245,
 235 245, 225 245, 205 245, 195 245), (235 255, 285 255, 285 495, 205 495, 205 365,
 205 355, 205 295, 225 295, 235 295, 235 285, 235 255), (225 285, 205 285, 205 255,
 225 255, 225 285))
Columns, Junction Points & Walls Area = 74.00 Square foot

Whole Geometry = POLYGON ((195 245, 195 255, 195 285, 195 295, 195 355, 195 365,
 195 495, 195 505, 205 505, 285 505, 295 505, 295 495, 295 255, 295 245, 285 245,
 235 245, 225 245, 205 245, 195 245))
Apartment 1 Bath Covered Area = 260.00 Square foot

Usable Area Geometry = MULTIPOLYGON (((235 255, 235 285, 235 295, 225 295, 205 295,
 205 355, 205 365, 205 495, 285 495, 285 255, 235 255)), ((225 285, 225 255,
 205 255, 205 285, 225 285)))
Apartment 1 Bath Usable Area = 186.00 Square foot
========================================================
Apartment 1 Bedroom 1 = POLYGON ((285 245, 285 255, 285 495, 285 505, 295 505,
 455 505, 465 505, 465 495, 465 295, 465 285, 465 255, 465 245, 465 205, 465 195,
 465 185, 465 175, 455 175, 375 175, 365 175, 365 185, 365 245, 345 245, 335 245,
 295 245, 285 245), (335 255, 345 255, 365 255, 375 255, 455 255, 455 285, 455 295,
 455 495, 295 495, 295 255, 335 255), (375 245, 375 185, 455 185, 455 195, 455 205,
 455 245, 375 245))
Columns, Junction Points & Walls Area = 106.00 Square foot

Whole Geometry = POLYGON ((285 245, 285 255, 285 495, 285 505, 295 505, 455 505,
 465 505, 465 495, 465 295, 465 285, 465 255, 465 245, 465 205, 465 195, 465 185,
 465 175, 455 175, 375 175, 365 175, 365 185, 365 245, 345 245, 335 245, 295 245,
 285 245))
Apartment 1 Bedroom 1 Covered Area = 538.00 Square foot
```

```
Usable Area Geometry = MULTIPOLYGON (((335 255, 295 255, 295 495, 455 495, 455 295,
 455 285, 455 255, 375 255, 365 255, 345 255, 335 255)), ((375 245, 455 245,
 455 205, 455 195, 455 185, 375 185, 375 245)))
Apartment 1 Bedroom 1 Usable Area = 432.00 Square foot
=========================================================
Apartment 2 Living Room = POLYGON ((455 -5, 455 5, 455 175, 455 185, 455 195,
 455 205, 465 205, 585 205, 595 205, 615 205, 625 205, 665 205, 665 225, 665 235,
 665 285, 665 295, 675 295, 815 295, 825 295, 825 285, 825 5, 825 -5, 815 -5,
 465 -5, 455 -5), (815 5, 815 285, 675 285, 675 235, 675 225, 675 205, 675 195,
 675 155, 675 145, 665 145, 625 145, 615 145, 615 155, 615 195, 595 195, 585 195,
 465 195, 465 185, 465 175, 465 5, 815 5), (665 155, 665 195, 625 195, 625 155,
 665 155))
Columns, Junction Points & Walls Area = 144.00 Square foot

Whole Geometry = POLYGON ((455 -5, 455 5, 455 175, 455 185, 455 195, 455 205,
 465 205, 585 205, 595 205, 615 205, 625 205, 665 205, 665 225, 665 235, 665 285,
 665 295, 675 295, 815 295, 825 295, 825 285, 825 5, 825 -5, 815 -5, 465 -5,
 455 -5))
Apartment 2 Living Room Covered Area = 921.00 Square foot

Usable Area Geometry = MULTIPOLYGON (((815 5, 465 5, 465 175, 465 185, 465 195,
 585 195, 595 195, 615 195, 615 155, 615 145, 625 145, 665 145, 675 145, 675 155,
 675 195, 675 205, 675 225, 675 235, 675 285, 815 285, 815 5)), ((665 155, 625 155,
 625 195, 665 195, 665 155)))
Apartment 2 Living Room Usable Area = 777.00 Square foot
=========================================================
Apartment 2 Bedroom = POLYGON ((455 285, 455 295, 455 495, 455 505, 465 505,
 505 505, 515 505, 815 505, 825 505, 825 495, 825 295, 825 285, 815 285, 675 285,
 665 285, 515 285, 505 285, 465 285, 455 285), (515 295, 665 295, 675 295, 815 295,
 815 495, 515 495, 515 295), (505 495, 465 495, 465 295, 505 295, 505 495))
Columns, Junction Points & Walls Area = 134.00 Square foot

Whole Geometry = POLYGON ((455 285, 455 295, 455 495, 455 505, 465 505, 505 505,
 515 505, 815 505, 825 505, 825 495, 825 295, 825 285, 815 285, 675 285, 665 285,
 515 285, 505 285, 465 285, 455 285))
Apartment 2 Bedroom Covered Area = 814.00 Square foot

Usable Area Geometry = MULTIPOLYGON (((515 295, 515 495, 815 495, 815 295, 675 295,
 665 295, 515 295)), ((505 495, 505 295, 465 295, 465 495, 505 495)))
Apartment 2 Bedroom Usable Area = 680.00 Square foot
=========================================================
All objects = POLYGON ((-5 -5, -5 5, -5 245, -5 255, -5 495, -5 505, 5 505,
 195 505, 205 505, 285 505, 295 505, 455 505, 465 505, 505 505, 515 505, 815 505,
 825 505, 825 495, 825 295, 825 285, 825 5, 825 -5, 815 -5, 465 -5, 455 -5, 245 -5,
 235 -5, 205 -5, 195 -5, 5 -5, -5 -5), (815 495, 515 495, 515 295, 665 295,
 675 295, 815 295, 815 495), (815 285, 675 285, 675 235, 675 225, 675 205, 675 195,
 675 155, 675 145, 665 145, 625 145, 615 145, 615 155, 615 195, 595 195, 585 195,
 465 195, 465 185, 465 175, 465 5, 815 5, 815 285), (665 285, 515 285, 505 285,
 465 285, 465 255, 465 245, 465 205, 585 205, 585 225, 585 235, 595 235, 665 235,
 665 285), (665 225, 595 225, 595 205, 615 205, 625 205, 665 205, 665 225),
 (665 195, 625 195, 625 155, 665 155, 665 195), (505 495, 465 495, 465 295,
 505 295, 505 495), (455 495, 295 495, 295 255, 335 255, 345 255, 365 255, 375 255,
 455 255, 455 285, 455 295, 455 495), (455 245, 375 245, 375 185, 455 185, 455 195,
 455 205, 455 245), (455 175, 375 175, 365 175, 345 175, 335 175, 245 175, 245 135,
 245 125, 245 5, 455 5, 455 175), (365 245, 345 245, 345 185, 365 185, 365 245),
 (335 245, 295 245, 285 245, 235 245, 225 245, 205 245, 195 245, 155 245, 145 245,
 145 255, 155 255, 195 255, 195 285, 195 295, 195 355, 140 355, 130 355, 130 365,
 140 365, 195 365, 195 495, 5 495, 5 255, 85 255, 95 255, 95 245, 85 245, 5 245,
 5 5, 195 5, 195 125, 195 135, 195 175, 195 185, 205 185, 235 185, 245 185,
 335 185, 335 245), (285 495, 205 495, 205 365, 205 355, 205 295, 225 295, 235 295,
 235 285, 235 255, 285 255, 285 495), (235 175, 205 175, 205 135, 235 135,
 235 175), (235 125, 205 125, 205 5, 235 5, 235 125), (225 285, 205 285, 205 255,
 225 255, 225 285))
Columns, Junction Points & Walls Area = 596.50 Square foot

Whole Apartment = POLYGON ((-5 -5, -5 5, -5 245, -5 255, -5 495, -5 505, 5 505,
 195 505, 205 505, 285 505, 295 505, 455 505, 465 505, 505 505, 515 505, 815 505,
 825 505, 825 495, 825 295, 825 285, 825 5, 825 -5, 815 -5, 465 -5, 455 -5, 245 -5,
 235 -5, 205 -5, 195 -5, 5 -5, -5 -5))
Whole Apartment Covered Area = 4233.00 Square foot

Usable Area Geometry = MULTIPOLYGON (((815 495, 815 295, 675 295, 665 295, 515 295,
 515 495, 815 495)), ((815 285, 815 5, 465 5, 465 175, 465 185, 465 195, 585 195,
 595 195, 615 195, 615 155, 615 145, 625 145, 665 145, 675 145, 675 155, 675 195,
 675 205, 675 225, 675 235, 675 285, 815 285)), ((665 285, 665 235, 595 235,
 585 235, 585 225, 585 205, 465 205, 465 245, 465 255, 465 285, 505 285, 515 285,
 665 285)), ((665 225, 665 205, 625 205, 615 205, 595 205, 595 225, 665 225)),
 ((665 195, 665 155, 625 155, 625 195, 665 195)), ((505 495, 505 295, 465 295,
 465 495, 505 495)), ((455 495, 455 295, 455 285, 455 255, 375 255, 365 255,
 345 255, 335 255, 295 255, 295 495, 455 495)), ((455 245, 455 205, 455 195,
 455 185, 375 185, 375 245, 455 245)), ((455 175, 455 5, 245 5, 245 125, 245 135,
 245 175, 335 175, 345 175, 365 175, 375 175, 455 175)), ((365 245, 365 185,
 345 185, 345 245, 365 245)), ((335 245, 335 185, 245 185, 235 185, 205 185,
 195 185, 195 175, 195 135, 195 125, 195 5, 5 5, 5 245, 85 245, 95 245, 95 255,
 85 255, 5 255, 5 495, 195 495, 195 365, 140 365, 130 365, 130 355, 140 355,
 195 355, 195 295, 195 285, 195 255, 155 255, 145 255, 145 245, 155 245, 195 245,
 205 245, 225 245, 235 245, 285 245, 295 245, 335 245)), ((285 495, 285 255,
 235 255, 235 285, 235 295, 225 295, 205 295, 205 355, 205 365, 205 495, 285 495)),
 ((235 175, 235 135, 205 135, 205 175, 235 175)), ((235 125, 235 5, 205 5, 205 125,
 235 125)), ((225 285, 225 255, 205 255, 205 285, 225 285)))
Whole Apartment Usable Area = 3636.50 Square foot
```

Again, an easy way of validating the area computations for the system redesign is with a tabular layout. See Table 3.2.

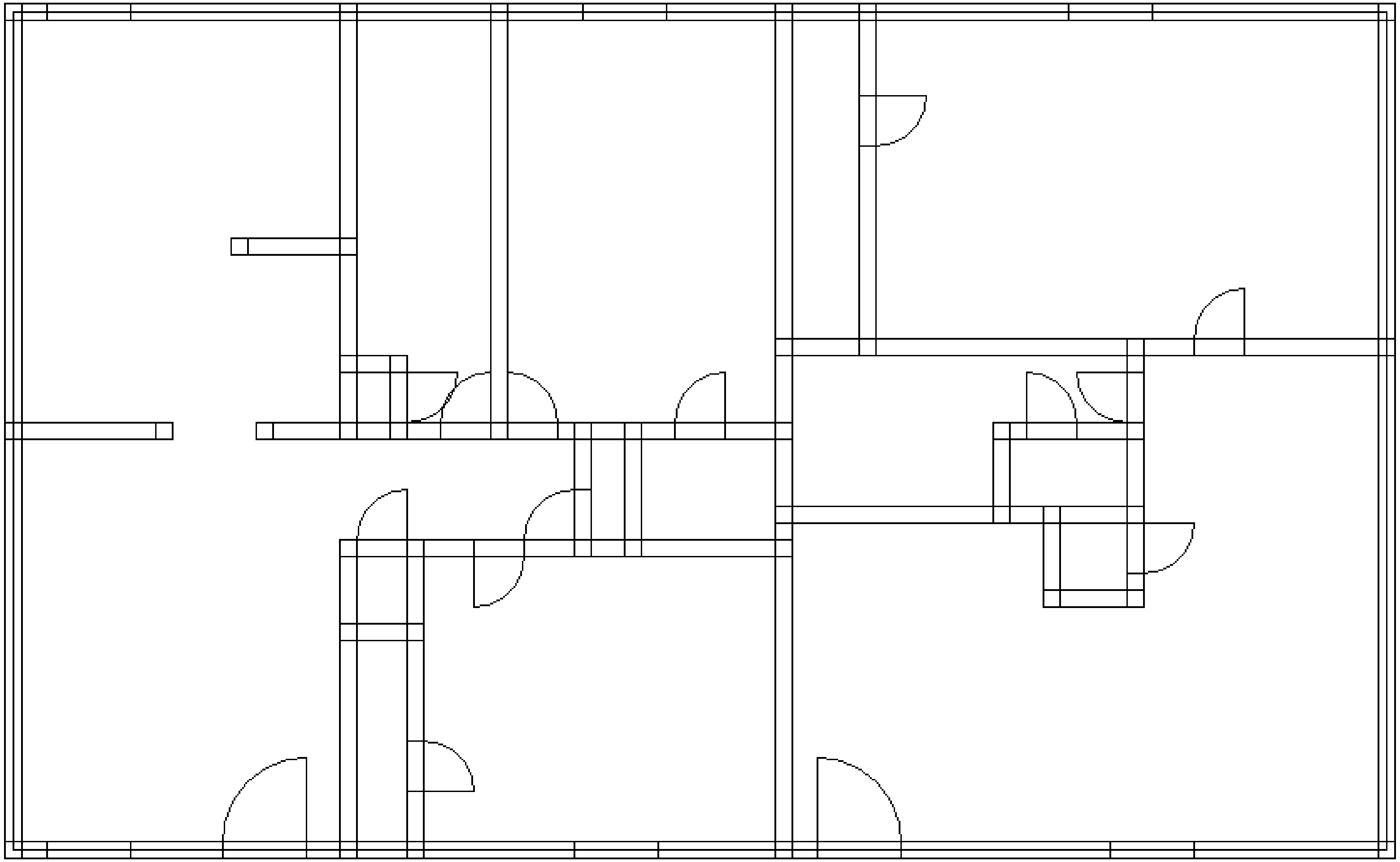


Figure 3.7: The redesign of the simple room example.

```
--------------------------------------------------------
Area of Apartment 1
--------------------------------------------------------

   Living Room : 456.00 square foot.
   Kitchen:      449.50
   Hallway:      102.00
   Bathroom:     186.00
   Bedroom 1:    432.00
   Bedroom 2:    393.00
   Portal 1:       5.00
   Portal 2:       6.00
   ------------------------------------
   Subtotal:    2029.50 square foot.

--------------------------------------------------------
Area of Apartment 2
--------------------------------------------------------

   Living Room : 777.00 square foot.
   Bathroom:     150.00
   Bedroom:      680.00
   ------------------------------------
   Subtotal:    1607.00 square foot.

--------------------------------------------------------
Area of Whole Apartment
--------------------------------------------------------

   Area of Apartment 1: 2029.50 square foot.
   Area of Apartment 2: 1607.00
   ------------------------------------
   Subtotal:    3636.50 square foot.
```

Table 3.2: Summary of usable room areas for the system redesign.

3.4 Assessment of Approach 1

In computer science circles, the appeal of scripting languages is that they provide high-level solutions to problems that involve systems integration and/or require incremental development. So as a first cut to addressing the step-by-specification of building floorplan layouts, scripting seems like a good idea. Unfortunately, this approach runs into trouble because of the shear complexity of the problem: buildings containing thousands of elements are common; modeling abstractions are organized into hierarchies; strong dependency relationships exist between various types of elements. And, yet, the whole apartment problem specification requires approximately 10,000 lines of Java. We need to find a better approach.

Chapter 4

Approach 2: Interactive Graphical Specification of Floorplans

4.1 Problem Statement

While the last chapter demonstrated that parametric modeling concepts can be used in building floorplan systems, unfortunately, the approach is very low level, tedious, and not readily scalable. To put the problem in perspective, approximately 10,000 lines of Java code were needed to specify the geometry and parametric dependency relationships for the small house example.

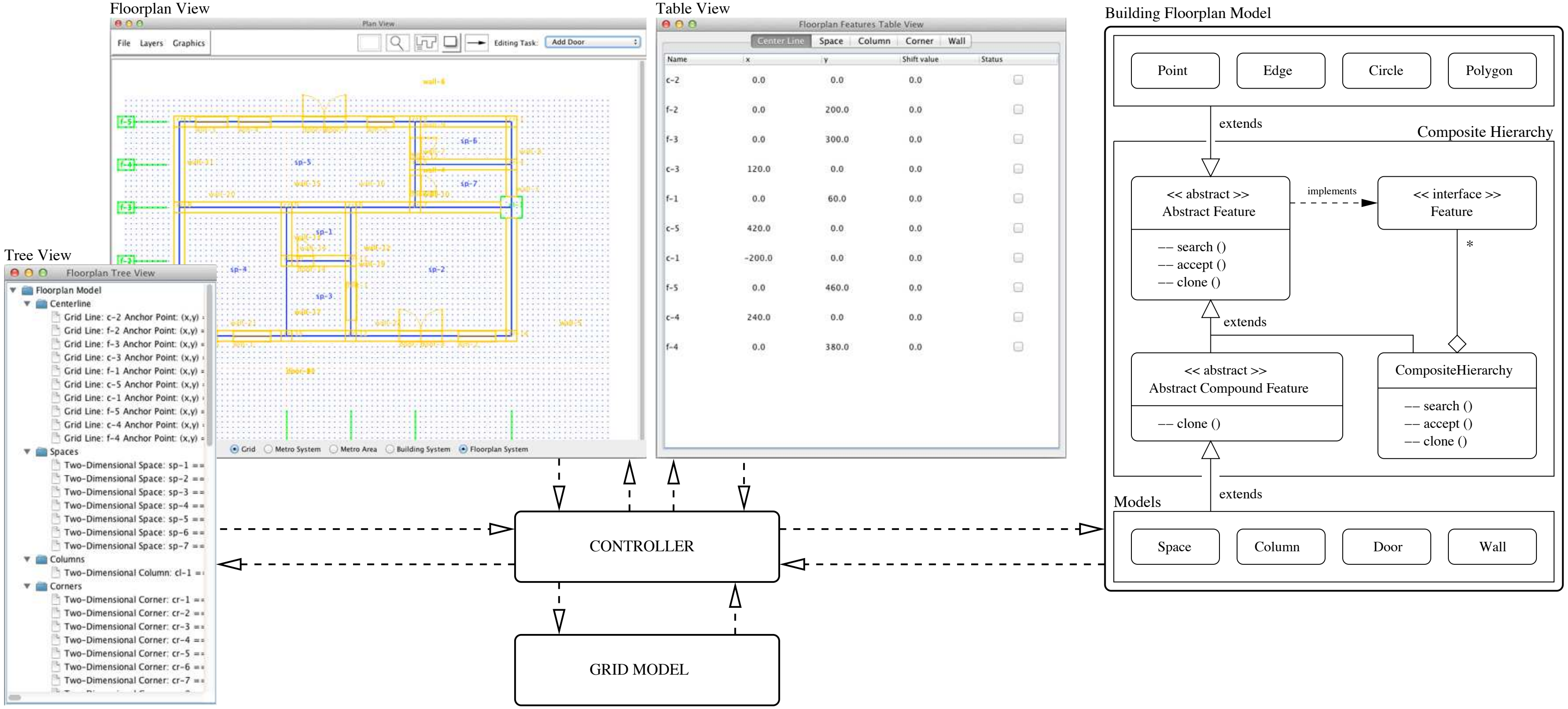
In an attempt to overcome this problem, in this chapter we develop an interactive graphical-based modeling technique for the specification of floorplans. In contrast to the first approach, which used a bottom-up approach to floor plan specification, the second approach allows designers to work through a top-down specification of floor plans. The procedure begins with a graphical specification of spaces; columns are added to the corner points of spaces; walls are added to the edges joining columns; doors and walls are inserted into walls. Groups of spaces can be assigned to rooms. Algorithms are developed to compute the area of rooms. Because floorplan systems are defined by a multitude of component types and spatial entities, we need to be disciplined in the software development. We address this problem through a novel use software design patterns.

4.2 Graphical User Interface Design and Implementation

Figure 4.1 shows the essential details of the system architecture and composite hierarchy framework for modeling and visualizing building floorplans. We employ the MVC software design pattern to links the models, views and controllers together. To simplify the details of implementation, the discipline-specific models, views and controllers are concrete extensions of abstract implementations. The primary purpose of the abstract-level specifications is to take care of the details of model-controller and view-controller communication.

Domain specific models are extensions of abstract model. Domain specific views are extensions of abstract view. And the engineering controller is an extension of the abstract controller. A key benefit in using the visitor design pattern is that format-specific views can be designed independently of the application at hand. Also notice that a model is provided for the coordinate system grid against which floorplan coordinates are defined. Our goal is to provide mechanisms where an engineering will interact with a view, primarily the engineering view, and the results of edit operations will be sent to the engineering controller. All of the registered models will receive the details of an edit operation, update the model (or models) accordingly, and propagate the new details to the views.

Composite Hierarchy Floorplan Model. Figure 2.11 is a class diagram for the



Floorplan Tree View

- Floorplan Model
  - Centerline
    - Grid Line: c–2 Anchor Point: (x,y) =
    - Grid Line: f–2 Anchor Point: (x,y) =
    - Grid Line: f–3 Anchor Point: (x,y) =
    - Grid Line: c–3 Anchor Point: (x,y) =
    - Grid Line: f–1 Anchor Point: (x,y) =
    - Grid Line: c–5 Anchor Point: (x,y) =
    - Grid Line: c–1 Anchor Point: (x,y) =
    - Grid Line: f–5 Anchor Point: (x,y) =
    - Grid Line: c–4 Anchor Point: (x,y) =
    - Grid Line: f–4 Anchor Point: (x,y) =
  - Spaces
    - Two–Dimensional Space: sp–1 ==
    - Two–Dimensional Space: sp–2 ==
    - Two–Dimensional Space: sp–3 ==
    - Two–Dimensional Space: sp–4 ==
    - Two–Dimensional Space: sp–5 ==
    - Two–Dimensional Space: sp–6 ==
    - Two–Dimensional Space: sp–7 ==
  - Columns
    - Two–Dimensional Column: cl–1 ==
  - Corners
    - Two–Dimensional Corner: cr–1 ==
    - Two–Dimensional Corner: cr–2 ==
    - Two–Dimensional Corner: cr–3 ==
    - Two–Dimensional Corner: cr–4 ==
    - Two–Dimensional Corner: cr–5 ==
    - Two–Dimensional Corner: cr–6 ==
    - Two–Dimensional Corner: cr–7 ==

Floorplan Features Table View

Center Line | Space | Column | Corner | Wall

Name	x	y	Shift value	Status
c–2	0.0	0.0	0.0	□
f–2	0.0	200.0	0.0	□
f–3	0.0	300.0	0.0	□
c–3	120.0	0.0	0.0	□
f–1	0.0	60.0	0.0	□
c–5	420.0	0.0	0.0	□
c–1	-200.0	0.0	0.0	□
f–5	0.0	460.0	0.0	□
c–4	240.0	0.0	0.0	□
f–4	0.0	380.0	0.0	□

Figure 4.1: Floorplan editor collage.

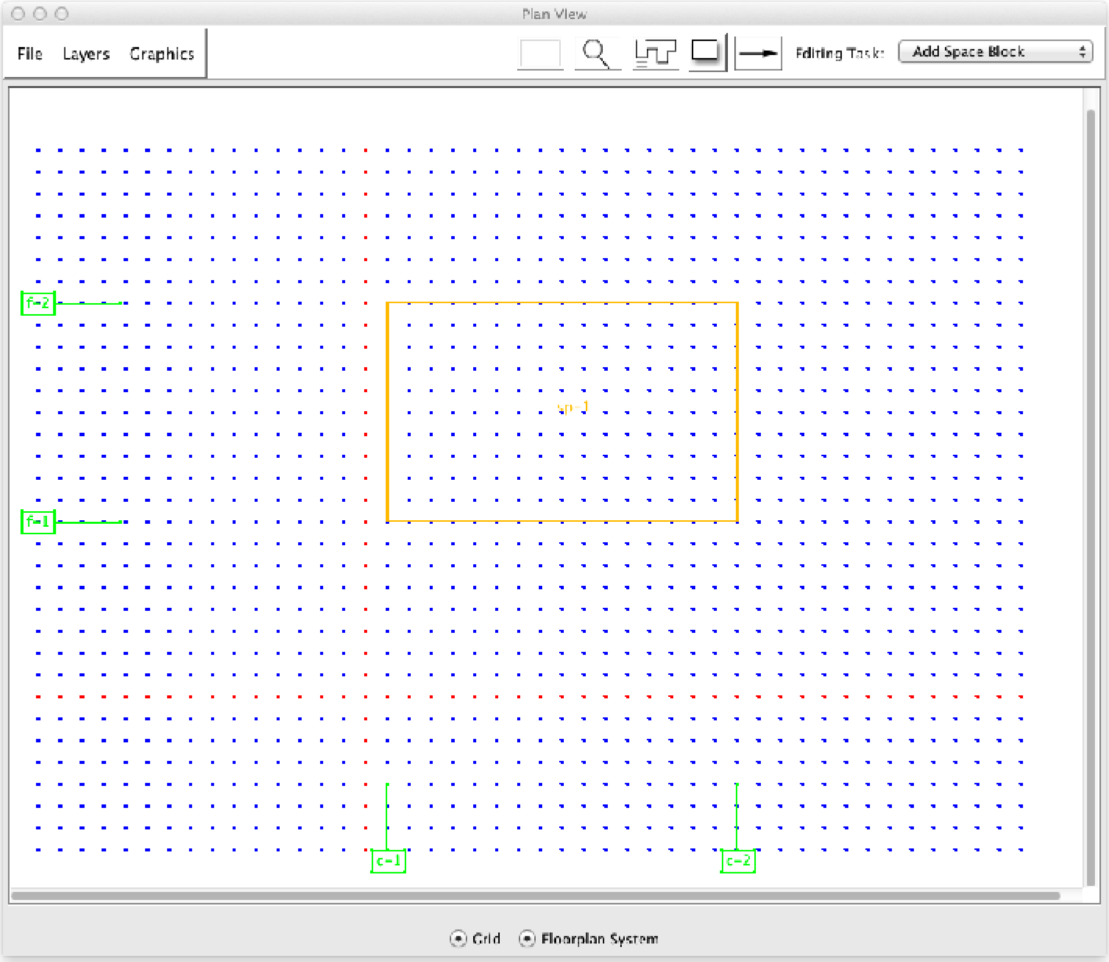


Figure 4.2: Graphical representation of the centerline model inside the software.

composite hierarchy software pattern representation of building floorplans.

Centerline Model. Figure 4.2 is a screen-dump of the graphical representation of the centerline model employed by the floorplan model.

4.3 Example 1. Graphical Specification for a Simple Room

In Section 3.3 we presented a procedure for scripting a simple room floor plan. In this section, a step-by-step instruction will be presented to show how the graphical specification software helps users create a mutable floor plan model.

Precondition setup in editor view:

1. Click the "Grid" and "Floorplan System" radio button at the bottom of the editor view to activate the display of the grid and floor plan as shown below.

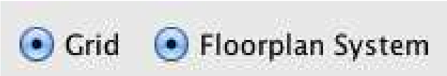


Figure 4.3: Graphical display radio buttons inside the editor view of the software.

2. Select "Options", "Snap to Grid" to force the point to the nearest grid.

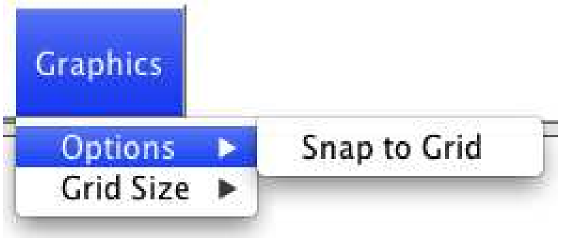


Figure 4.4: Snap to Grid setup inside the editor view of the software.

3. Select "Graphics", "Grid Size", "20" to set the grid size to 20 pixels.

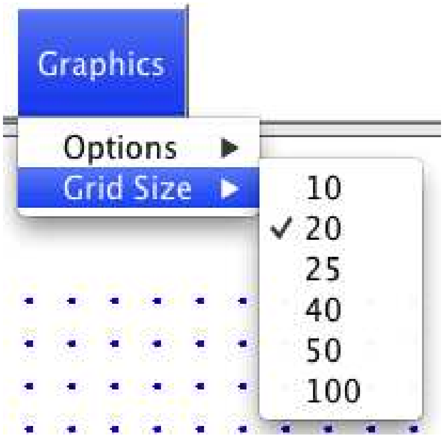


Figure 4.5: Grid size setup inside the editor view of the software.

Step 1. Select "Add Space Block" in the Combo box at upper-right of the editor view and click ”Draw Rectangle” button. Then, click and drag on the grid to create a rectangle space of the floor plan.

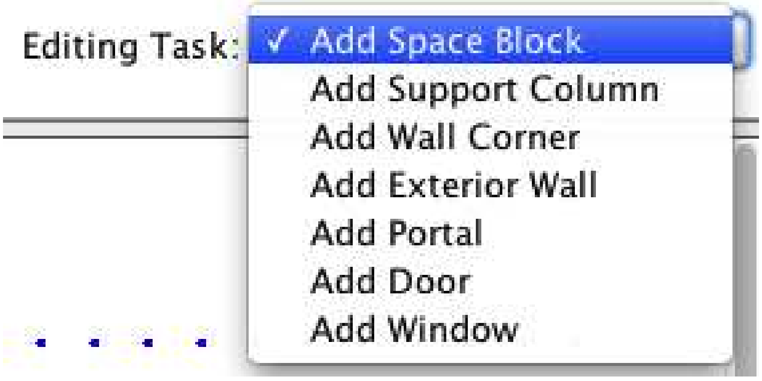


Figure 4.6: Floor plan components combo box inside the editor view of the software.

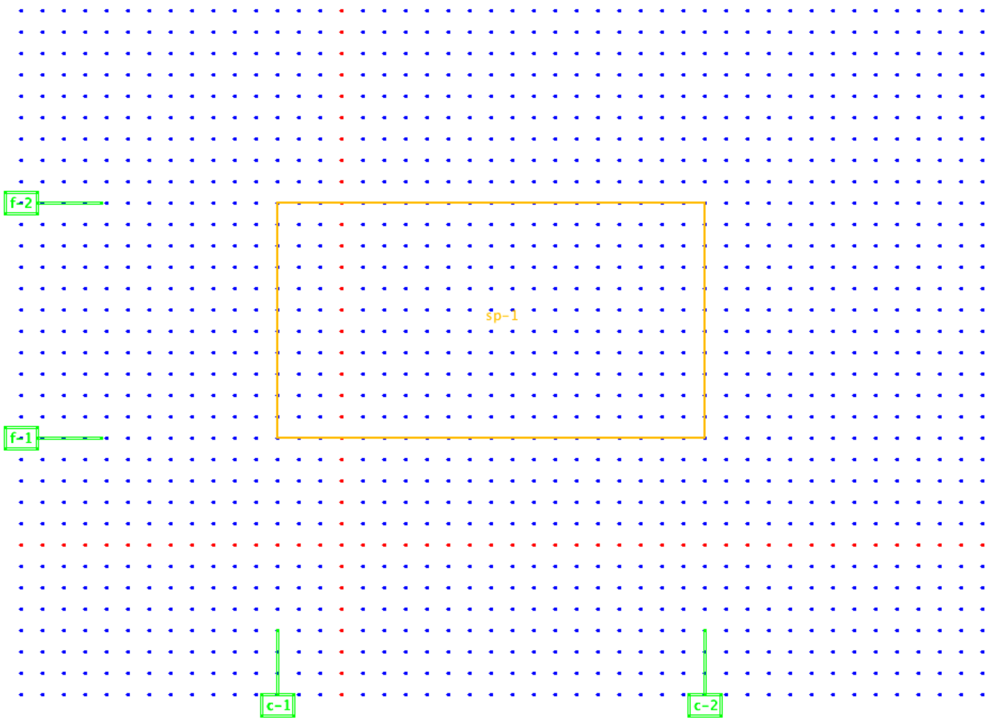


Figure 4.7: Space component added into the floor plan model.

Step 2. Select ”Add Support Column” in the Combo box at the upper-right of the editor view. Then, click and drag to include the upper-right corner of the Space that created in Step 1. The results are shown in Figure 4.8.

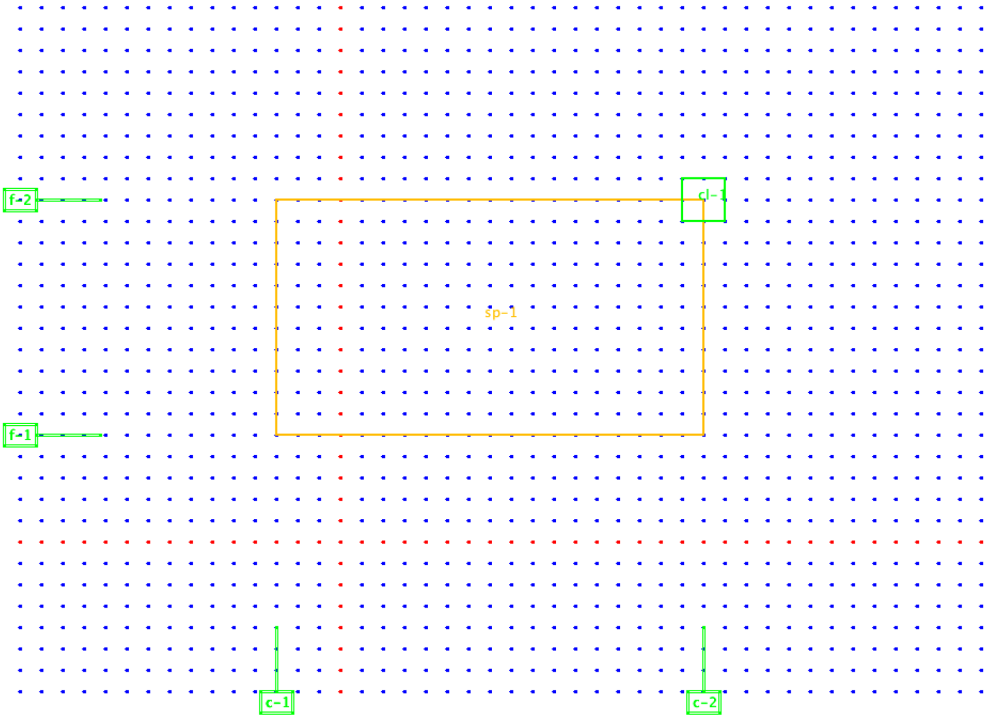


Figure 4.8: Column component added into the floor plan model.

Step 3. Change the grid size to 10 pixels to have a thinner wall and select "Add Wall Corner" in the Combo box. Then, click and drag to include other corners of the Space.

Step 4. Select "Add Exterior Wall" in the Combo box. Then, click and drag to include all components to create all walls around the area. See Figure 4.10.

Step 5. Go to table view and click on the Space tab at the top of the window and select the space in the table. Then, enter the name of the room and role of the room and click the "Define Room" button to define the room.

Step 6. Click on the Room tab to verify the room that defined in Step 5 is created. See Figure 4.12.

Step 7. Click on the floorplan tab and insert the data shown in Table 4.1. Then,

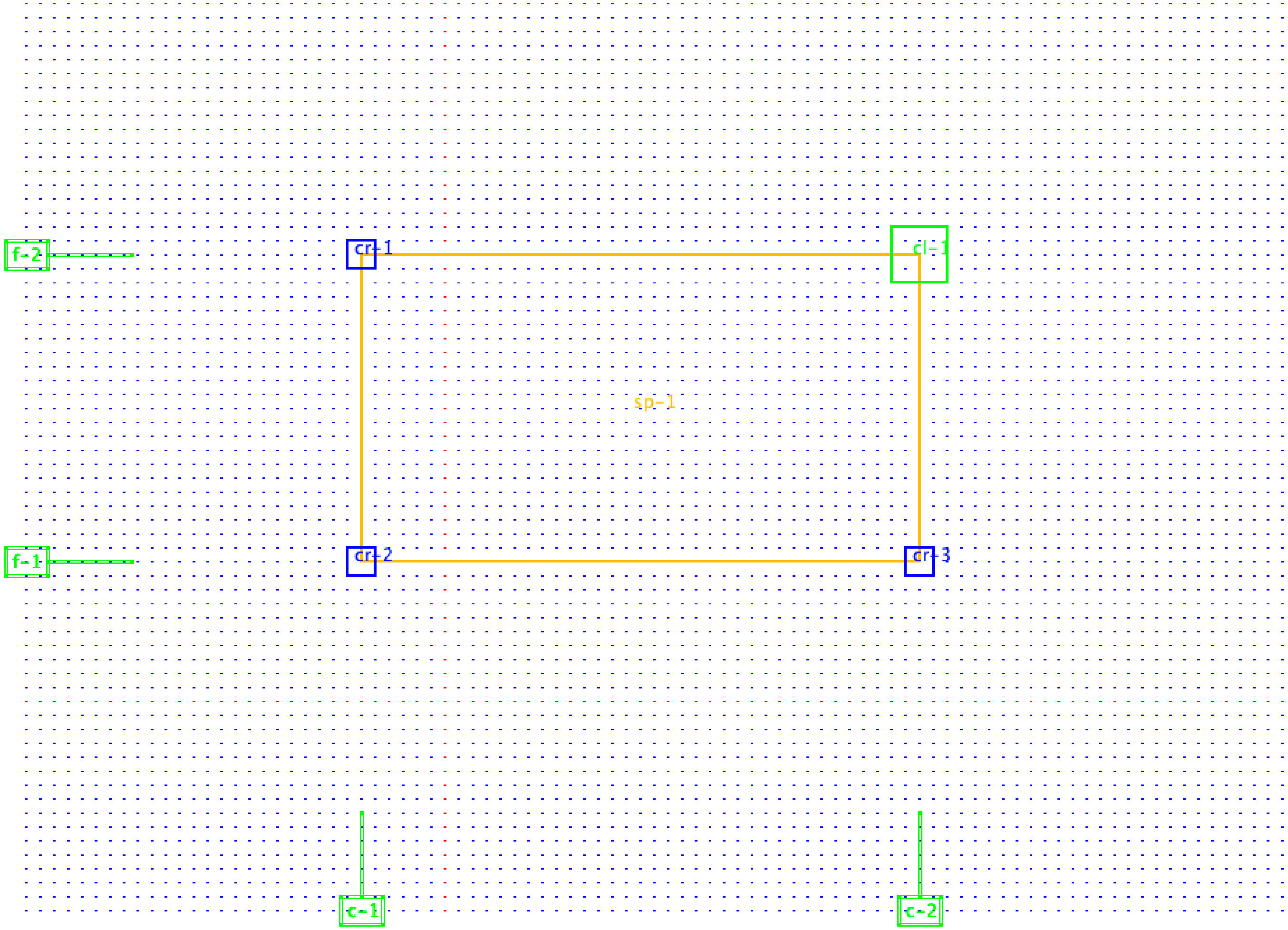


Figure 4.9: Corner components added into the floor plan model.

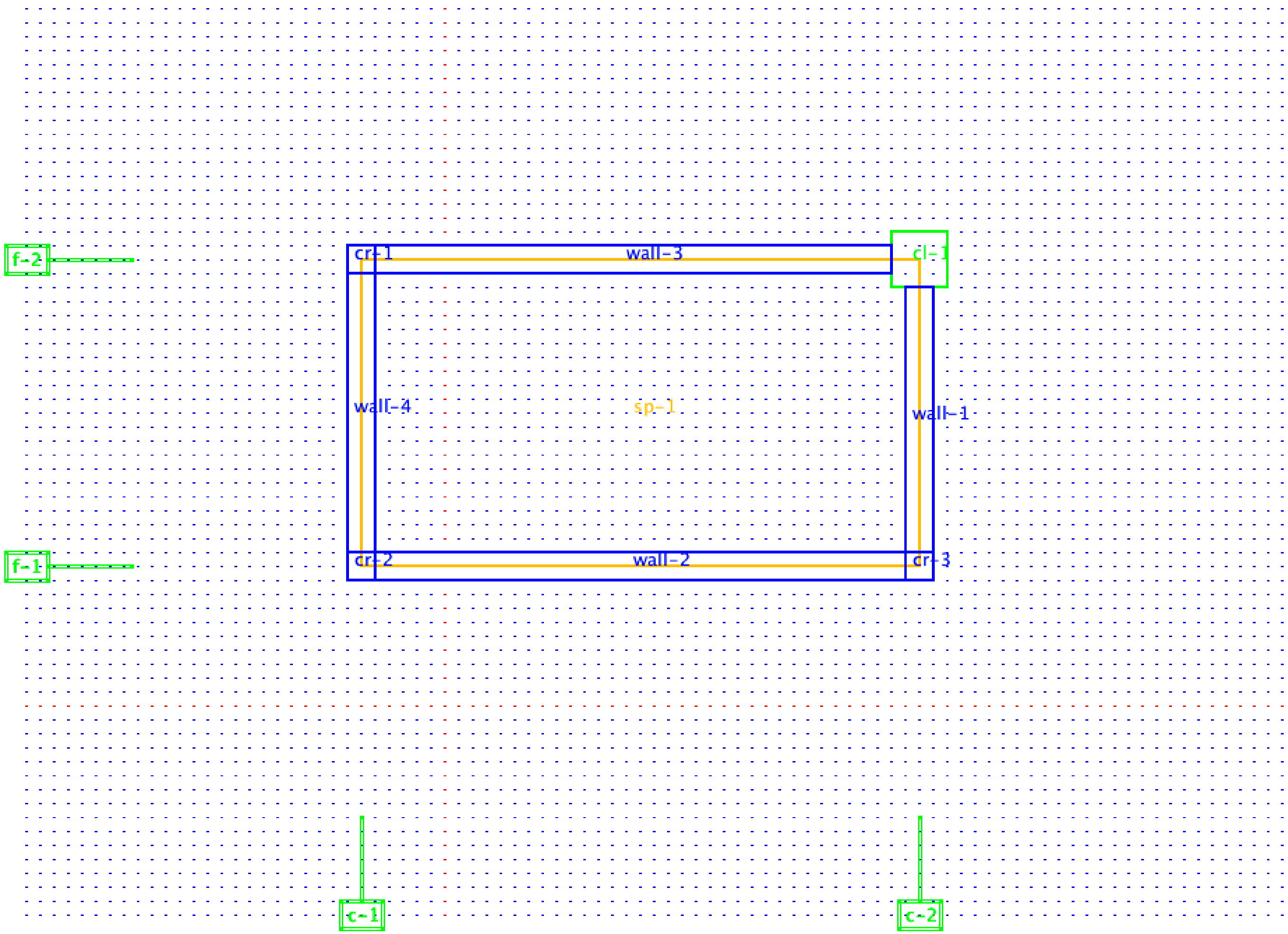
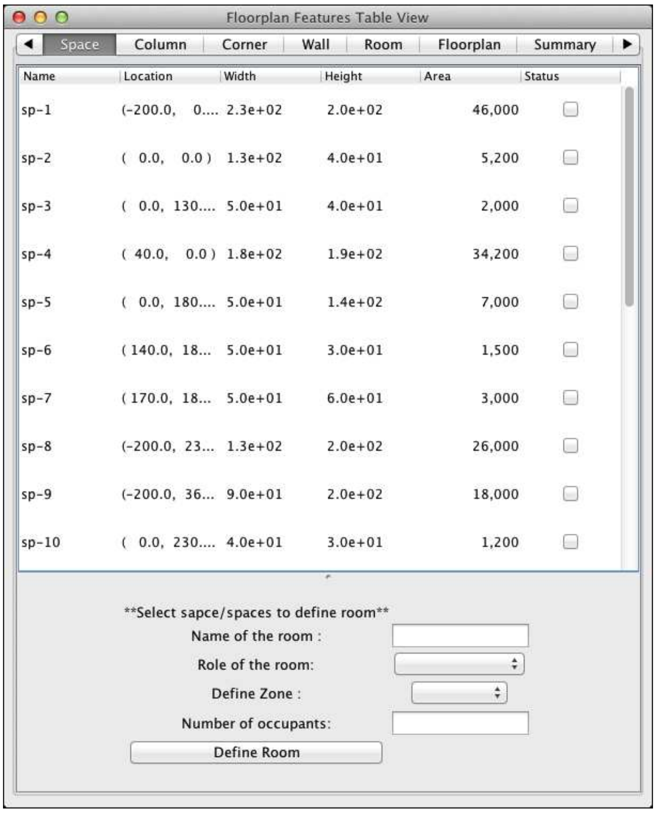


Figure 4.10: Wall components added into the floor plan model.



Name	Location	Width	Height	Area	Status
sp-1	(-200.0, 0....	2.3e+02	2.0e+02	46,000	□
sp-2	(0.0, 0.0)	1.3e+02	4.0e+01	5,200	□
sp-3	(0.0, 130....	5.0e+01	4.0e+01	2,000	□
sp-4	(40.0, 0.0)	1.8e+02	1.9e+02	34,200	□
sp-5	(0.0, 180....	5.0e+01	1.4e+02	7,000	□
sp-6	(140.0, 18...	5.0e+01	3.0e+01	1,500	□
sp-7	(170.0, 18...	5.0e+01	6.0e+01	3,000	□
sp-8	(-200.0, 23...	1.3e+02	2.0e+02	26,000	□
sp-9	(-200.0, 36...	9.0e+01	2.0e+02	18,000	□
sp-10	(0.0, 230....	4.0e+01	3.0e+01	1,200	□

Select sapce/spaces to define room

Name of the room :

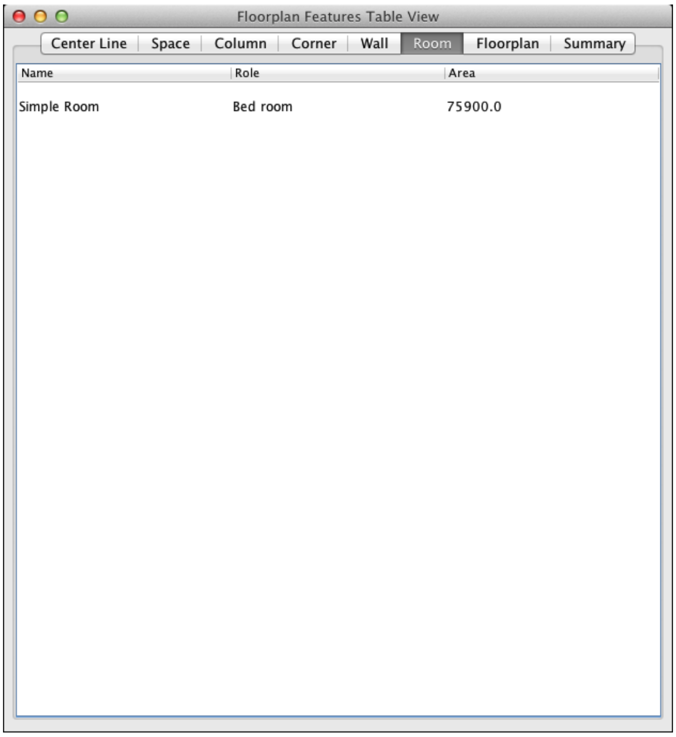
Role of the room:

Define Zone :

Number of occupants:

Define Room

Figure 4.11: Space tab inside the table view of the software.



Name	Role	Area
Simple Room	Bed room	75900.0

Figure 4.12: Room tab inside the table view of the software.

click "Set values" button.

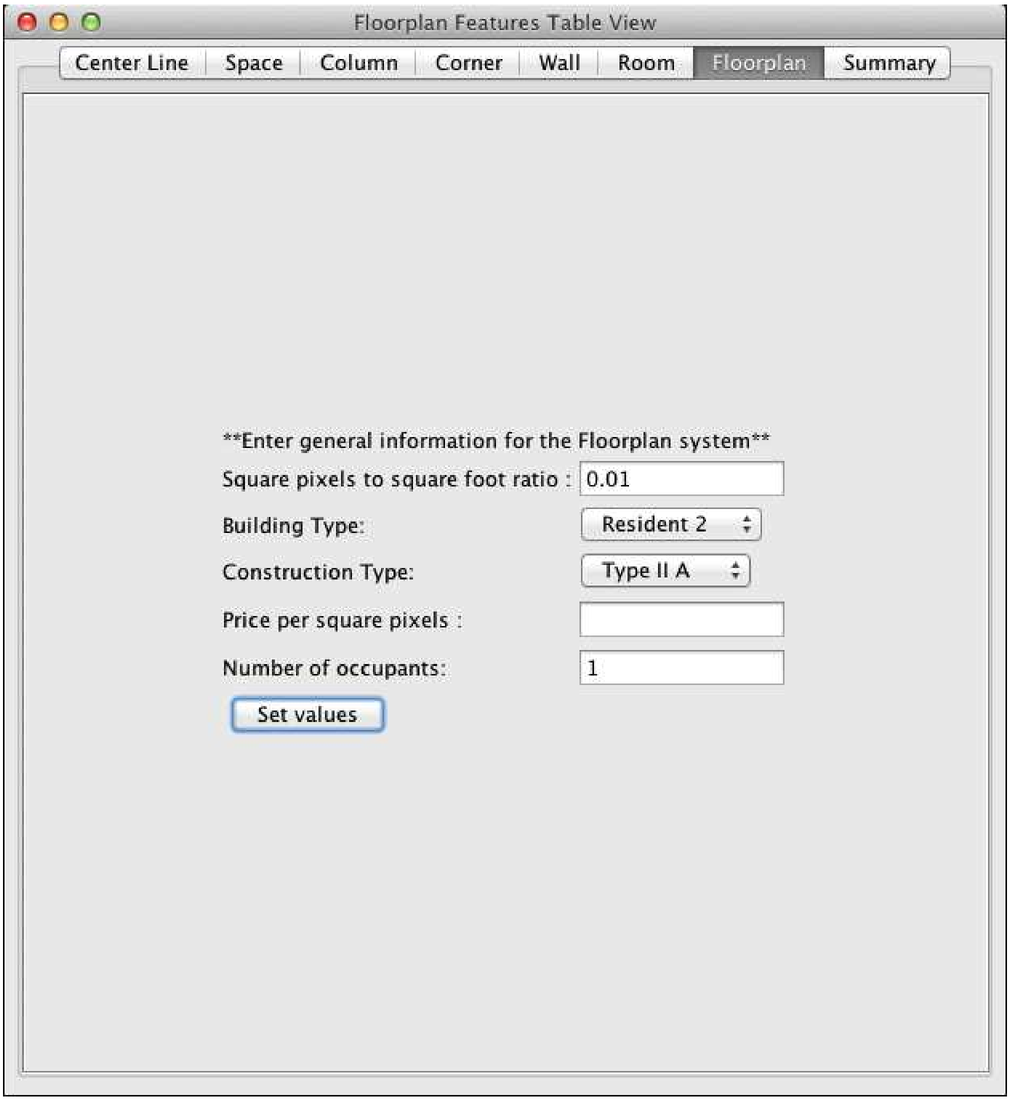
Fields	**Data**
Sq pixels/Sq foot	0.01
Building Type	Resident 2
Construction Type	Type II A
Price per Sq pixels	
Number of occupants	1

Table 4.1: Floor plan model general settings for simple room example.

Step 8. Click on the Center Line tab and select c-1 center line and modify the Shift value to the desired value to redesign the floor plan system. See Figure 4.14.

Step 9. Click on the Summary tab to check the output data for the simple room example floor plan model. See Figure 4.15.

Step 10. To redesign the floorplan system, repeat Step 8 with different desired values for center line location. The data can be gathered and plotted as below.



Enter general information for the Floorplan system

Square pixels to square foot ratio : 0.01

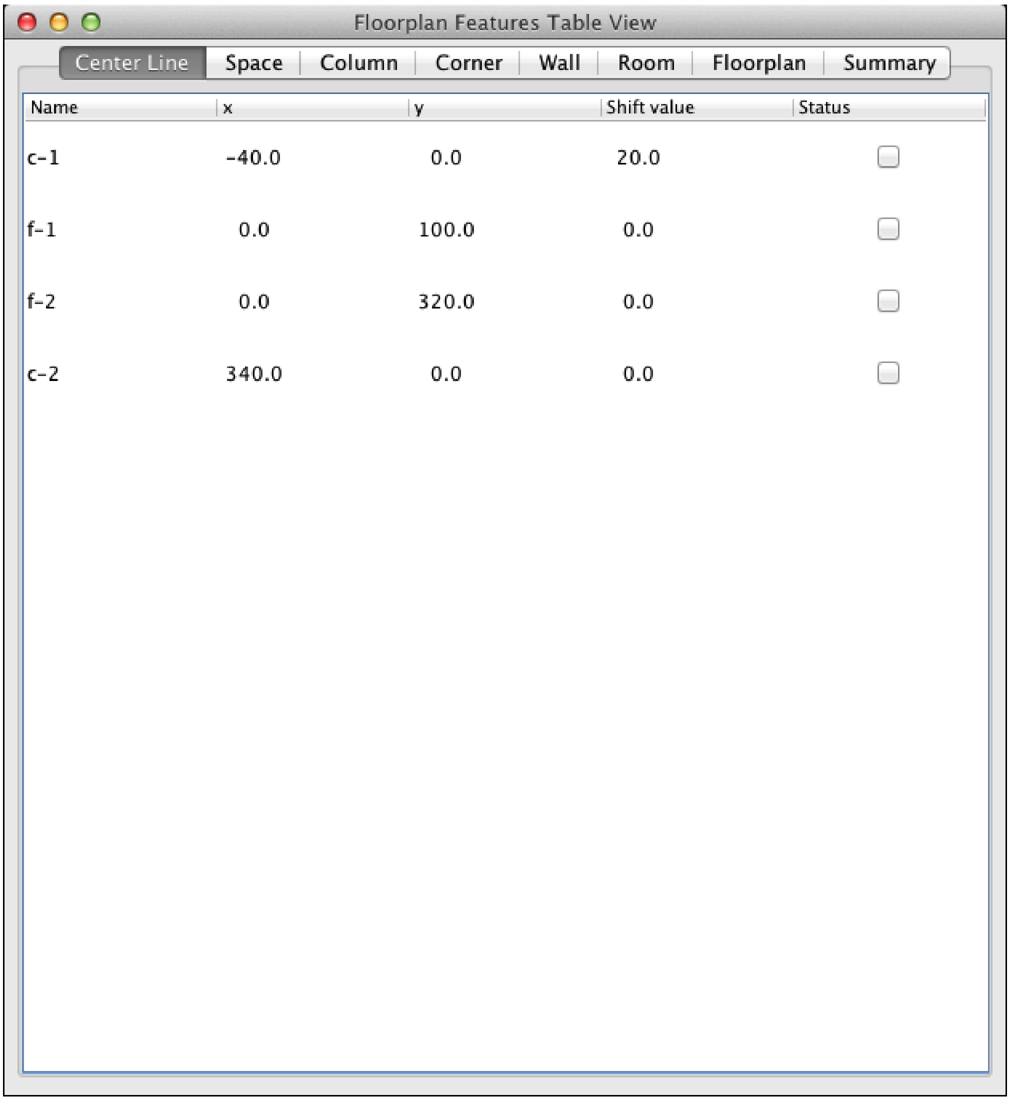
Building Type: Resident 2

Construction Type: Type II A

Price per square pixels :

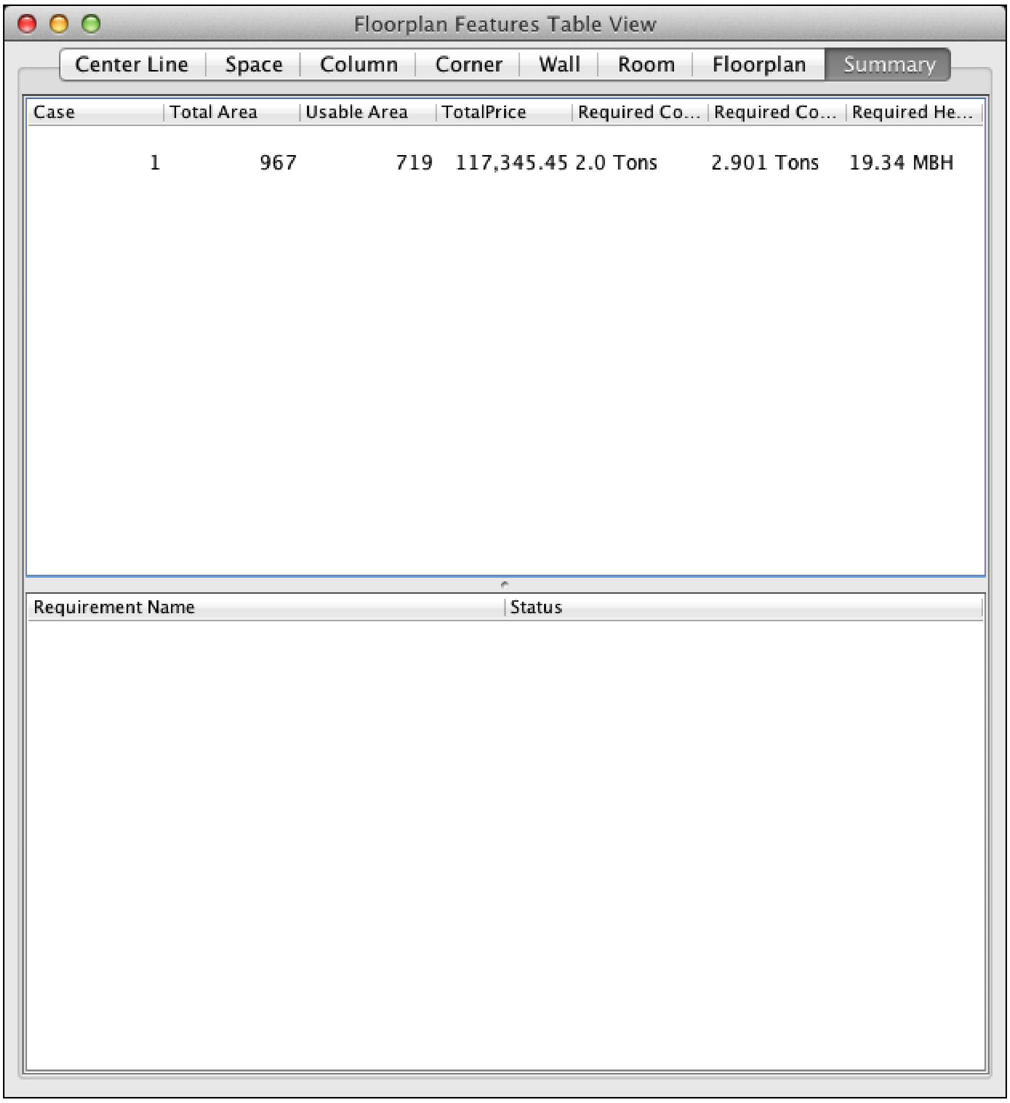
Number of occupants: 1

Figure 4.13: Floorplan tab inside the table view of the software.



Name	x	y	Shift value	Status
c-1	-40.0	0.0	20.0	□
f-1	0.0	100.0	0.0	□
f-2	0.0	320.0	0.0	□
c-2	340.0	0.0	0.0	□

Figure 4.14: Centerline tab inside the table view of the software.



Case	Total Area	Usable Area	TotalPrice	Required Co...	Required Co...	Required He...
1	967	719	117,345.45	2.0 Tons	2.901 Tons	19.34 MBH

Requirement Name	Status

Figure 4.15: Summary tab inside the table view of the software.

4.4 Example 2. Graphical Specification for a Simple House

The purpose of this example is to highlight the advantages of the graphical approach to floorplan specification. As with the last section example for simple room floor plan, we can extend the model to be a more complex floor plan with multiple rooms. The step-by-step procedure below demonstrates how a more complex floor plan model get created.

Step 1. Sets up the editing environment inside the editor view with the configuration below. This step allows the use to have a more friendly editing environment.

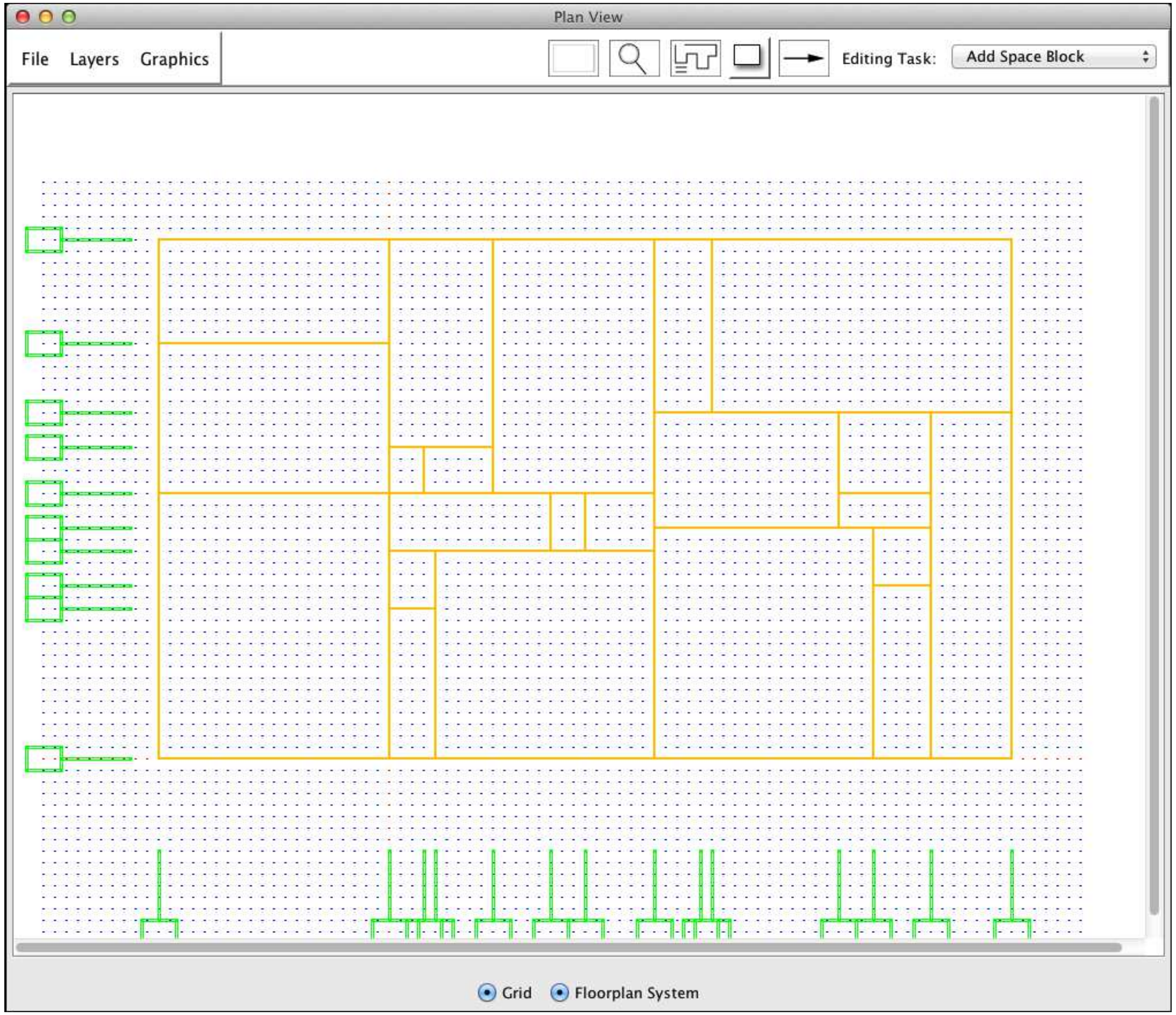
Fields	**Data**
Grid layer	Active
Grid size	10
Snap to grid	Active

Table 4.2: Environment configuration for a simple house example.

Step 2. Creates space models for the simple house example. The floor plan of simple house example is a combination of rectangular spaces as shown in Figure 4.16.

Step 3. Creates the corner points for the simple house model. The results are shown on Figure 4.17.

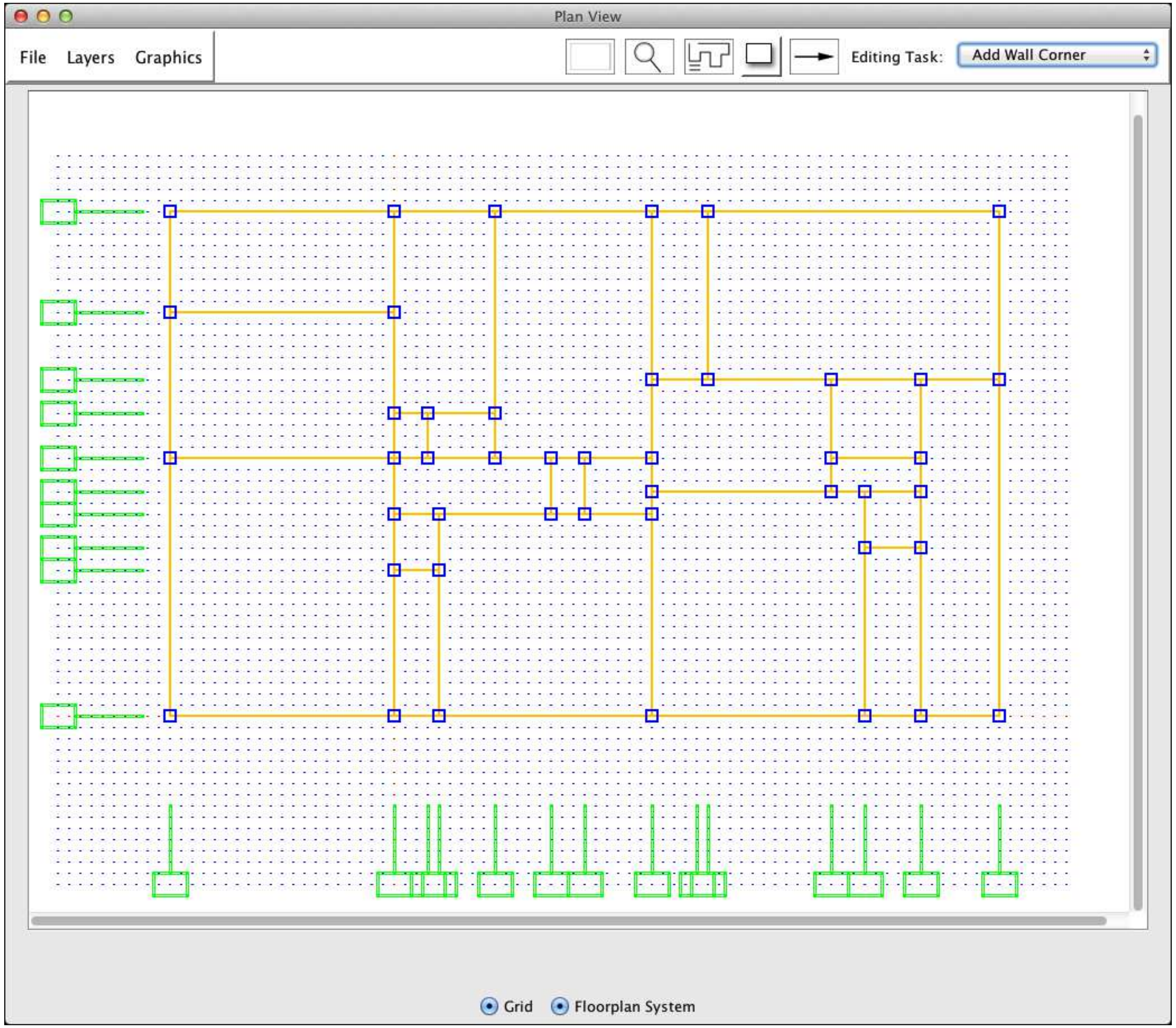
Step 4. Create walls to complete the layout of the simple room model. Since there are a lot of open spaces in this simple house example, not all edges of the space needs to create a wall in between the two corner points. The results are shown in



Editing Task: Add Space Block

● Grid ● Floorplan System

Figure 4.16: Simple house spaces floor plan layout.



Editing Task: Add Wall Corner

◉ Grid ◉ Floorplan System

Figure 4.17: Simple house model with corners.

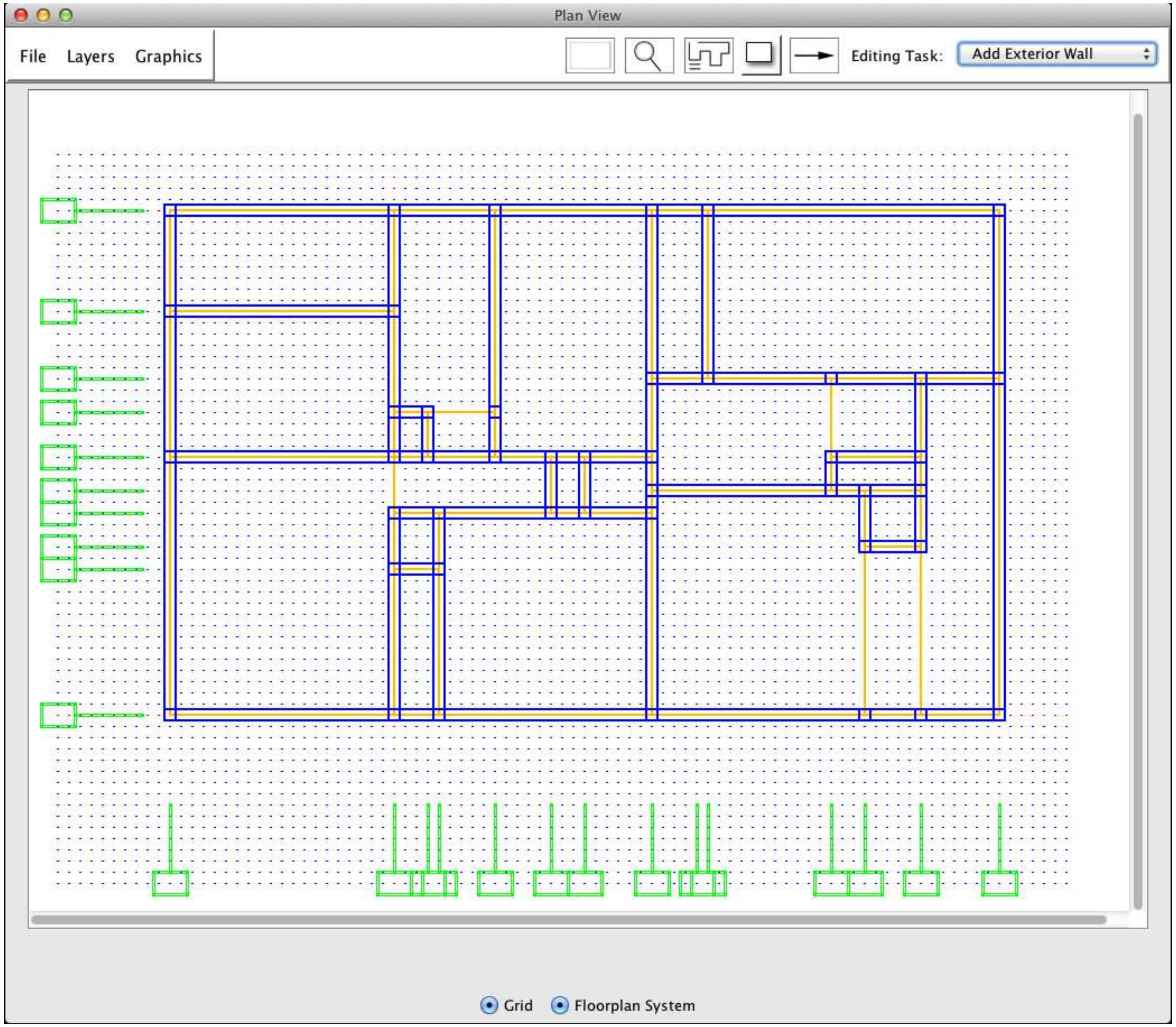
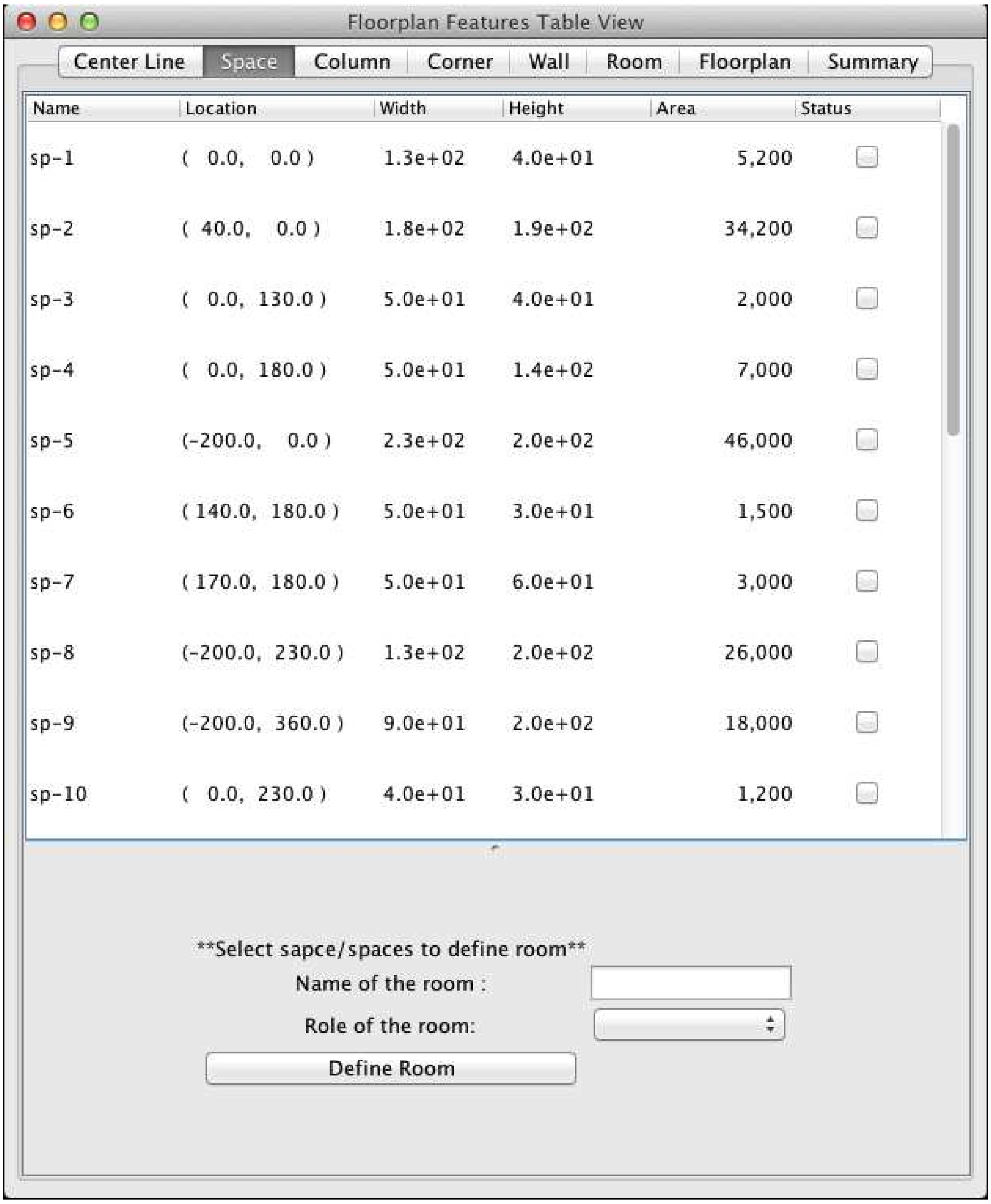


Figure 4.18: Simple house model with walls.

Figure 4.18.

Step 5. Defines the room inside the table view space tab by selecting the space that created in step 1 and enter the attributes of the room. Table 4.19 contains a summary of the spaces inside this floor plan model.



Name	Location	Width	Height	Area	Status
sp-1	(0.0, 0.0)	1.3e+02	4.0e+01	5,200	☐
sp-2	(40.0, 0.0)	1.8e+02	1.9e+02	34,200	☐
sp-3	(0.0, 130.0)	5.0e+01	4.0e+01	2,000	☐
sp-4	(0.0, 180.0)	5.0e+01	1.4e+02	7,000	☐
sp-5	(-200.0, 0.0)	2.3e+02	2.0e+02	46,000	☐
sp-6	(140.0, 180.0)	5.0e+01	3.0e+01	1,500	☐
sp-7	(170.0, 180.0)	5.0e+01	6.0e+01	3,000	☐
sp-8	(-200.0, 230.0)	1.3e+02	2.0e+02	26,000	☐
sp-9	(-200.0, 360.0)	9.0e+01	2.0e+02	18,000	☐
sp-10	(0.0, 230.0)	4.0e+01	3.0e+01	1,200	☐

Select sapce/spaces to define room

Name of the room :

Role of the room:

Figure 4.19: Simple house model spaces summary inside table view space tab.

Step 6. Checks the summary of the whole floor plan model.

4.5 Assessment of Approach 2

With the two examples above, we can have a better understanding of how the graphical specification software is capable of. Compared with the script modeling base approach, the graphical specification approach not only boosting up the efficiency of creating a floor plan model, but also giving the user a better understanding of what are the components that's inside the floor plan model.

Chapter 5

Building Floorplan Case Studies

This chapter presents a full system analysis of the "two apartment units" floorplan model introduced in Chapters 3 and 4. Simplified system analyses are provided for building code requirements verification, heat pump energy consumption trade off, and life-cycle present value cost trade off of the building floorplan system. We also investigate the sensitivity of "HVAC component selection" to the nature of HVAC energy zones and perturbations in floorplan area.

5.1 Objectives and Scope

The design objective of our case study is to find a building design that is: (1) sized to fulfill the needs of the user/occupants, and (2) supported by a HVAC system that takes into account initial and lifecycle operational costs. We also investigate the sensitivity of "HVAC component selection" to the nature of HVAC energy zones and floorplan area. Without the computational framework from Chapter 4 in place, these studies would not be possible.

The scope of investigation is summarized in Figures 5.1 and 5.2. Figure 5.1 defines the two separate zones (i.e., Zone 1 and Zone 2) that will be used in the HVAC system selection study. The two green rectangles define zones that will each

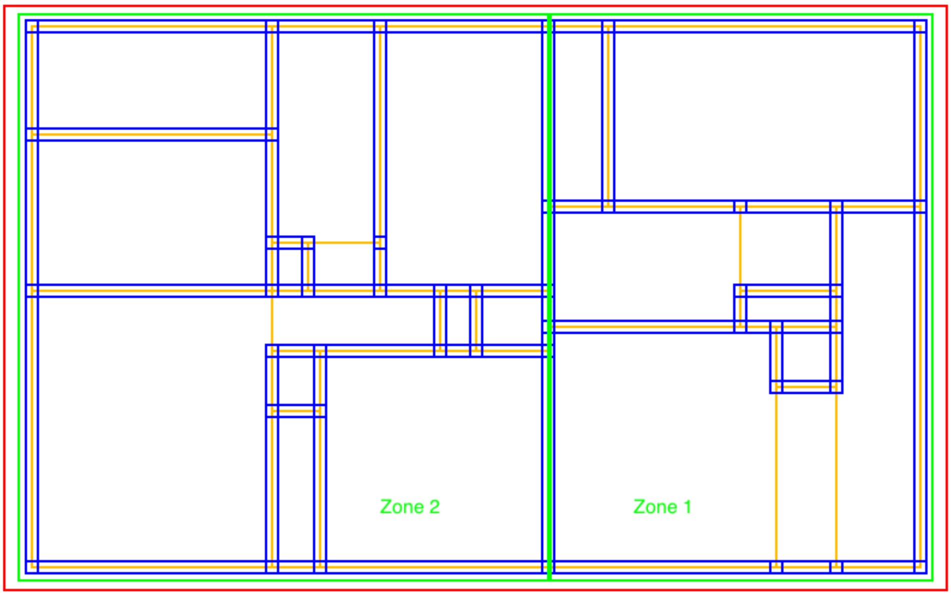


Figure 5.1: The HVAC system cover zone inside the two-apartments floorplan model. Zones 1 and 2 are shown in green. The red rectangle shows the case where apartments 1 and 2 are bundled into a single HVAC zone.

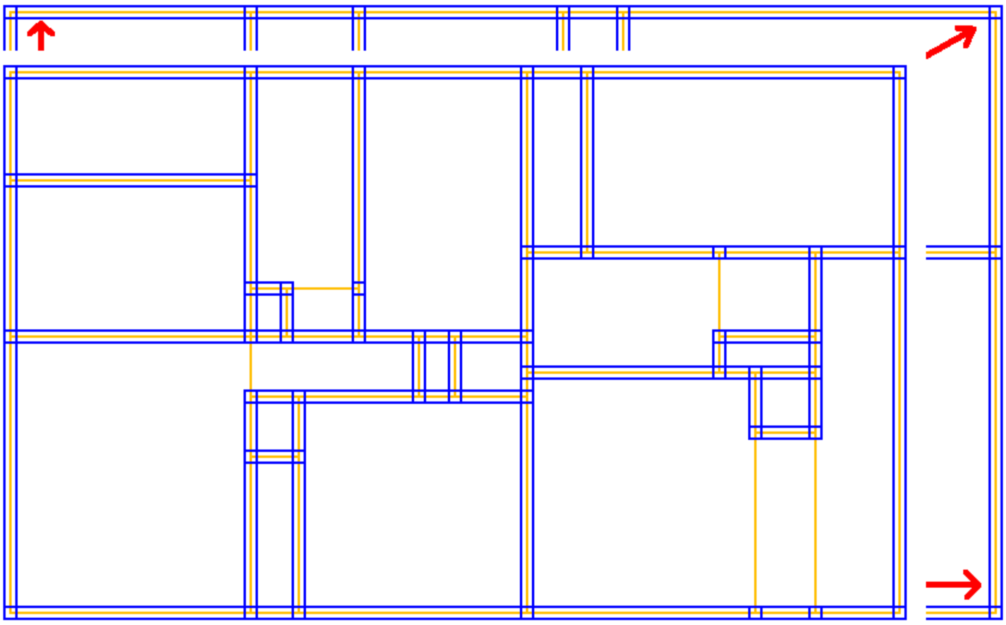


Figure 5.2: Plan view of the original floorplan design and the redesigned floorplan. The smaller floorplan is the original design.

have their own individual heat pump unit; the red rectangle defines the case study where a single heat pump unit covers the whole floorplan – in other words, two apartments share the resources of a single heat pump.

Figure 5.2 shows the difference between the original and redesigned floorplan models. The former will be studied in Section 5.4.1. The latter will be discussed in Section 5.4.2.

5.2 Building Code Requirements

The International Building Code (IBC) [1] prescribes basic requirements for securing public safety, health, and general welfare of residents for both new and existing buildings. Systems model analysis in this research project corresponds to satisfaction of a few basic area constraints for different types of rooms, occupancy groups (e.g., business, education, residential), and types of construction (e.g., masonry, wood). The IBC requirements that have been implemented in this study are as follows:

- **Section 1208.1 Minimum Room Width.** This section prescribes minimum allowable width of a room used as a habitable space or kitchen. Specifically, either side of a habitable space shall be not less than seven feet wide; kitchens shall have a clear passageway of at least three feet wide between counter fronts to walls. Since there is no clear specification in IBC for the required width of a counter front, we require it to be at least two and a half feet.

- **Section 1208.3 Room Area.** This section prescribes minimum room areas for different floorplan. Specifically, dwellings must have at least one habitable room which shall be not less than one hundred and twenty square feet. All other habitable rooms shall have a net floor area greater than seventy square feet.

- **Section 1208.4 Efficiency Dwelling Unit.** This section prescribes four requirements for a dwelling unit to be efficient. The first requirement regulates the size of the living room. If the occupancy of the unit exceeds two, then the living room should have an extra hundred squared feet of space per additional occupant. Other requirements explain what an efficient dwelling unit shall provide, and is beyond the scope of this case study.

This research also employed regulations prescribed in the International Property Maintenance Code (IPMC) [2]. IPMC's primary purpose is to regulate the minimum maintenance requirements for existing buildings. Listed below are detail explanations of all the requirements that are implemented in this study to prevent occupancy overcrowding within the building environment:

- **Section 404.4.1 Room Area.** In IBC, the minimum size of a room area does not depend on the anticipated room occupancy. Section 404.1.1 of IPMC overcomes this weakness by extending the regulation of Section 1208.3 in IBC to avoid overcrowding in living environments. They specifically require the minimum area of the bedroom shall be increased by fifty square feet for each extra occupant.

- **Section 404.5 Overcrowding - Living room.** The living room area regulation (IBC Section 1208.4) is for efficiency dwelling unit. Although IPMC Section 404.5 is less restrictive than IBC Section 1208.4, it could be used as a minimum size requirement for living rooms, and also, serve to prevent overcrowding in living environments. IPMC Section 404.5 specifically requires that the living room size be no less than a hundred and twenty square feet for one through five occupants, and a hundred and fifty square feet for six or more occupants.

With the selected IBC and IPMC requirements input into the interactive graphical-based modeling software will directly output the results of building code requirements satisfaction to the user as one of the way to analyze the building floorplan system model.

5.3 Formulation of Energy Problem

According to a residential energy consumption survey by U.S Energy Information Administration, newer U.S. homes are 30% larger but consume about as much energy as older homes due to better equipment and building isolation [26]. However, there is also a 56% increase in energy for air conditioning. This observation highlights the importance of choosing the right heating and cooling system for the building system.

Nowadays, the air-source heat pump system is a great choice for providing efficient heating and cooling for residential building and can be mainly used in nearly all parts of the United States [28]. The U.S. Environmental Protection Agency also administers the ENERGY STAR for home program to encourage people using a more efficient cooling equipment. With the sizing guidelines [37] from ENERGY STAR program, resident in different areas of the United States can easily find the applicable size of the heat pump that can be used for their home. In this case, more energy utility cost can be saved throughout the heat pump life cycle.

Cooling and Heating Usage in the US. Based on the residential air-source heat pump energy and cost saving calculator provided by the Federal Energy Management Program or ENERGY STAR, we can get cooling and heating usage for different areas in United States and the energy consumption and air-source heat pump life-cycle energy cost. The relevant formula are as follows:

$$LCC = Cost_i + E * Cost_e * \frac{(1+d)^t - 1}{d(1+d)^t} \tag{5.1}$$

$$E = C * (\frac{SummerUsage_{city}}{SEER} + \frac{WinterUsage_{city}}{HSPF}) * 0.001_{(W/kW)} \tag{5.2}$$

$$C = S * 12000_{Btu/Tons} \tag{5.3}$$

In equations 5.1 through 5.3, LCC is the life-cycle cost in USD of an air-source heat pump, $Cost_i$ is the initial cost in USD of purchasing an air-source heat pump, E is the annual energy power consumption (kWh), $Cost_e$ is the electricity cost (kWh/USD), d is the annual discount rate, t is the lifetime of an air-source heat pump (year), C is the air-source heat pump capacity (Btu), S is the air-source heat

City	**Sizing Group**	**Cooling Usage**	**Heating Usage**	**Electricity Cost (USD/kWh)**
Seattle, WA	9	282	2956	\$0.0877
Los Angeles, CA	11	1630	1070	\$0.1622
Washington, DC	25	1320	2061	\$0.1284
Miami, FL	41	3931	265	\$0.1198
Dallas, TX	65	1926	1343	\$0.1179

Table 5.1: City selection and basic information. The cooling usage and the heating usage are in hr, and the electricity cost is USD per kWh.

pump size (Tons), $SummerUsage_{city}$ is the summer usage in hr of a city, $SEER$ is the seasonal energy efficiency ratio (Btu/hr)/(Watts/hr), $WinterUsage_{city}$ is the winter usage in hr of a city, and $HSPF$ is the heating seasonal performance factor (Btu/hr)/(Watts/hr).

Cooling/Heating Usage Data for Various US Cities. Table 5.1 represents the selection of cities in United States with their sizing group from ENERGY STAR sizing guidelines and related cooling and heating usage. The selection of cities is based on the sizing group to make the selection more diverse, and the electricity cost data are based on Electric Power Monthly by U.S. Energy Information Administration[27]. The ENERGY STAR sizing guidelines consist of a set of nine maps covering the continental U.S; maps are divided into counties. Contiguous counties that have the same sizing recommendations are combined into sizing groups.

5.3.1 Electricity Cost Study

With the energy cost formula in place (see equations 5.1 through 5.3 and Table 5.1), one can tell the electricity cost is one of the main factors of energy consumption cost. Unfortunately, decision making is complicated by variations in electricity price throughout the United States. In an ideal world the cost of electricity would be zero and the decision problem for heat pump selection would boil down to choosing the heat pump that has the minimum initial cost. However, we are not living in an ideal world, and cause-and-effect relationships are no longer straightforward. Figure 5.3 graphs energy consumption cost and electricity cost across a family of US cities and specifically shows how electricity cost matters in the selected cities with the comparison of a less efficient (SEER 13) 3 tons heat pump and a high efficient (SEER 16) one. We also present in Table 5.2 the threshold electricity price difference for each city when initial price difference of the SEER 16 model can be compensated in the first year.

In Electric Power Monthly by U.S. Energy Information Administration, the cheapest electricity cost around the country is in Washington region with a cost of \$0.0877 per kWh, and the most expensive electricity cost in continental U.S. is New York region with the cost of \$0.1946 per kWh. However, Hawaii and Alaska have a more expensive electricity cost at \$0.1947 per kWh and \$0.3506 per kWh outside continental U.S. region.

While one might be tempted to purchase the least expensive heat pump

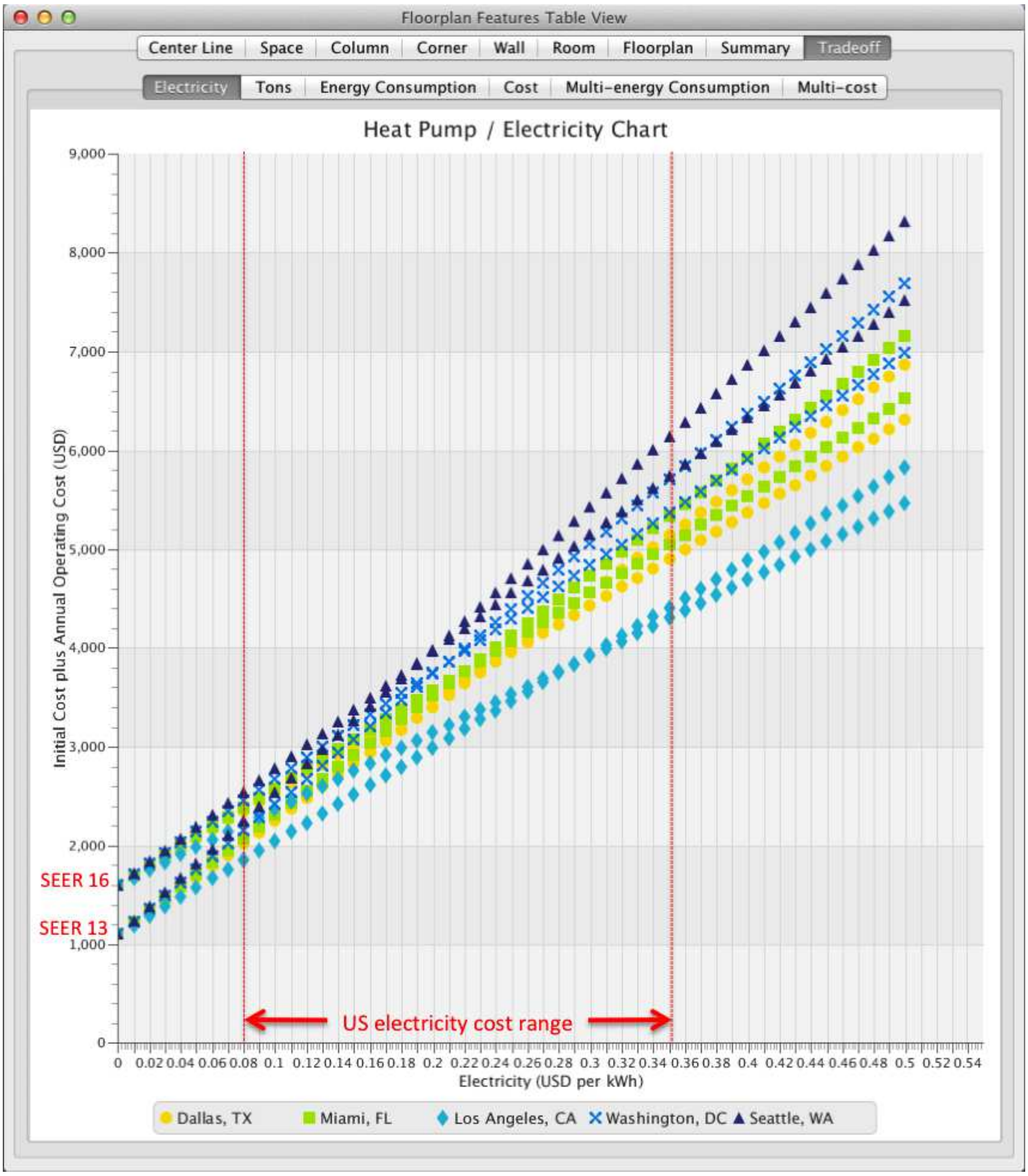


Figure 5.3: First year energy consumption costs vs electricity price with a 3 tons SEER 13 and a 3 tons SEER 16 heat pump. Plots are presented for five US cities (Dallas, TX; Miami, FL; Los Angeles, CA; Washington DC; Seattle WA).

City	SEER 13 Consumption	SEER 16 Consumption	Electricity Cost Threshold
Seattle, WA	14424.00	11836.18	$0.1932
Los Angeles, CA	9452.31	7722.24	$0.2890
Washington, DC	13167.69	10780.11	$0.2094
Miami, FL	12108.92	9848.96	$0.2212
Dallas, TX	11532.00	9422.76	$0.2371

Table 5.2: Annually energy consumption in kWh for both 3 tons SEER 13 and SEER 16 heat-pumps and the related electricity cost per kWh threshold.

(and less energy efficient), from a lifecycle perspective this is a bad strategy – in fact, as the price of electricity increases, it make makes more sense to pay a little bit more up-front and purchase the high-energy efficient unit.

5.3.2 Air-Source Heat Pump Component Library

The heat pump library (see Table 5.3) contains a set of heat pumps that are used inside simplified life cycle energy consumption and cost trade off analyses. The selection of heat pumps for the library is based on the most common heat pump sizes, ranging from 1.5 tons to 5 tons with SEER ratings from 13 to 16. Notice that all of the heat pump sizes that have non-integer values either have a SEER rating of 13 or 14. There are no non-integer heat pump sizes having a SEER rating of 16. Similarly, the library contains no commercial heat pump sizes having 4.5 tons since none could be found. As such, if the usable area of the floorplan model requires a 4.5 tons heat pump, a 5 tons heat pump will be automatically assigned.

Heat pump size	**SEER**	**HSPF**	**Price (US $)**
1.5 Tons	13	7.8	$900
1.5 Tons	14	8.5	$1,100
2 Tons	13	7.8	$900
2 Tons	14	8.5	$1,000
2 Tons	16	9.5	$1,500
2.5 Tons	13	7.8	$900
2.5 Tons	14	8.5	$1,100
3 Tons	13	7.8	$1,100
3 Tons	14	8.5	$1,200
3 Tons	16	9.5	$1,600
3.5 Tons	13	7.8	$1,100
3.5 Tons	14	8.5	$1,300
4 Tons	13	7.8	$1,200
4 Tons	14	8.5	$1,400
4 Tons	16	9.5	$1,900
5 Tons	13	7.8	$1,600
5 Tons	14	8.5	$1,700
5 Tons	16	9.5	$2,200

Table 5.3: Air-source heat pump library. Here SEER is the seasonal energy efficiency ratio (Btu/hr)/(Watts/hr) and HSPF is the heating seasonal performance factor (Btu/hr)/(Watts/hr).

5.4 Building/HVAC System Assessment and Tradeoff

In this tradeoff analysis, we study the original and the modified two-apartment floorplan designs based on ENERGY STAR air-source heat pump sizing guidelines for different city. The graphical-based modeling software framework provides 5 location trade off studies for the floorplan model. Therefore, one physical modification of the floorplan model will provide a case with 5 sub-cases as the location of the model, and we can compare two different floorplan model, which is two cases in the software, and multiple locations comparison in the trade off function.

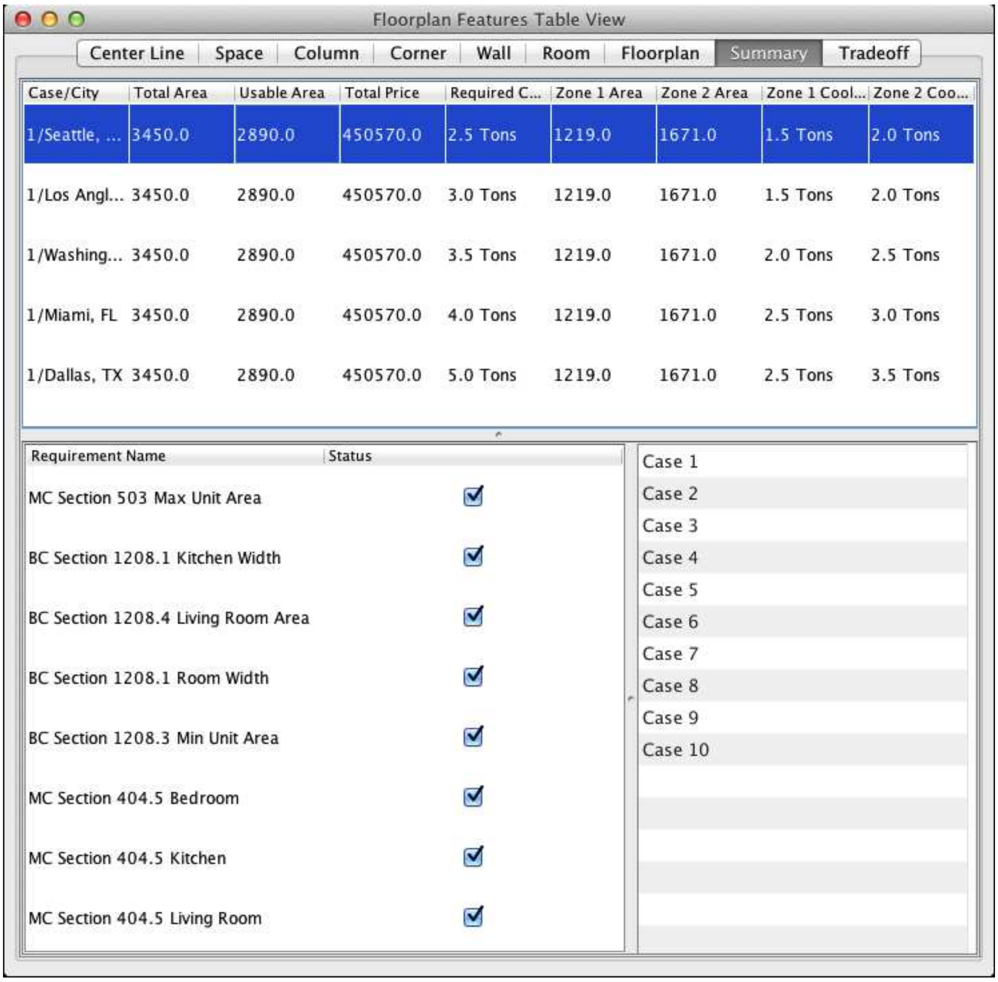
5.4.1 Original Floorplan System

First, we verified all the IPC and IPMC requirements described in Section 5.2. The results are shown in the bottom left panel of Figure 5.4. The original floorplan design model satisfies all 8 requirements from IPC and IPMC.

We assign the smaller apartment 2 as Zone 1 (Z1) and apartment 1 as Zone 2 (Z2) in the air-source heat pump sizing chart in Figure 5.5, energy consumption chart in Figure 5.6, and life-cycle cost chart in Figure 5.7.

We note that the whole floorplan model in Seattle only needs a 2.5 tons air-source heat pump to cover. For other cities, they all need more than a 0.5 tons jump from the bigger zone required capacity. However, it is still worth of consider choosing one heat pump system for the whole floorplan system.

Now, let us focus on the Los Angeles data points subset in Figure 5.6 because



Case/City	Total Area	Usable Area	Total Price	Required C...	Zone 1 Area	Zone 2 Area	Zone 1 Cool...	Zone 2 Coo...
1/Seattle, ...	3450.0	2890.0	450570.0	2.5 Tons	1219.0	1671.0	1.5 Tons	2.0 Tons
1/Los Angl...	3450.0	2890.0	450570.0	3.0 Tons	1219.0	1671.0	1.5 Tons	2.0 Tons
1/Washing...	3450.0	2890.0	450570.0	3.5 Tons	1219.0	1671.0	2.0 Tons	2.5 Tons
1/Miami, FL	3450.0	2890.0	450570.0	4.0 Tons	1219.0	1671.0	2.5 Tons	3.0 Tons
1/Dallas, TX	3450.0	2890.0	450570.0	5.0 Tons	1219.0	1671.0	2.5 Tons	3.5 Tons

Requirement Name	Status
MC Section 503 Max Unit Area	☑
BC Section 1208.1 Kitchen Width	☑
BC Section 1208.4 Living Room Area	☑
BC Section 1208.1 Room Width	☑
BC Section 1208.3 Min Unit Area	☑
MC Section 404.5 Bedroom	☑
MC Section 404.5 Kitchen	☑
MC Section 404.5 Living Room	☑

Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10

Figure 5.4: The original floorplan model requirement verification result.

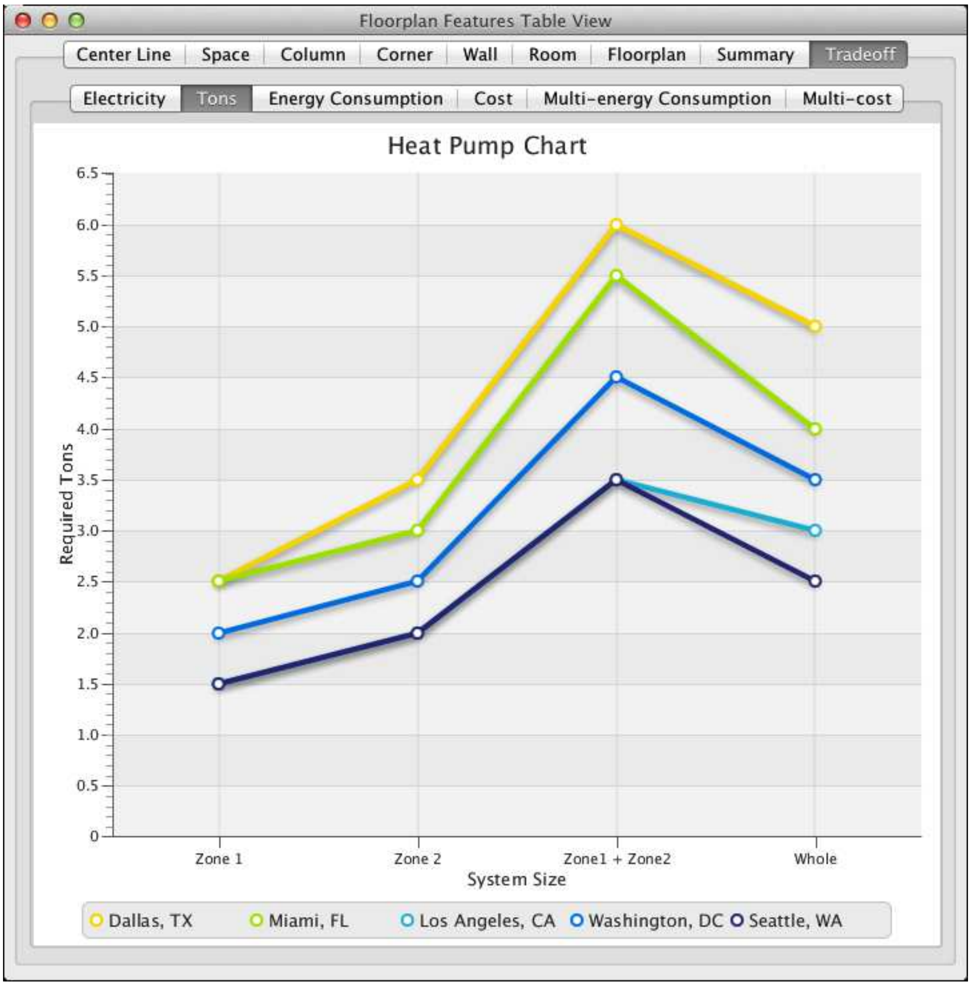


Figure 5.5: The required air-source capacities chart in tons for original design of the two apartment floorplan design by city.

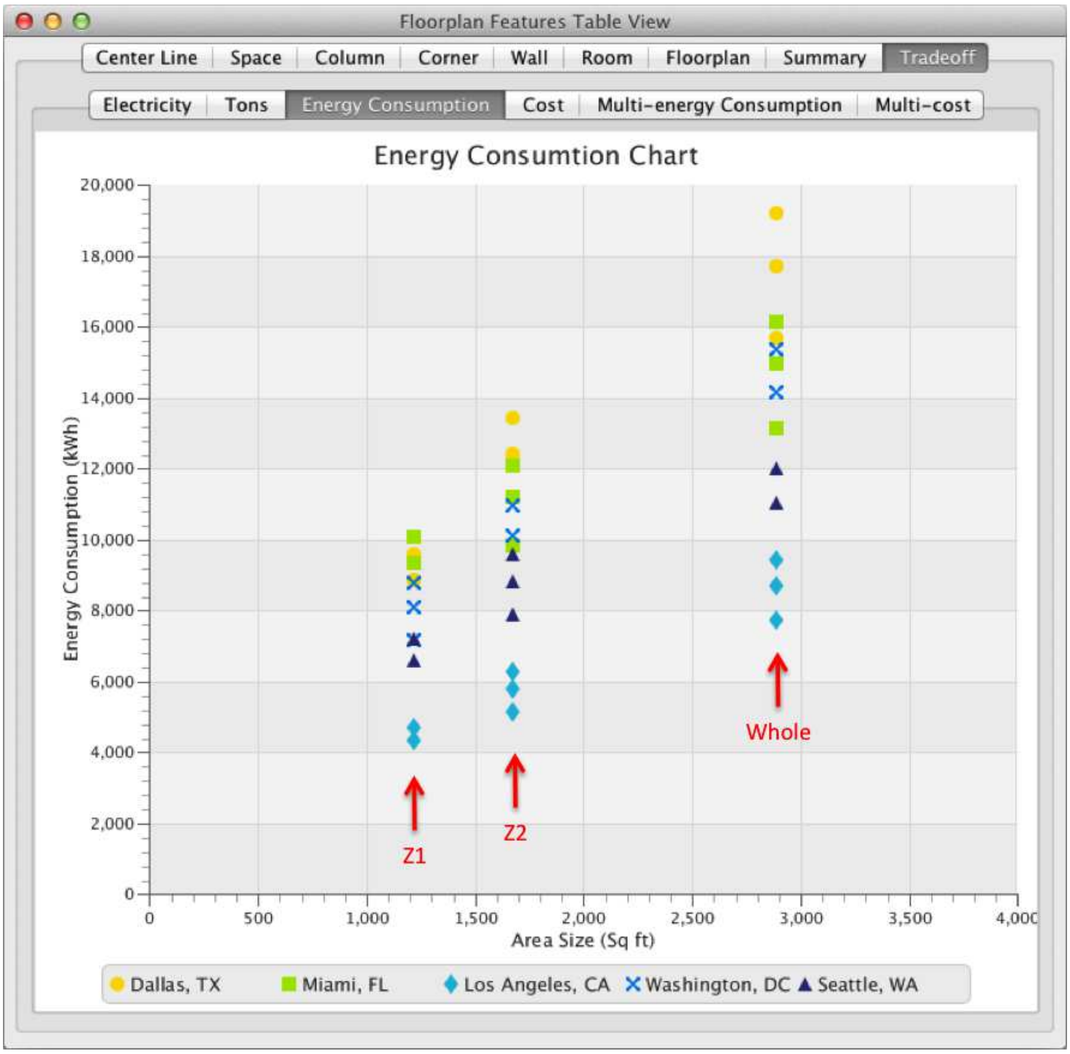


Figure 5.6: The energy consumption for the selection of air-source heat pumps chart for the original two apartment floorplan design by city. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2).

Los Angeles consumes less energy annually. The first point to note is that the difference in energy consumption for the high efficient heat pump unit and low efficient heat pump unit will increase when the size of the floorplan system increases. Moreover, if we design the floorplan HVAC system separately with zone 1 and zone 2, the combined energy consumption of the two high efficient heat pump units will be at least 10% more than the design that consider the zone as a whole (i.e., the occupants of the apartments share the resources of a single heat pump). Even if we chose the least efficient heat pump from the library in Table 5.3 for whole floorplan coverage, the annual energy consumption of the two high efficient heat pump units are still 0.6% higher. After doing the same analyses for the other cities, it is evident that Los Angeles is unusual in the sense that two zones HVAC design are only slightly less efficient (i.e., an increase of 0.6%) than a setup where a single HVAC unit is used across multiple apartments. As such, Los Angeles provides users with unusual flexibility in their selection between choosing either a one- or two-zone HVAC system design. Table 5.4 shows the comparison between the two zone case and one zone case, and the "Increase %" column represents the increased percentage between the two most efficient heat pump design combined energy consumption and the least efficient single heat pump design energy consumption.

As we mentioned in Section 5.3.1, the electricity cost is one of the other main factor for life cycle cost analysis. We simply calculated the relative life cycle cost of the selected air-source heat pumps at a average of 18.4 years lifetime [36] with the floorplan model. The results are shown as Table 5.7.

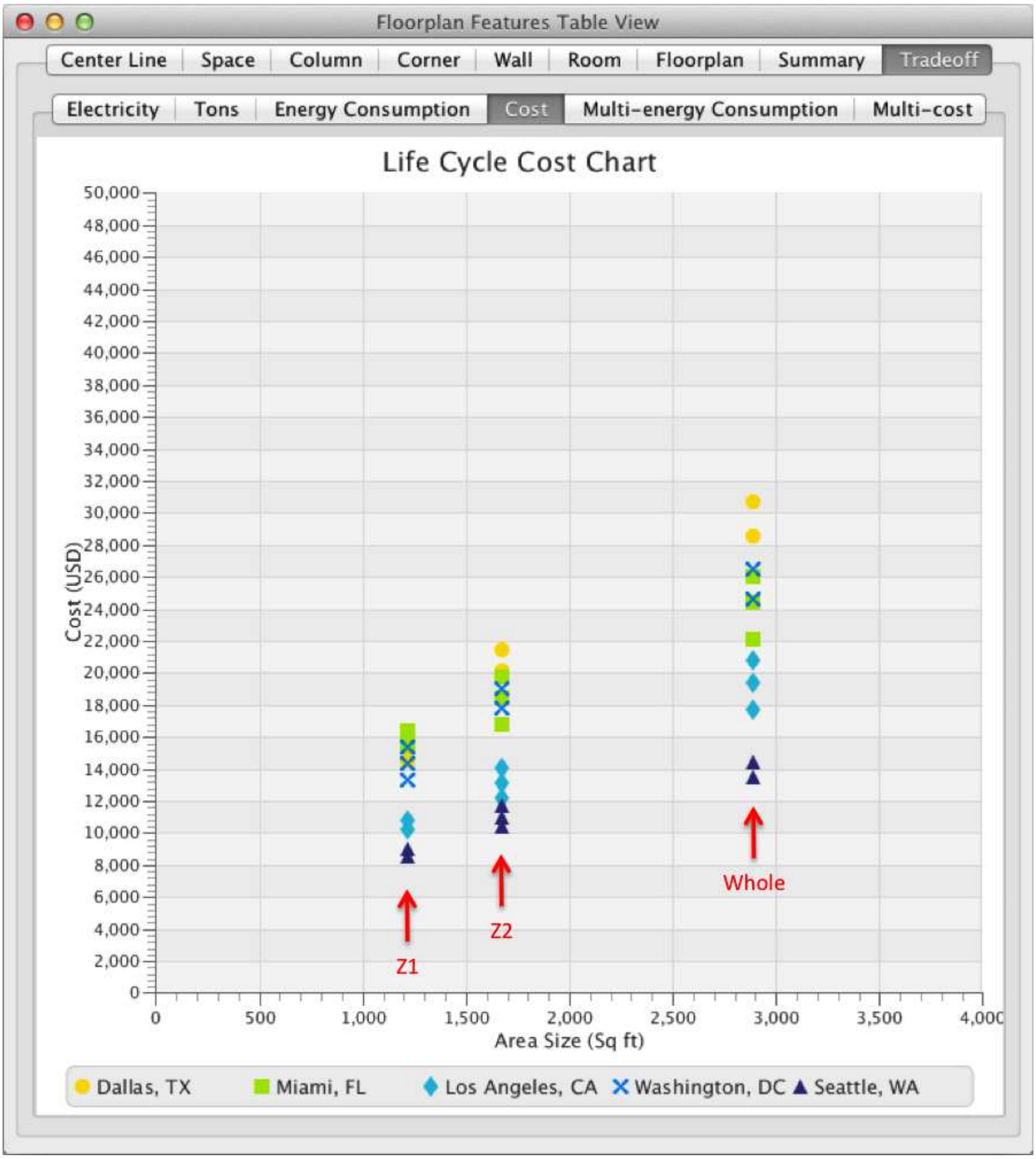


Figure 5.7: The life cycle cost analysis for different heat pump chart of the original two apartment floorplan model by city. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2).

City	**Two Zones Combined Consumption (kWh)**	**One Zone Consumption**		**Increased %**
		Most Efficient Unit(kWh)	**Least Efficient Unit(kWh)**	
Seattle, WA	6622.3 + 7890.8	11037.2	12020.0	20.7%
Los Angeles, CA	4361.6 + 5148.2	7722.2	9452.3	0.6%
Washington, DC	7186.7 + 10102.7	14143.8	15362.3	12.5%
Miami, FL	9358.9 + 9849.0	13131.9	16145.2	19%
Dallas, TX	8867.1 + 12414.0	15704.6	19220.0	10.7%

Table 5.4: Energy consumption comparison between two zone with most efficient HVAC design and one zone HVAC design.

We can calculate the life cycle cost by using equation 5.1 which is based on present value formula for annual recurring uniform amount [55] with a discount rate of 4%. As we can see the electricity cost make a huge difference between the more efficient air-source heat pump and the less efficient one, and it's also worth of notice that Seattle has a cheaper life cycle cost due to the cheaper electricity cost although the annual energy consumption is higher than Los Angeles.

5.4.2 Redesigned Floorplan System

The redesigned floorplan area is 22.7% larger than the original floorplan area. The usable area increases by 25.2%. This transformation is illustrated in Figure 5.2, and described in detail in Chapters 3 and 4. Table 5.5 provides a side-by-side comparison of building system analysis results for the redesigned and original floorplan models. Notice that the redesigned floorplan model (see Figure

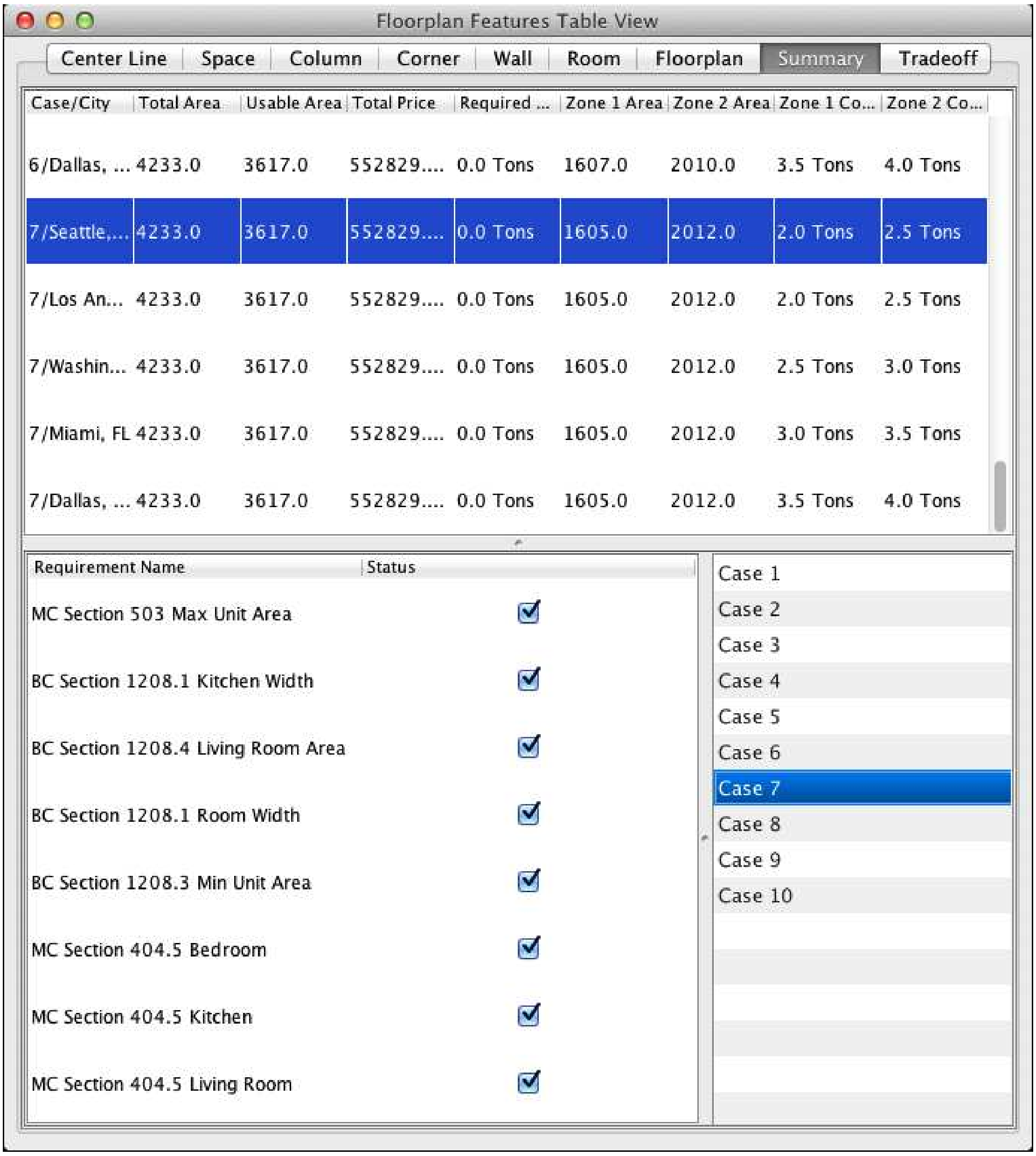
	Original Floorplan	**Redesign Floorplan**	**Increased %**
Total Area (sq ft)	3450	4233	22.7%
Usable Area (sq ft)	2890	3617	25.2%

Table 5.5: The comparison between original and redesign floorplan model area data.

5.8) satisfies all of the IBC and IPMC requirements.

With the increase of usable area for both apartment 1 and apartment 2, the needs of heating and cooling capacity clearly increased. The air-source heat pump sizing chart in Figure 5.9, energy consumption chart in Figure 5.10, and life-cycle cost chart in Figure 5.11 for redesign floorplan model will be shown below.

Moreover, since Seattle and Los Angeles required the same amount of heating and cooling capacity, the Los Angeles line is exactly covered by the Seattle line in Figure 5.9. Also observe that the 5 tons commercial air-source heat pump can't cover the whole usable area of the whole floorplan model anymore. Therefore, the one and only way to satisfy heating and cooling needs is to utilize multiple air-source heat pump units for each apartment. In other words, there is no way that a single heat pump can cover the whole floorplan model.



Case/City	Total Area	Usable Area	Total Price	Required ...	Zone 1 Area	Zone 2 Area	Zone 1 Co...	Zone 2 Co...
6/Dallas, ...	4233.0	3617.0	552829....	0.0 Tons	1607.0	2010.0	3.5 Tons	4.0 Tons
7/Seattle,...	4233.0	3617.0	552829....	0.0 Tons	1605.0	2012.0	2.0 Tons	2.5 Tons
7/Los An...	4233.0	3617.0	552829....	0.0 Tons	1605.0	2012.0	2.0 Tons	2.5 Tons
7/Washin...	4233.0	3617.0	552829....	0.0 Tons	1605.0	2012.0	2.5 Tons	3.0 Tons
7/Miami, FL	4233.0	3617.0	552829....	0.0 Tons	1605.0	2012.0	3.0 Tons	3.5 Tons
7/Dallas, ...	4233.0	3617.0	552829....	0.0 Tons	1605.0	2012.0	3.5 Tons	4.0 Tons

Requirement Name	Status
MC Section 503 Max Unit Area	☑
BC Section 1208.1 Kitchen Width	☑
BC Section 1208.4 Living Room Area	☑
BC Section 1208.1 Room Width	☑
BC Section 1208.3 Min Unit Area	☑
MC Section 404.5 Bedroom	☑
MC Section 404.5 Kitchen	☑
MC Section 404.5 Living Room	☑

Case 1
Case 2
Case 3
Case 4
Case 5
Case 6
Case 7
Case 8
Case 9
Case 10

Figure 5.8: The redesigned floorplan model requirement verification result.

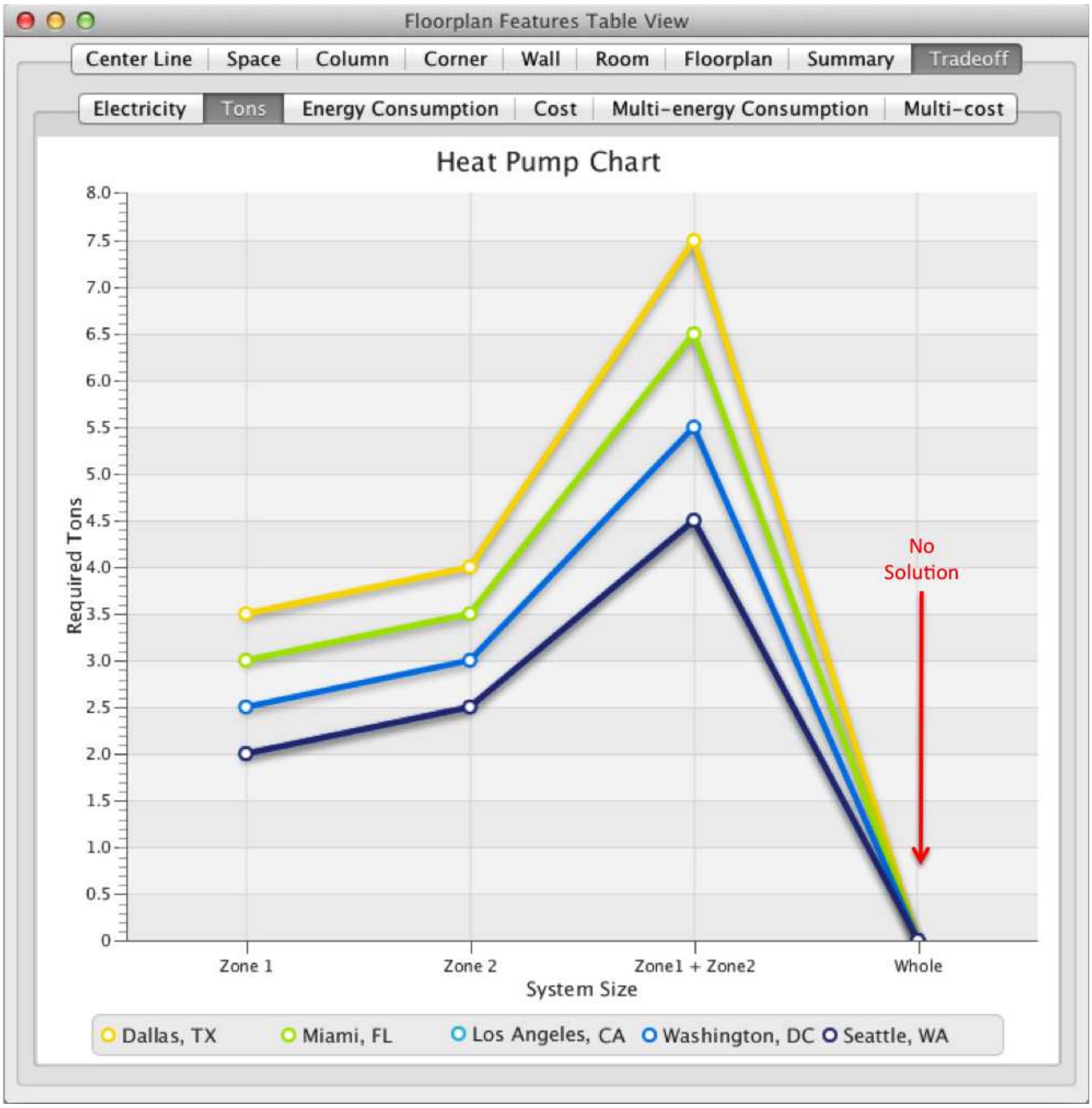


Figure 5.9: The required air-source capacities chart in tons for redesigned model of the two apartment floorplan design by city.

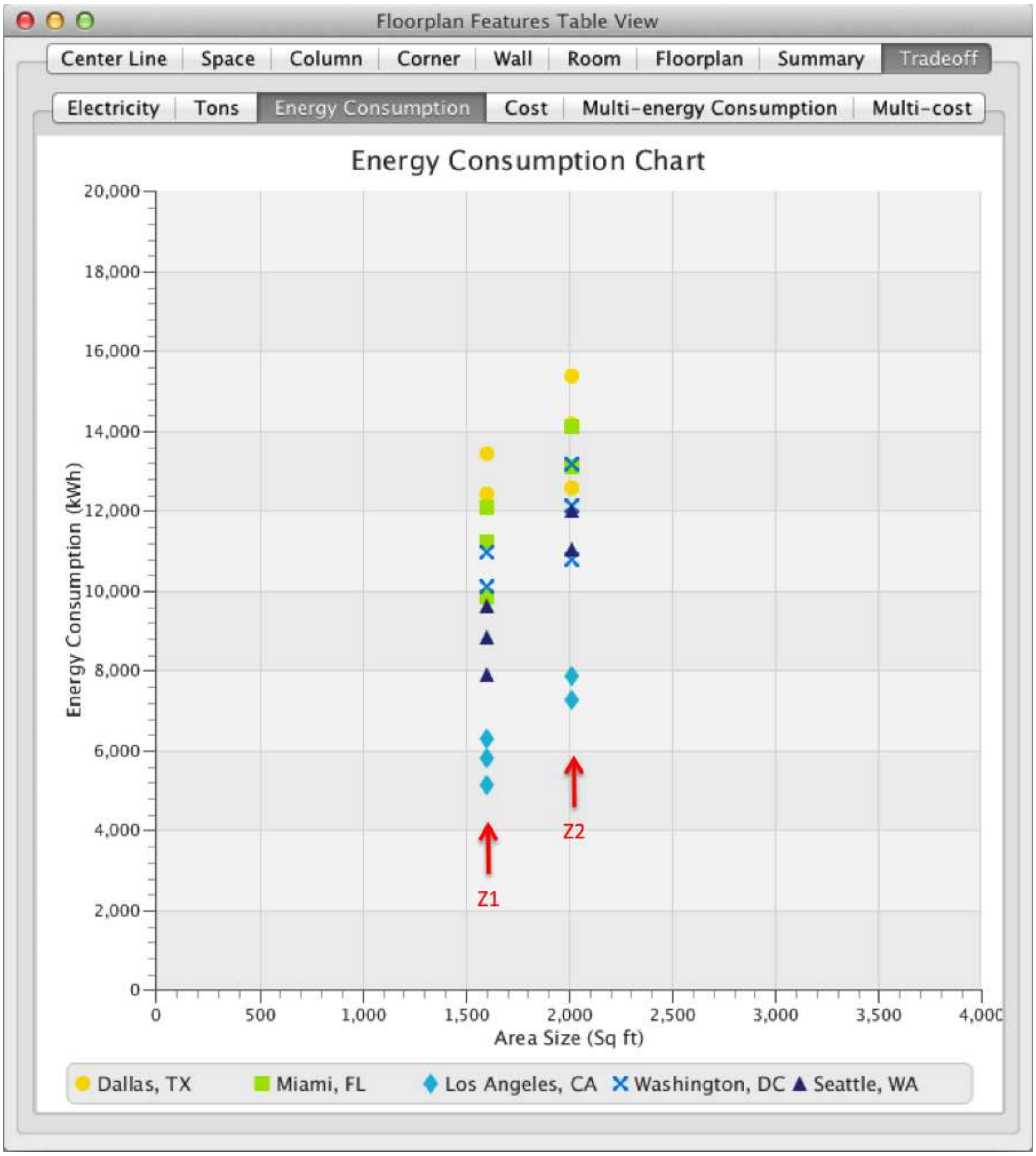


Figure 5.10: The energy consumption for the selection of air-source heat pumps chart for the redesigned two apartment floorplan model by city. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2).

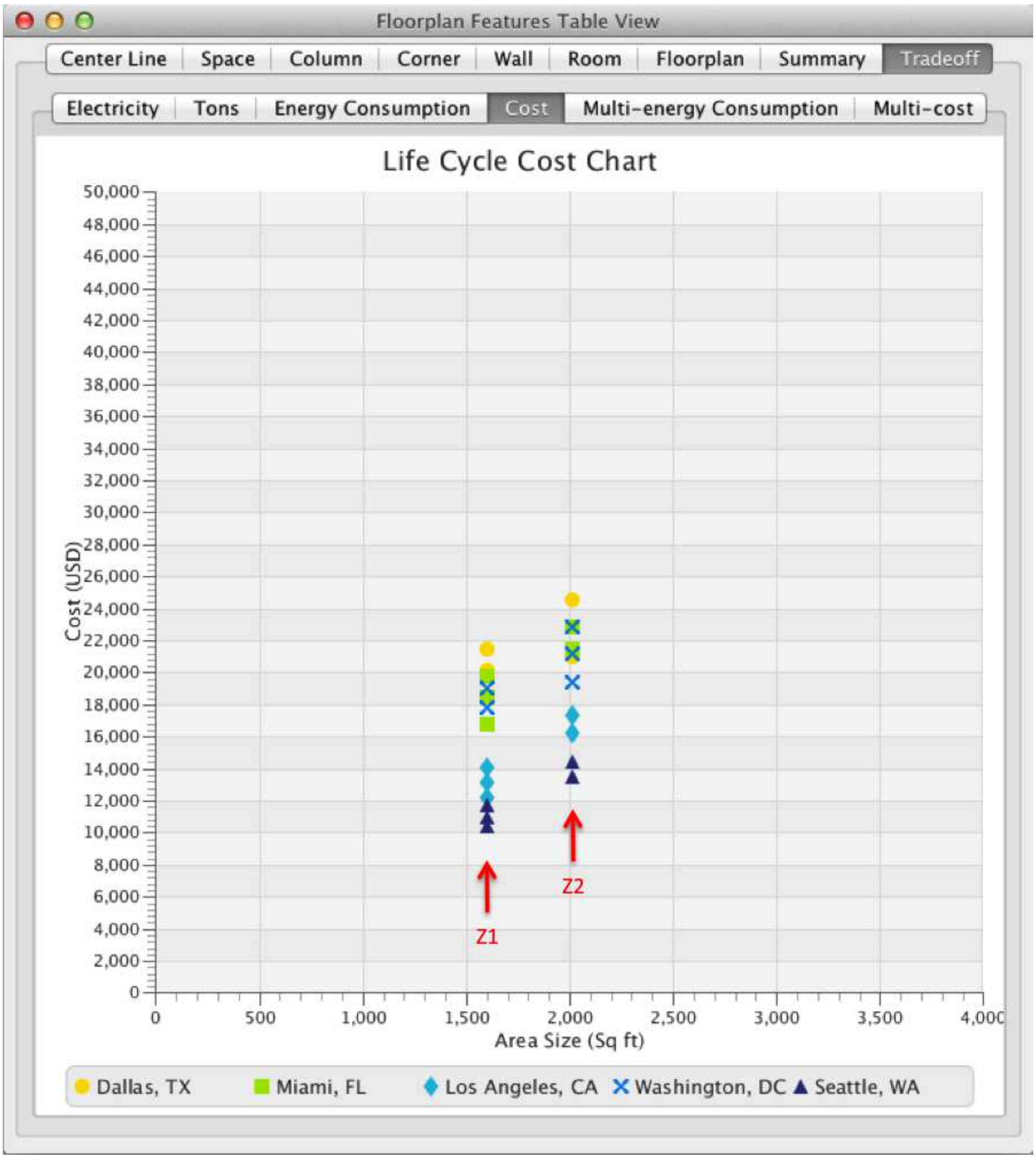


Figure 5.11: The life cycle cost analysis for different heat pump chart of the redesigned two apartment floorplan model by city. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2).

5.5 Sensitivity Analysis for Two Design Floorplan Models

Now that the multi-energy consumption (see Figure 5.12) and multi-life cycle cost charts (see Figure 5.13) are in place, we can investigate the sensitivity of HVAC system decisions to adjustments in the floorplan area design. In this specific example, the redesigned floorplan area has energy consumption and life cycle costs that are significant increases over the original design, and are beyond the capabilities of a 5 ton heat pump. As a consequence we are forced to use a two-zone setup for the redesigned floorplan model.

Figure 5.12 shows energy consumption for various air-source heat pumps used in the original and redesigned two apartment floorplan models. Figure 5.13 shows life cycle cost analysis for various heat pump in the original and redesigned two-apartment floorplan models. Generally speaking, energy consumption requirements increase with floorplan area. Table 5.6 provides a city perspective on the best designs (i.e., most efficient heat pumps) for various HVAC zone assumptions coupled with the original and redesigned floorplan models.

The main conclusions of the sensitivity analysis are as follows: while the floorplan model redesign increases the usable area by 25.2%, the consequences are most dramatic – an increase of 62.74% in energy consumption; an increase of 63.88% in life cycle cost – when the original HVAC system design was setup to cover the whole floorplan as a single zone. When a two zone setup is employed for the HVAC design in Washington, Miami and Dallas, average life cycle costs only increase by

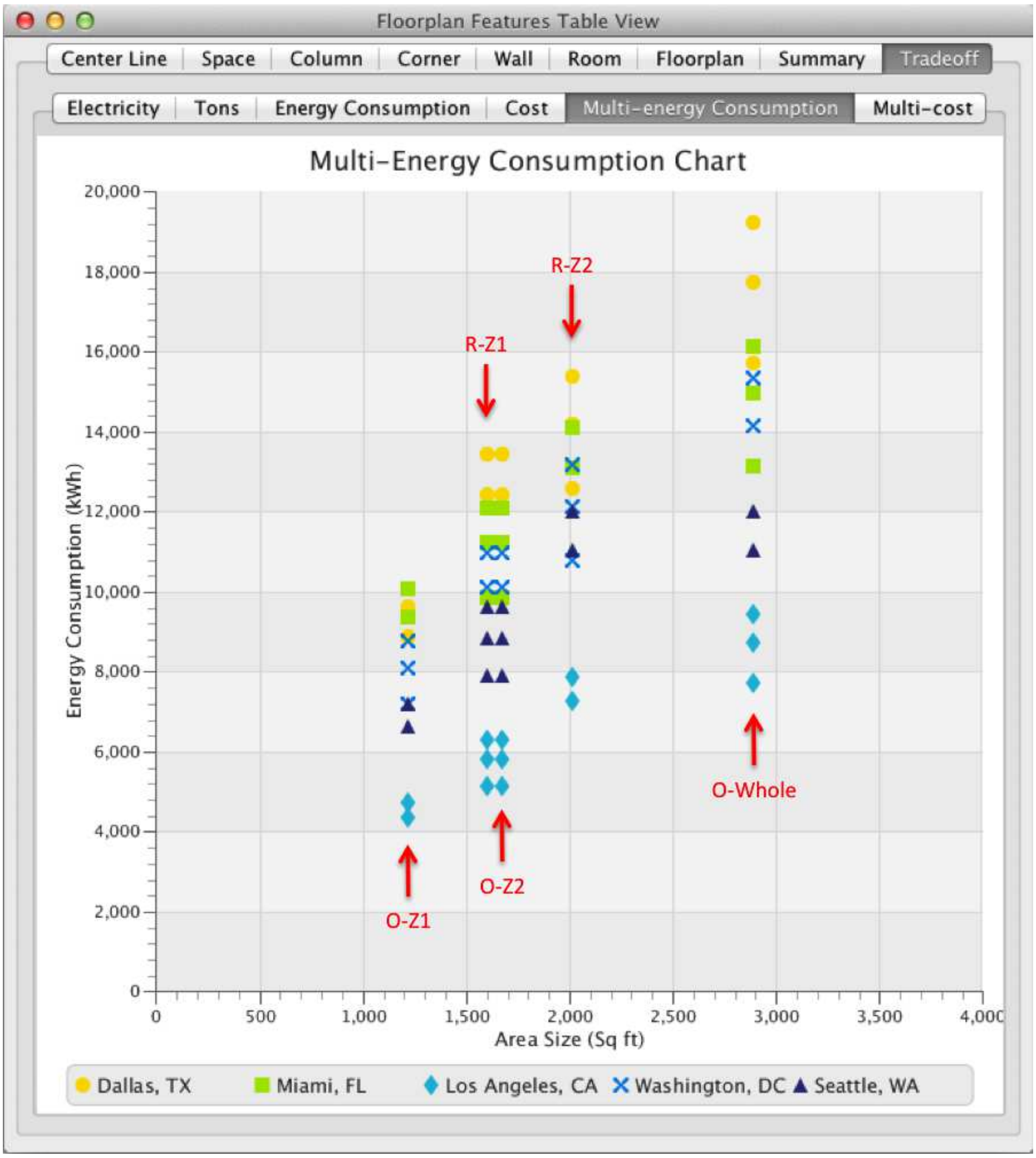


Figure 5.12: Energy consumption for various air-source heat pumps used in the original and redesigned two-apartment floorplan models. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2). The original designs for zones 1 and 2 are represented by tags O-Z1 and O-Z2, respectively. O-Whole represents the case where the entire floorplan is a single zone. R-Z1, R-Z2 are the redesign case studies.

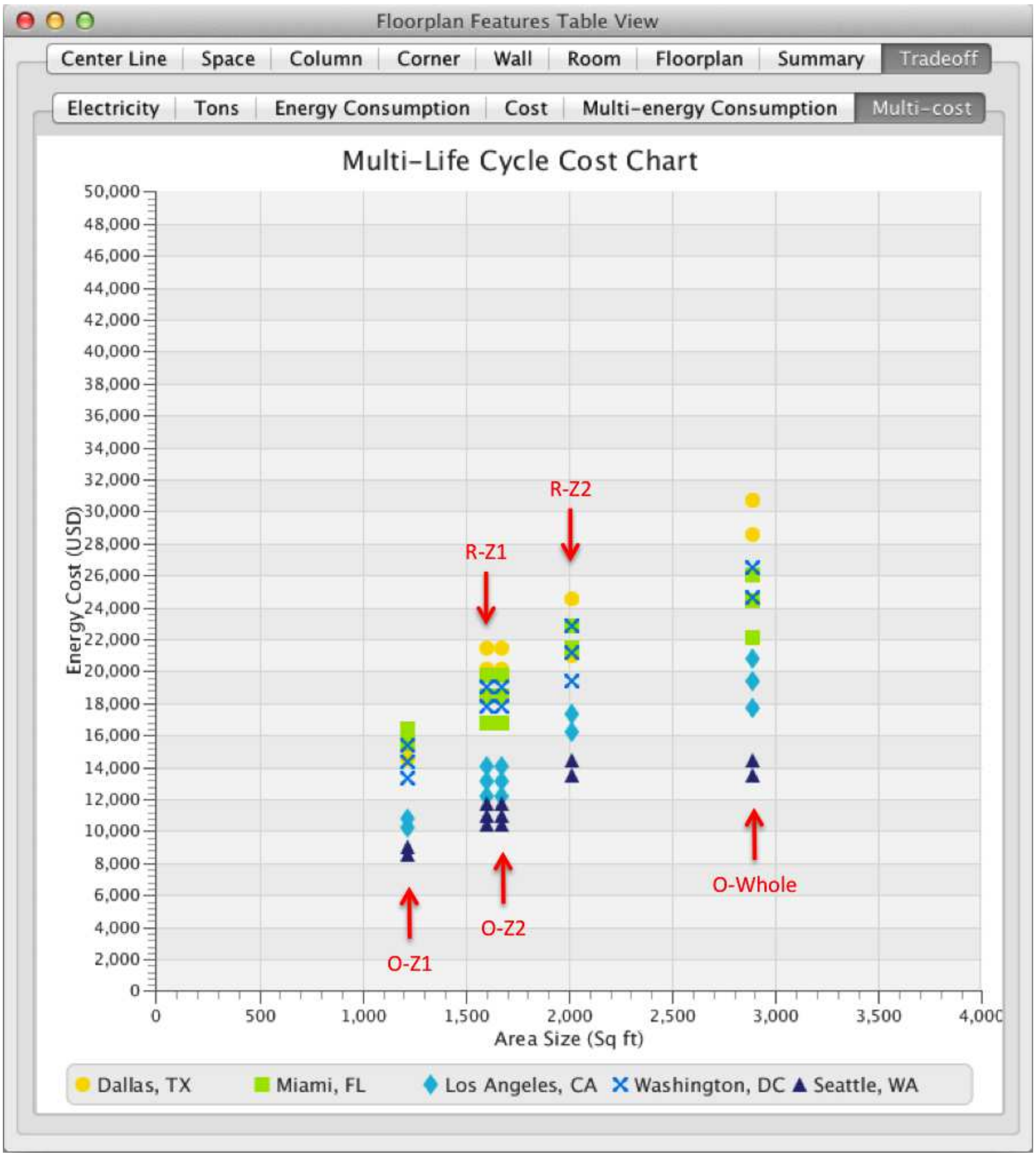


Figure 5.13: Life cycle cost analysis for various heat pump in the original and redesigned two-apartment floorplan models. Results are bundled into three groups: Zone 1 (Z1), Zone 2 (Z2) and the whole floorplan (i.e., Whole = union of zones 1 and 2). The original designs for zones 1 and 2 are represented by tags O-Z1 and O-Z2, respectively. O-Whole represents the case where the entire floorplan is a single zone. R-Z1, R-Z2 are the redesign case studies.

	Original	**Redesign**	**Increased %**
Seattle, WA			
Whole EC (kWh)	11037.2	18928.0	71.5%
Combined EC (kWh)	14513.1	18928.0	30.4%
Whole LCC (dollar)	13539.7	23933.1	76.8%
Combined LCC (dollar)	18957.2	23933.1	26.2%
Los Angeles, CA			
Whole EC (kWh)	7722.2	12417.5	60.8%
Combined EC (kWh)	9509.8	12417.5	30.6%
Whole LCC (dollar)	17697.0	28484.2	61.0%
Combined LCC (dollar)	22423.0	28484.2	27.0%
Washington, DC			
Whole EC (kWh)	14143.8	20882.8	47.6%
Combined EC (kWh)	17289.4	20882.8	30.6%
Whole LCC (dollar)	24638.9	37159.1	50.8%
Combined LCC (dollar)	31129.6	37159.1	19.4%
Miami, FL			
Whole EC (kWh)	13131.9	22951.4	74.8%
Combined EC (kWh)	19207.9	22951.4	19.5%
Whole LCC (dollar)	22117.9	38235.8	72.9%
Combined LCC (dollar)	32272.3	38235.8	18.5%
Dallas, TX			
Whole EC (kWh)	15704.6	24977.7	59.0%
Combined EC (kWh)	21281.1	24977.7	17.4%
Whole LCC (dollar)	25995.3	41045.6	57.9%
Combined LCC (dollar)	34445.9	41045.6	19.2%

Table 5.6: The comparison between original and redesign floorplan model energy consumption and life cycle cost data. Legend: EC represents energy consumption; LCC stands for life cycle cost.

19.03%. In some cities (e.g., Los Angeles) energy costs are sensitive to modifications to floorplan areas.

Chapter 6

Conclusions and Future Work

6.1 Summary and Conclusions

The long-term objectives of this research are development of model-based systems engineering (MBSE) procedures and computer-aided tools for the parametric modeling, system-level assessment, and trade-study analysis of buildings. With a focus on top-down parametric representations of two-dimensional building floorplans, coupled with support for HVAC component selection, assessment and sensitivity analysis, the work presented in the thesis is an initial step to that end. The program of research began with a scripting approach to building floorplan specification and floor area computations. While we were able to show that the approach works, unfortunately, it is also very tedious. Thousands of lines of Java source code were needed just to specify all of the details in a two-apartment floorplan system model. The second phase of the research addressed these scaleability challenges through the design and implementation of an interactive, graphically-based floorplan editor. The editor makes extensive use of software design patterns (especially composite hierarchies, model-view-controller, and visitor design patterns), and is far more efficient than the scripting approach. Use of the model-view-controller (MVC) software design pattern was particularly successful. MVC is a great way to display engineering views of floorplans alongside tree and table views, with the latter focusing on organizational perspectives and data, respectively. Computational support was developed for the evaluation of analytic functions associated with building code regulations, an electricity cost study, and simplified HVAC component selection and architecture-energy sensitivity analysis.

A two-apartment building model case study has been presented. The case study shows that it is possible to simultaneously consider parametric representations of floorplans alongside HVAC system selection. Furthermore, in decision making for the selection of HVAC system components, the price of electricity and the assignment of HVAC components to spatial regions (zones) actually matters. Solutions to this problem are not necessarily straight forward because state-of-the-art practice for HVAC zones is based on "rules of thumb" and is not yet a quantitative science. Still, over the twenty-to-fifty year (or more) working lifetime of a regular building, decisions made at the very frontend of development can have a large impact on lifecycle costs.

6.2 Future Work

The scope of work in this study has been restricted to the hierarchical specification of two-dimensional floorplans. Future work should extend these techniques to simplified representations and visualization (see Figure 6.1) for three-dimensional building geometries. Three-dimensional building architectures will be created through a top-down refinement of centerlines (for column lines, frame po-

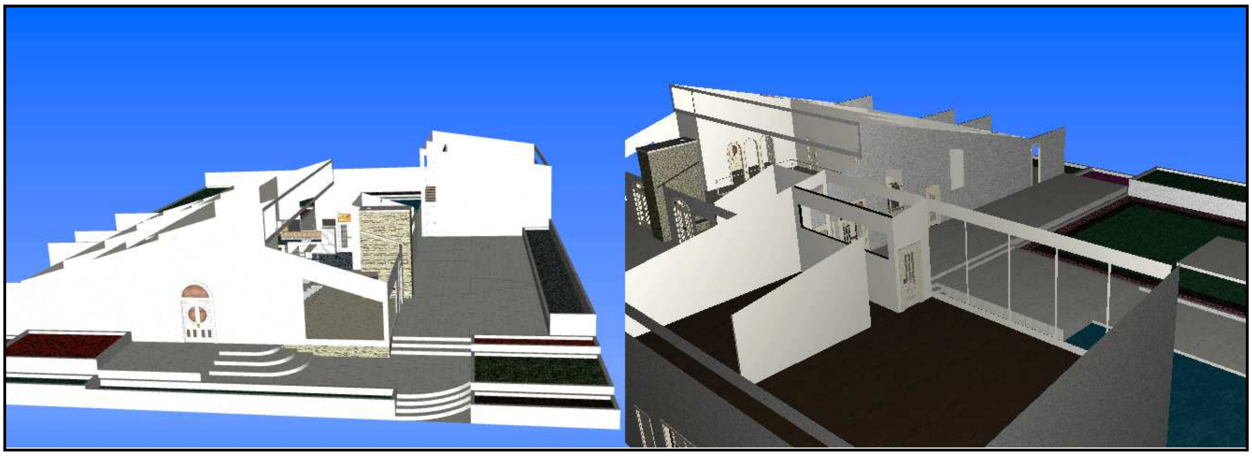


Figure 6.1: Abbreviated three-dimensional visualization of a house with JavaFX. The complete Wavefront model (obj file format) contains 781,000 vertices and faces; approximately 1/5th of the model is displayed [42, 65].

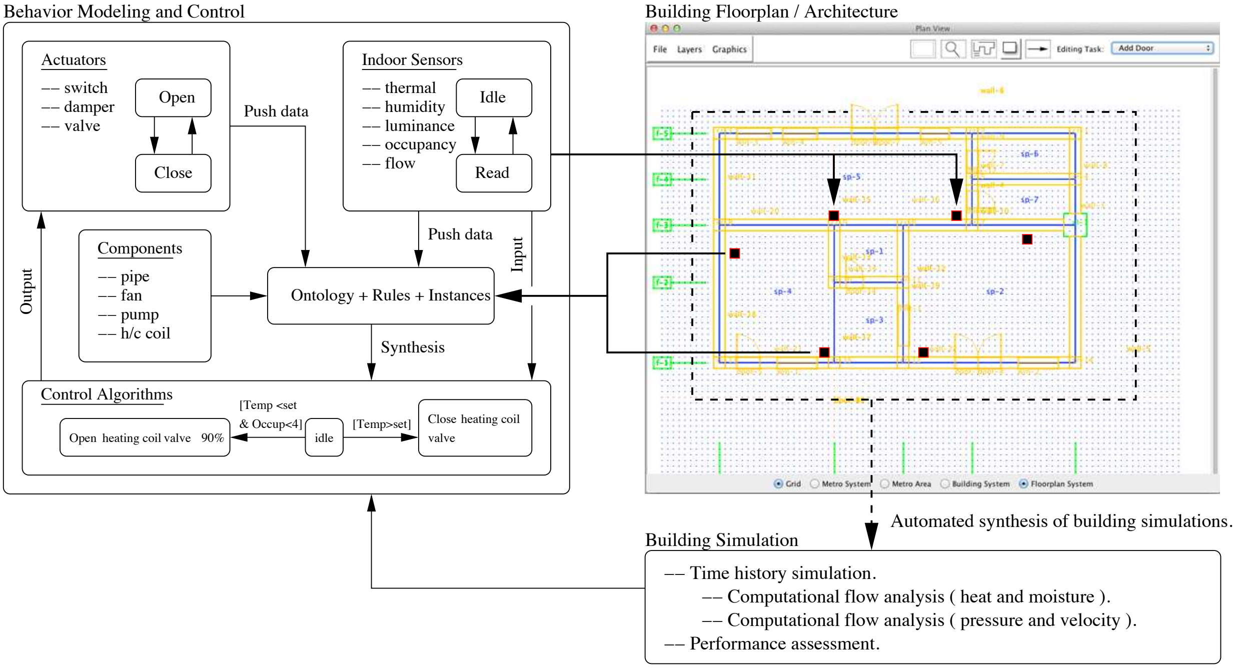


Figure 6.2: Framework for integrated development of building floorplans (and simplified three-dimensional representations of buildings), simulation and control.

sitions and floor elevations) into three-dimensional air volumes, followed by the attachment of features (e.g., walls, doors, windows) to the boundaries connecting adjacent air volumes. The coordinate properties of individual air volumes will depend on the positioning of centerlines. In state-of-the-art BIM packages (e.g., Google SketchUp), three-dimensional solid objects are modeled by just representing the edges of the solid (wire frame representation) of by modeling the surfaces of the solid. While these abstractions provide good support for visualization, they fail to provide information about the regions inside and outside the solid (for instance, whether a point is inside or outside the solid). Solid modeling techniques, on the other hand, are based on the idea that for any physical object, its boundaries or skin divide three dimensional Euclidean space in two regions. Algorithms can be devised where engineering properties (e.g., surface area; air volume) can be evaluated through a systematic traversal of the air volume nodes and edges.

This study has employed simplified HVAC system modeling assumptions and procedures. Looking to the future, even preliminary building models will need to consider combinations of discrete and continuous HVAC system behavior. Figure 6.2 shows, for example, a framework for integrated development of building floorplans (and simplified three-dimensional representations of buildings), simulation and control. Preliminary steps in the discrete modeling of dynamic behaviors with statecharts have already been taken by Delgoshaei [20, 21, 22]. A sensible way of incorporating continuous behaviors would be to return to the scripting – perhaps building geometries and properties can be specified through the use of an editor, but step-by-step solution procedures can be scripted. Finally, our procedures for parametrically adjusting the building geometry have been completely manual. Future work should consider the use of formal approaches to optimization-based design and tradeoff analysis to automate (or partially automate) this process.

Bibliography

[1] *2012 International Building Code*, chapter General Building Heights and Areas. International Code Council, Inc, 4051 West Flossmoor Road, Country Club Hills, IL 60478, 2011.

[2] *2012 International Property Maintenance Code*, chapter Light Ventilation and Occupancy Limitations. International Code Council, Inc, 4051 West Flossmoor Road, Country Club Hills, IL 60478, 2012.

[3] Aish R. and Woodbury R. Multi-level Interaction in Parametric Design. In *Smart Graphics*, pages 151–162. Springer Berlin Heidelberg, 2005.

[4] Apt K.R. *Principles of Constraint Programming.* Cambridge University Press, 2006.

[5] Austin M.A. and Wojcik C. Ontology-Enabled Traceability Mechanisms. In *Proceedings of Twentieth Annual International Symposium of The International Council on Systems Engineering (INCOSE)*, Chicago, USA, July 12-15 2010.

[6] Austin M.A., Baras J.S., and Kositsyna N.I. Combined Research and Curriculum Development in Information-Centric Systems Engineering. In *Proceedings of the Twelth Annual International Symposium of The International Council on Systems Engineering (INCOSE)*, Las Vegas, USA, July 2003.

[7] Austin M.A., Mayank V., and Shmunis N. PaladinRM: Graph-Based Visualization of Requirements Organized for Team-Based Design. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 9(2):129–145, May 2006.

[8] Barton P.K. *Building Services Integration.* E and FN Spon, London, 1983.

[9] Bentley K., 2003. Does the Building Industry Really Need to Start Over? – A Response from Bentley to AutoDesk's BIM/Revit Proposal for the Future.

[10] Bentley Generative Components, 2014. Parameteric representation for a generated compound membrane. Adapted from http://www.bentley.com/en-GB/Products/GenerativeComponents/. Accessed October 2, 2014).

[11] Bombard Electric, 2014. See http://www.bombardelectric.com).

[12] Broderbund. 3D Home Architect Design Suite. *Deluxe 6*, 2004. See http://www.broderbund.com.

[13] Building Technology Center for the Built Environment, University of California, Berkeley, CA 94720., 2009. See http://www.cbe.berkeley.edu/mixedmode/aboutmm.html and links therein (Accessed, Feb. 18, 2009).

[14] Building Technology at MIT., 2009. See http://bt.mit.edu/ and links therein (Accessed, Feb. 18, 2009).

[15] Burdett R. and Sudjic D. *The Endless City*. Phaidon Press, 2008.

[16] Chen S.Y. and Chiu M.L. Designing Smart Skins for Adaptive Environments. *Computer-Aided Design and Applications*, 4(6):751–760, 2007.

[17] Chong Y.T., Chen C.H., and Leong K.F. A Heuristic-Based Approach to Conceptual Design. *Research in Engineering Design*, 20(2):97–116, 2009.

[18] Daniels K. *Advanced Building Systems: A Technical Guide for Architects and Engineers*. Birkhauser, 2003.

[19] de Vries B., Jessurun A.J. and van Wijk J.J. Interactive 3D Modeling in the Inception Phase of Architectural Design. *The Eurographics Association*, 2001.

[20] Delgoshaei P. and Austin M.A. Software Design Patterns for Ontology-Enabled Traceability. In *Proceedings of Ninth Annual Conference on Systems Engineering Research*, Redondo Beach, CA, April 14-16 2011.

[21] Delgoshaei P. and Austin M.A. Software Patterns for Traceability of Requirements to Finite-State Machine Behavior: Application to Rail Transit Systems Design and Management. In *Proceedings of Fourtenth Annual International Symposium of The International Council on Systems Engineering*, Rome, Italy, 2012.

[22] Delgoshaei P., and Austin M.A. Software Patterns for Traceability of Requirements to Finite-State Machine Behavior: Application to Rail Transit Systems Design and Management. In *22nd Annual International Symposium of The International Council on Systems Engineering (INCOSE 2012)*, Rome, Italy, 2012.

[23] Downs L. Interchange Format for Symbolic Building Design. Dissertation submitted in partial satisfaction for the MS degree in Computer Science, University of California, Berkeley, CA, 1999.

[24] Dverk D.P. *Architecture Programming: Information Management for Design*. John-Wiley and Sons, 1993.

[25] Eastman C.M., Techolz P., Sacks R., and Liston K. *BIM Handbook: A Guide to Building Information Modeling*. John-Wiley and Sons, Hoboken, NJ, 2008.

[26] EIA: Newer US Homes are 30% Larger but Consume about as much energy as Older Homes, U.S. Energy Information Administration, February 12, 2013 (Accessed March 25, 2015).

[27] Electric Power Monthly (EPM) with Data for November 2014, U.S. Energy Information Administration, January 2015. .

[28] An air-source heat pump can provide efficient heating and cooling for your home and can be used in nearly all parts of the United States, U.S. Department of Energy, February 24, 2015 (Accessed March 25, 2015). See http://energy.gov/energysaver/articles/air-source-heat-pumps.

[29] ENSE 622: Systems Engineering Requirements, Design, and Trade-Off Analysis, 2014. Master of Science in Systems Engineering (MSSE) Program, Institute for Systems Research, University of Maryland, College Park, MD 20742. Web site: http://www.isr.umd.edu/∼austin/ense622.html.

[30] Estefan J.A. Survey of Model-Based Systems Engineering (MBSE) Methodologies, Version 8, INCOSE MBSE Initiative, 2008.

[31] 2014. European Union, EU Directive of the European Parliament and of the Council of 19 May, 2010, on the Energy Perfomance of Buildings, Official Journal of the European Union. Accessed September, 2014.

[32] Flemming U., and Chien S-F. Schematic Layout Design in SEED Environemnt. *Journal of Computing in Civil Engineering, ASCE*, 1(4):162–169, 1995.

[33] Fowler M., and Scott K. *UML Distilled Second Edition.* Addison-Wesley, Reading, Massachusetts, 2000.

[34] Geyer P. Systems Modeling for Sustainable Building Design. *Advances in Engineering Informatics*, 26:656–668, 2012.

[35] Grabska E., Lachwa A., and Slusarczyk G. New Visual Languages Supporting Design of Multi-Storey Buildings. *Advances in Engineering Informatics*, 26:681–690, 2012.

[36] Gregory R., Peter C., Alex L., James M., and Robert V.B. Consumer Life-Cycle Cost Impacts of Energy-Efficiency Standards for Residential-Type Central Air Conditioners and Heat Pumps. 2001.

[37] Manufactured Home Cooling Equipment Sizing Guidelines (for ENERGY STAR qualified manufactured homes and homes built to the HUD standards), Manufactured Housing Research Alliance, 2005.

[38] Hensen J.L.M. and Lambero R. *Building Performance Simulation for Design and Optimization*, chapter Introduction to Building Performance Simulation, pages 1–14. Spon Press (an imprint of Taylor & Francis), London and New York, 2010.

[39] Hudson R. Frameworks for Practical Parametric Design in Architecture. In *Advances in Architectural Geometry*, Vienna University of Technology, 2008.

[40] Jacobson C.A. CPS and the Energy/Environmental Systems Industry: High Performance Buildings (Energy), Systems, Cyber-Physical Systems. *A CPS Short Course for NIST Executives,* National Institute of Standards and Technology, Gaithersburg, Maryland, January 19-20 2012.

[41] Java Topology Suite (JTS). See http://www.vividsolutions.com/jts/ (Accessed October 2, 2014).

[42] JavaFX, 2015. See http://www.oracle.com/technetwork/java/javafx/overview/index.html. Accessed: Jan. 8, 2015.

[43] Johnston S. *Greener Buildings – The Environmental Impact of Property*. MacMillan Press, 1993.

[44] Kelly N.J. *Towards a Design Environment for Building-Integrated Energy Systems: The Integration of Electrical Power Flow Modeling with Building Simulation*. PhD thesis, University of Strathclyde, Glasgow, U.K., 1998.

[45] Khemlani L., Timerman A., Benne B., Kalay Y. Intelligent Representation for Computer-Aided Building Design. *Automation in Construction*, 8(1), November 1998.

[46] Korman T.M, and Tatum C.B. Prototype Tool for Mechanical, Electrical, and Plumbing Coordination. *Journal of Computing in Civil Engineering, ASCE*, 20(1), 2006.

[47] Krygiel E., Demchak G., and Dzambasova T. *Mastering Revit Architecture 2009*. Sybex, an Imprint of Wiley Publishing Inc, 2009.

[48] Madeen D.A., Palma R.M. Architectural AutoCAD: Drafting, Design and Presentation. 2001.

[49] Markus T.A., Whyman P., Morgan J., Whitton D., Maver T., Canter D. and Fleming J. *Building Performance*. John-Wiley and Sons, 1972.

[50] OpenDesign. TurboCAD Deluxe. *Version 10*, 2004. The deluxe edition handles 2d- and 3d drawings, definition of simple solids, and boolean operations. A professional edition is supported by the ACIS solid modeling package.

[51] Peng X. *Modelling of Indoor Thermal Conditions for Comfort Control in Buildings*. PhD thesis, Delft University of Technology, The Netherlands, 1996.

[52] Rosenman M.A., Gero J.S. Modeling Multiple Views of Design Objects in a Collaborative CAD Environment. *Computer-Aided Design*, 28(3):193–205, 1996.

[53] Sacks R., Eastman C.M. and Lee G. Process Improvements in Precast Concrete Construction Using Top-down Parametric 3-D Computer Modeling. *PCI Journal*, 48(3):46–55, 2003.

[54] Sequin C. and Downs L. Symbolic CAD Tools for Architecture. Technical report, University of California, Berkeley, CA, 1997.

[55] Sieglinde K. Fuller and Stephen R. Petersen. Life-Cycle Costing Manual for the Federal Energy Management Program. 1996.

[56] Stelting S. and Maassen O. *Applied Java Patterns*. SUN Microsystems Press, Prentice-Hall, 2002.

[57] Terzidis K. *Algorithmic Architecture*. Elsevier, 2006.

[58] Tsai J.J. and Gero J.S.,. A Qualitative Energy-Based Unified Representation for Buildings. *Automation in Construction*, 19:20–42, 2010.

[59] Turrin M., von Buelow P., and Stouffs R. Design Explorations of Performance Driven Geometry in Architectural Design using Parametric Modeling and Genetic Algorithms. *Advances in Engineering Informatics*, 25:656–675, 2011.

[60] Uddin M.S. *Digital Architecture*. McGraw-Hill, 1999.

[61] Underwood C.P. and Francis W.H.Y. *Modeling Methods for Energy in Buildings*. Blackwell Publishing, Oxford, England, 2004.

[62] Wagner A., Klebe M., and Parker C. Monitoring Results of a Naturally Ventilated and Passively Cooled Office Building in Frankfurt, Germany. *International Journal of Ventilation*, 6(1), 2007.

[63] Wang J. Improved Engineering Design Concept Selection using Fuzzy Sets. *International Journal of Computer Integrated Manufacturing*, 15(1):18–27, 2002.

[64] Watson A. Digital Buildings – Challenges and Opportunities. *Advances in Engineering Informatics*, 25:573–581, 2011.

[65] Wavefront File Format, 2015. See http://en.wikipedia.org/wiki/Wavefront_.obj_file. Accessed: Jan. 8, 2015.

[66] Whole Building Design Guide, A Program of the National Institute of Building Sciences, See http://www.wbdg.org (Accessed September 12, 2014).

[67] Wetter M. *Building Performance Simulation for Design and Optimization*, chapter A View on Future Building System Modeling and Simulation, pages xxx–yyy. Spon Press (an imprint of Taylor & Francis), London and New York, 2011.