ABSTRACT

Title of dissertation: SEMANTIC MODELS AND REASONING FOR BUILDING SYSTEM OPERATIONS: FOCUS ON KNOWLEDGE-BASED CONTROL AND FAULT DETECTION FOR HVAC

Parastoo Delgoshaei, Doctor of Philosophy, 2017

Dissertation directed by: Mark Austin
Associate Professor, Department of Civil and Environmental Engineering, and Institute for Systems Research

According to the U.S. Energy Information Administration (EIA), the Building Sector consumes nearly half (47.6%) of all energy produced in the United States. Seventy-five percent (74.9%) of the electricity produced in the United States is used just to operate buildings. At the same time, decision making for building operations still heavily rely on human knowledge and practical experience and may be far from optimal.

In a step toward mitigating these deficiencies, this dissertation reports on a program of research to identify opportunities for using semantic models and reasoning in building system operations. The work focuses on knowledge-based control and fault detection for heating, ventilation and air conditioning (HVAC) systems. Decision-making procedures for building system operations are complicated by the multiplicity of participating domains (e.g., architecture, equipment, sensors, occupants, weather, utilities) that need to be considered. The key opportunity of this

approach is a means to utilize semantic models for knowledge representation, integration of heterogeneous data sources, and executable processing of semantic graph models in response to external events. The results of this dissertation are condensed into three case-study applications; (1) Semantic-assisted model predictive control (MPC) for detection of occupant thermal comfort, (2) Semantic-based utility description for MPC in a chiller plant operation, and (3) Knowledge-based fault detection and diagnostics for HVAC systems.

Keywords: semantic model, reasoning, ontology, model predictive control, heating ventilation and air-conditioning.

SEMANTIC MODELS AND REASONING FOR BUILDING SYSTEM OPERATIONS: FOCUS ON KNOWLEDGE-BASED CONTROL AND FAULT DETECTION FOR HVAC

by

Parastoo Delgoshaei

Dissertation 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
Doctor of Philosophy
2017

Advisory Committee:
Associate Professor Mark A. Austin, Chair/Advisor
Assistant Professor Allison Reilly
Professor John S. Baras
Professor Jelena Srebric
Dr Amanda J. Pertzborn

To the memory of my grandmother, Tayebeh, and

my beloved parents, Simin and Rahmat

who always strived to provide the best education for their children.

Acknowledgments

My graduate school experience has been a life-changing one that I will cherish forever. I was fortunate to meet, learn, and work with many amazing people during these years, and without their help, completion of this work would have not been possible. Foremost, I express my deep sense of gratitude to my advisor and good friend, Dr. Mark Austin, for his continuous support, patience, motivation and enthusiasm. His guidance helped me throughout the time of my research and writing of this dissertation. I would like to thank the members of my Ph.D. dissertation committee: Dr. John Baras for the partial financial support and sharing his valuable insights regarding the path of the project; Dr. Amanda Pertzborn for her support and guidance on the project over the past couple of years at National Institute of Standards and Technology (NIST); Dr. Jelena Srebric for her support and feedback during the final stages of this dissertation, and Dr. Allison Reilly for providing insightful comments throughout this process.

I owe my deepest thanks to Dr. Mohammad Heidarinejad for his much valued support. He pushed me forward when I was dealing with very tough challenges in life and wanted to stand still. Completion of this work would have not been possible without his indispensable mentorship.

I would like to acknowledge the financial support of the NIST Graduate Student Measurement Science and Engineering (GMSE) Fellowship Award, which helped me focus on school without any financial concerns. I would like to specifically thank Dr. Joretta Joseph for being a great point of contact at National Physical

Science Consortium. In addition, I would like to thank the Clark School Program for Exceptional Mentoring (PEM) sponsored by the Sloan Foundation and the University of Maryland Graduate School for providing the travel grant and the opportunity to present and get feedback about my research. Many thanks to the wonderful staff at Institute for Systems Research (ISR): Ms. Kim Edwards, Ms. Alexis Jenkins, Ms. Regina King for their administrative support. I would like to acknowledge the support from the Dr. Xiao Chen and Dr. Vikas Chandan for sharing their valuable insights and work.

I wish to acknowledge the tremendous support provided by senior mentors and colleagues at NIST. Specifically, I would like to express my gratitude to Mr. Steven Bushby, Dr. Andy Persily, Dr. Daniel Veronica, Mr. Farhad Omar, and Dr. Mike McCabe for their input and guidance along this journey. Also, I thank Ms. Sandra Heckman and Mr. Patrick Chen for their administrative support.

I extend my deep gratitude to my fellow graduate students and colleagues at the Systems Engineering and Integration laboratory (SEIL). Dr. Leonard Petnga, a true friend, always provided insightful feedback and support when I needed it the most. I thank Maria Coehlo for her valuable suggestions and fruitful discussions, and Dr. Chrysa Papagianni, Eddie Tseng, and David Daily for productive discussions, which have improved the quality of this dissertation.

I am grateful for my friends: Sahar Akram, Amir Ahrari, Fatima Alimardani, Bahar Heidarzadeh, Azadeh Keshtgar, Yazdan Movahedi, Ladan Rabiee, Yasaman Samei, Niloofar Shadab, and Hamidreza Shadman, who have always been a great source of support and encouragement. Special thanks to my friend Jamie Wratten,

who always wanted me to dream big and achieve my goals. His presence at my defense session made me realize how friends are close even when they live thousands of miles of away. I, would, also like to thank Nima Zolghadr for being a very supportive friend and a "go-to-person" to reach out to when something was wrong.

Lastly, all that I am I owe to my parents, Simin and Rahmat, and my brothers, Parhum and Payam, and their devotion. I am so grateful to have them as a constant source of inspiration and guidance in my life. Profound thanks to my sister-in-law, Parastou, who always encouraged me during this journey, and my aunt Heshmat for her sincere help with my dad's recent sickness making it possible for me to focus my attention on my studies. Last, but certainly not least, I would like to thank friends and relatives in Iran and United States for their love, prayers, and support during my family's most difficult time coinciding this research effort.

Table of Contents

List of Figures ix

1 Introduction 1
1.1 Problem Statement . . . 1
1.2 Motivation . . . 2
1.3 State-of-the-Art Building Energy Control . . . 4
1.4 Research Scope and Objectives . . . 5
1.5 Research Hypothesis and Questions . . . 7
1.6 Contributions . . . 8
1.6.1 Framework for Data-Driven Semantic-based Reasoning . . . 9
1.6.2 Integration of Semantic and Physics-based Modeling . . . 11
1.6.3 Semantic Framework for Fault Detection and Diagnosis . . . 12
1.6.4 Software Design Patterns for Systems Integration . . . 13
1.7 Organization . . . 15

2 Languages and Tools for Semantic Knowledge Representation 17
2.1 The Semantic Web . . . 17
2.1.1 Semantic Web Vision . . . 17
2.1.2 Semantic Web Technical Structure . . . 18
2.2 Languages for Semantic Modeling . . . 20
2.2.1 Resource Description Framework (RDF) . . . 20
2.2.2 Web Ontology Language (OWL) . . . 22
2.3 Semantic Modeling with Ontologies and Rules . . . 26
2.3.1 Ontologies . . . 27
2.3.2 Individuals . . . 27
2.3.3 Axioms . . . 28
2.3.4 Reasoning . . . 28
2.4 Semantic Web Tools . . . 29
2.4.1 OWL Supported Tools . . . 29
2.4.2 Working with Jena and Jena Rules . . . 30
2.5 Simple Examples . . . 32
2.5.1 A Jena Rule to Reset the Room Setpoint. . . . 32

2.5.2 Simplified Modeling of Event-Driven Component Dynamics . . 32
2.5.3 Semantic Modeling of Valve Behavior 38
2.5.4 Semantic Modeling in Intelligent Buildings 40
2.6 Data-Driven Generation of Individuals in Semantic Graphs 42

3 State-of-the-Art Engineering for HVAC 43
3.1 Models of Computation for Behavior Modeling 45
3.1.1 Introduction to Models of Computation 45
3.1.2 Five Approaches to System/Model Development 47
3.1.3 Co-Simulation of Subsystem-Level Processes 49
3.2 State-of-the-Art Tools for Building Performance Simulation 50
3.2.1 Procedural-Based Tools 51
3.2.2 Equation- and Object-based Tools 52
3.2.3 Actor-based Tools 53
3.3 BCVTB Software Architecture with MoC Annotations 55
3.4 State-of-the-Art Control for HVAC 58
3.4.1 Supervisory Control Strategies 59
3.4.2 Model Predictive Control 59
3.5 State-of-the-Art Fault Detection and Diagnostics for HVAC 64
3.5.1 Automated Fault Detection and Diagnostics 64
3.5.2 Procedures for Fault Detection 65
3.5.3 Procedures for Diagnostic Analysis of Faults 65
3.6 Summary 68

4 Semantic Knowledge Modeling for Buildings 70
4.1 Introduction 70
4.2 Previous Work 72
4.3 Meta-Domain Ontologies and Rules 76
4.3.1 Temporal Ontology and Rules 76
4.3.2 Spatial Ontology and Rules 80
4.4 Engineering Ontologies and Rules 84
4.4.1 Building Ontology and Rules 84
4.4.2 Mechanical Equipment Ontology and Rules 86
4.4.3 Sensor Ontology and Rules 87
4.4.4 Fault Detection and Diagnostic Ontologies and Rules 90
4.4.5 Fault Detection and Diagnostic Procedures 90
4.5 Surrounding Environment Ontologies and Rules 92
4.5.1 Occupant Ontology and Rules 92
4.5.2 Weather Ontology and Rules 95
4.6 Economic Ontologies and Rules 97
4.6.1 Utility Ontology and Rules 97
4.7 Semantic Integration of Building Ontologies and Rules 99
4.8 Summary 101

5 Case Study Applications 102
5.1 Case Study 1: Semantically-Enabled Model Predictive Control 103
5.1.1 Problem Description 103
5.1.2 Problem Goals 103
5.1.3 Problem Setup 104
5.1.4 Results 112
5.1.5 Findings 116
5.2 Case Study 2: Knowledge-Assisted MPC for Utility Representation . 118
5.2.1 Problem Description 118
5.2.2 Problem Goals 119
5.2.3 Problem Setup 119
5.2.4 Simulation Results 126
5.2.5 Findings 129
5.3 Case Study 3: Knowledge-Based Fault Detection and Diagnostics . . 130
5.3.1 Problem Description 130
5.3.2 Snapshot of Semantic Graph Model Assembly 131
5.3.3 Test Problem Scenario and Hypothesis Evaluation Procedure . 136
5.3.4 Synthesis of Multi-domain Rules 137
5.3.5 Multi-domain Rule Evaluation 137
5.3.6 Findings 141

6 Conclusion and Future Work 143
6.1 Conclusions 143
6.2 Future Work 145

A Systems Integration and Behavioral Simulation with Whistle 147
A.1 Whistle Scripting Language Design 148
A.2 Example 1. Parsing a Simple Assignment Statement 150
A.3 Example 2: Oscillatory Flow between Two Tanks 151
A.4 Example 3: Continuous/Discrete Behavior of a Water Tank 155

Bibliography 160

List of Figures

1.1 Framework for design . . . 6
1.2 Contribution 1 . . . 10
1.3 Contribution 2: Integration of semantic models with simulation / control. 12
1.4 Integration of Semantic and Physical Models . . . 13
1.5 Collage of software design patterns. . . . 14
1.6 Collage of composite software design patterns and simple applications. 14

2.1 Technologies in the Semantic Web Layer Cake [51]. . . . 19
2.2 Example of RDF triple . . . 21
2.3 A simple RDF graph showing information about James. . . . 21
2.4 An OWL graph of relationships describing the semantics about a Human. 23
2.5 Formal definition of a Person and Properties in OWL. . . . 24
2.6 Forward chaining of facts and builtin functions to new assertions. . . . 31
2.7 Relationship between classes and properties in the component ontology. 33
2.8 Evolution of ontology graph as a function of time. . . . 34
2.9 Schematic for a valve ontology and rules. . . . 38
2.10 Schematic for a three-port valve. . . . 39
2.11 Data-driven approach to generation of individuals in semantic graphs. 42

3.1 Operating systems view of HVAC behaviors . . . 43
3.2 Integration of MPC with Modelica . . . 44
3.3 Network of models of computation . . . 47
3.4 Five approaches to system/model development . . . 48
3.5 Three strategies of data exchange in energy co-simulation . . . 50
3.6 Annotated co-simulation architecture (MPC with Modelica) . . . 56
3.7 Multi-level control structure for HVAC systems. . . . 57

4.1 Domain specific and domain independent ontologies . . . 72
4.2 Schematic of Allen's temporal intervals. [101] . . . 77
4.3 Time ontology and associated data and object properties. . . . 79
4.4 Two rules for reasoning with time. . . . 79
4.5 Spatial (geometry) ontology and associated data and object properties. 81
4.6 Rules to determine the rooms in which sensors have been placed. . . . 81

4.7 Schematic of building ontology classes and properties. 83
4.8 Rule to check if two zones intersect. 83
4.9 Schematic of equipment ontology classes and properties. 85
4.10 Rules for operation of mechanical equipment. 86
4.11 Sensor ontology classes and properties. 88
4.12 Rule to compute intersection of zones. 88
4.13 Fault detection and diagnostic ontology classes and properties. 89
4.14 Rule for detecting a faulty state. 90
4.15 Identification of faults, hypotheses and supporting evidence. 91
4.16 Schematic of occupant ontology classes and properties. 93
4.17 Rule for occupants location and thermal comfort. 94
4.18 Partial view of weather ontology classes and properties. 95
4.19 Rules to detect weather condition. 96
4.20 Schematic of utility ontology. 98
4.21 Sample Jena rules for utility ontology. 100

5.1 Plan view of large room with five thermal zones. 103
5.2 Physical simulation model and Dymola environment. 105
5.3 Flow diagram for data exchange in BCVTB 107
5.4 BCVTB framework DTS-based MPC and Modelica simulation model. 107
5.5 Setpoint, room temperature, and control signal. 109
5.6 Dynamic thermal sensation model. 109
5.7 Dynamic thermal sensation index versus time (sec) for Control Case 1.114
5.8 Dynamic thermal sensation index versus time (sec) for Control Case 2.114
5.9 Room occupancy, setpoint and temperature vs time for Case 1. 115
5.10 Room occupancy, setpoint and temperature vs. time for Case 2. . . . 115
5.11 Architecture for coupled semantic/MPC HVAC system control. 119
5.12 Multi-level control structure for HVAC systems. 120
5.13 Utility tariff ontology for Austin, Texas 122
5.14 Schematic of the thermal energy storage. 124
5.15 Simulation results for Austin, New York City and San Francisco . . . 128
5.16 Two-room building architecture, sensors, and building occupants. . . 131
5.17 Snapshot of fully assembled semantic graph model. 132
5.18 Fault detection diagnostic rules for operation of a heating coil. 134
5.19 Snapshot of multi-domain evaluation and chaining rules. 139

6.1 Hybrid behavior of a valve. 145

A.1 Base and derived units in engineering mechanics. 149
A.2 Parse tree for x = 2 in. 149
A.3 Summary of forces acting on a pipe element connecting two tanks. . . 152
A.4 Tank water levels (m) versus time (sec). 153
A.5 Volumetric flow rate (m^3/sec) versus time (sec). 153
A.6 Front elevation of tank, supply pipe, and exit pipe and valve. 156
A.7 Time-history response of tank with water supply and shut-off valve. . 157

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BAS	Building Automation System
BCVTB	Building Controls Virtual Test-Bed
BIM	Building Information Modeling
BMS	Building Management System
CCHP	Combined Cooling, Heating and Power
CPS	Cyber-Physical Systems
DAML	DARPA Agent Modeling Language
DARPA	Defense Advanced Research Projects Agency
DL	Description Logics
DTS	Dynamic Thermal Sensation
FDD	Fault Detection and Diagnostics
FOL	First Order Logic
FSM	Finite State Machine
GUI	Graphical User Interfaces
HVAC	Heating, Ventilation, and Air Conditioning
IFC	Industry Foundation Classes
IoT	Internet of Things
IMCE	Integrated Model-Centric Engineering
ISO	International Organization of Standardization
JAXB	XML Binding for Java
JSON	Javascript Object Notation
JTS	Java Topology Suite
MBSE	Model-Based Systems Engineering
MoC	Model of Computation
MVC	Model-View-Controller
NIST	National Institute of Standards and Technology
ODE	Ordinary Differential Equation
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SQL	Structured Query Language
SysML	System Modeling Language
SWRL	Semantic Web Rule Language
TES	Thermal Energy Storage
TOU	Time of Use
VAV	Variable Air Volume
WWW	World Wide Web
XML	Extensible Mark-up Language

Chapter 1: Introduction

1.1 Problem Statement

Over the past twenty years the notion of intelligent buildings (IBs) has has matured from conceptual frameworks to early phase implementations due to the potentials they offer for maximized occupants comfort with an optimized energy profile. Intelligent buildings are now envisioned as an integral part in achieving bigger goals of having smart cities in urbanized areas. Moreover, future buildings design guidelines are now heavily influenced by intelligent buildings capabilities [56]. Although some buildings are categorized as so-called intelligent, the application of intelligence in buildings is at the very best far from ideal [32].

There are various definitions for intelligent buildings Some definition of Intelligent buildings state that creates an environment which maximizes the effectiveness of the buildings occupants while at the same time enabling efficient management of resources with minimum life-time costs of hardware and facilities, some other definitions focused on the role of technology for automation and control of building functions. A comprehensive review of definitions for intelligent buildings has been done by Ghaffarianhoseini and co-investigators [56]. However, due to the emergence of Internet of Things (IoT) technologies and their impact on intelligent buildings,

new expectations are expected from the terminology. Memoori [90] predicts that the traditional building automation systems (BASs) will evolve into a BIoT over the next five years. These technologies and applications are poised to deliver increased efficiencies in all aspects of building intelligence from monitoring, analyzing and control without humans' intervention.

Modern buildings are now equipped with the state-of-the-art building automation systems making so-called best autonomous control decisions based on the acquired sensor data. The drawback of this approach is the decisions are made based on analysis of data with not based on the underlying knowledge.

This dissertation explores the extent to which emerging Semantic Web technologies can be exploited to both represent information and knowledge about the state of building and its surrounding domains. Moreover, integrate these technologies into modern control strategies such as model predictive control (MPC) to facilitate decision-making process by utilizing integrated knowledge of weather, occupant, utility.

1.2 Motivation

Within the United States, buildings are responsible for about 40% of energy usage and with demands for energy expected to increase into a foreseeable future, socio-economic pressures will drive the need for cities that are increasingly sustainable and smart about their consumption of energy resources [68, 131]. Due to the portion buildings contribute to city energy profile, intelligent buildings are where

opportunities exist for education and research.

According to a 2003 survey on commercial building energy consumption, heating, ventilation and air-conditioning (HVAC) systems are responsible for half of the energy consumed in commercial buildings [49]. Many of HVAC systems plagued by less-than-optimal operations and poorly maintained equipment. The long life-span and the footprints of buildings further emphasizes the importance of architectural and engineering disciplines to tackle these challenges [119].

Requirements for HVAC simulation and control are driven by a near-term trend toward performance-based design of buildings, and in a longer view, performance of buildings connected to the energy grid [138]. The importance of the control strategy in HVAC systems operation is due to several factors. First, as people become more aware of the benefits of increased comfort in a (indoor) controlled environment, those experiences lead to higher expectations [104,141]. Second, there is a growing need to reduce energy consumption, particularly of fossil fuels [59,141]. As a result, advanced control algorithms are required to achieve low levels of energy consumption in the buildings.

These facts, coupled with long-term forecasts for a steady increase in the demand for energy, point to a strong need for new approaches to building climate control that reduce levels of energy demand and greenhouse gas emissions.

1.3 State-of-the-Art Building Energy Control

State-of-the-art buildings are now recognized not only by their sustainable design and architecture, but more importantly by their functionality without the intervention of humans.

BASs provide a centralized control of a building's HVAC, lighting and other systems in an automated fashion. Due to significant improvements in sensing capabilities and automated control strategies have caused modern buildings operating more efficiently and started adhering to the ever increasing demand for better indoor environmental quality. Due to the advancements in sensing systems for building energy monitoring, BASs have much data about the building but, the data has too little structure. Consequently, the control decisions are made merely on numeric values obtained from the sensors and the knowledge derived from data is overlooked. Moreover, in many cases, the building control unit use strategies that are based on practical experience. As a case, the control logic may be programmed to work based on a fixed schedule (time dependent), or implemented as a set of rules (rule-based) which may be far from optimal.

To address these shortcomings there is a need for new approaches to building control strategies that employ mixtures of formal and advance MPC control algorithms. The formal methods will provide the MPC controllers in integrative knowledge about the building state as compared to the state value inputs in the conventional MPC. The MPC algorithms will then ensure that the optimal decision is made within the constraints based on the current integrative knowledge of

building and surrounding domains.

1.4 Research Scope and Objectives

This dissertation focuses on taking the initial step in integrating semantic web technologies for knowledge representation and reasoning in real buildings. The scope of investigation includes development of ontologies and rule sets to study the potentials offered by these technologies in HVAC systems simulation settings. It investigates how ontologies from a multitude of domains – utility, weather, occupant, and sensor – along with relevant rule sets can work together to provide multi-domain decision making support to the operation and fault detection in HVAC systems. To this end, we propose that the ontologies be developed in OWL2 and modified in Apache Jena [7], and that Jena Rules be used for the representation and creation of new knowledge.

Furthermore, it investigates leveraging this integrative semantic knowledge in MPC controller for HVAC systems. This investigation is supported by case studies relevant to HVAC systems, where the MPC queries the knowledge bases to obtain knowledge regarding occupancy position (Occupant and Building Ontology) and utility tariffs (Utility Ontology) to make optimal control decision. Simulation of these test problems involved use of the Modelica Building Library [92] and techniques for co-simulation between model predictive control (MPC) for the physical model is performed. Lastly, it investigates how automated inference-based fault detection and diagnostics can be achieved with developing ontologies. This approach is also

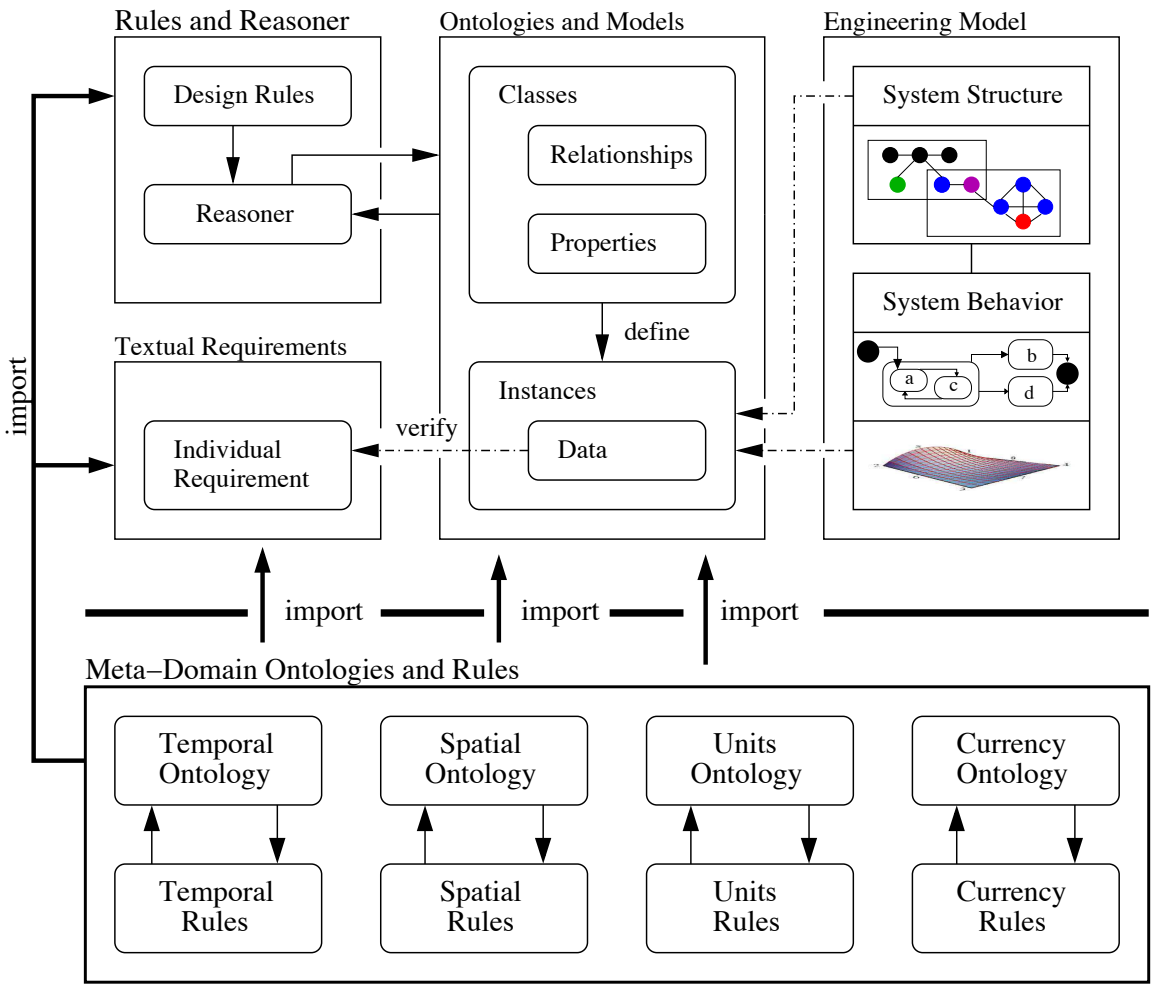


Figure 1.1: Framework for implementation of semantic models using ontologies, rules and reasoning mechanisms.

tested with a test problem.

To understand the appeal and potential of the proposed approach, Figure 1.1 shows the interconnection of knowledge base and physics-based systems. In real buildings, the sensors collect data from physical components, i.e. pump, valve. The data will be stored in the relevant ontologies. A change in the system state will change the property values for the individuals in ontologies. Consequently, ontological data will trigger the reasoner to perform rule checking tasks. Firing

the rules will trigger the data being processed based on the rule sets. and their information will be used in decision making tasks. After decisions are made, the required actions will performed by actuators. For the building, each domain, i.e. building, environment, weather, sensor and occupant will have a graph that evolves according to a set of domain-specific rules.

1.5 Research Hypothesis and Questions

Research Hypothesis. The hypothesis of the proposed program of research is as follows:

> Automated building operations can benefit from semantic knowledge representations and reasoning procedures designed to take advantage of Semantic Web technologies, data-driven development of ontologies and rules, and modern techniques in software engineering.

The motivating tenet of the research is that, ultimately, these next-generation techniques will become an integral part of procedures for optimized building automation, where monitoring, simulation, and optimization-based control decisions are made without the intervention of humans. This capability will allow humans to spend more time participating in the activities for which the building environment was created, and from system-level building perspective, lead to superior levels of building energy systems performance.

Research Questions. The program of investigation will attempt to answer the

following research questions:

1. **Knowledge Representation.** How can the knowledge of surroundings, i.e., occupants, weather, equipment be passed to control units?

2. **Inferencing.** What mechanisms can control units utilize to infer information from existing data? How easily can these reasoning mechanisms span domains?

3. **Integration.** What platform infrastructures make sense in terms of supporting integration of ontologies and rule sets with advanced control strategies and state-of-the-art system simulation models?

4. **Automated Inference-based Fault Detection and Diagnostics (FDD).** How can FDD mechanisms be implemented to mimic human thinking processes used to identify the source of a fault, possibly requiring comprehensive cross-domain knowledge of the system and its surroundings?

5. **Software.** What role can modern software technologies (e.g., JAXB) play in the development of software that can easily handle a variety of data specifications? What opportunities exist for simplifying software prototypes through the judicious use of software design patterns?

1.6 Contributions

This dissertation project investigates opportunities for using semantic models and reasoning in building system operations, with a focus on knowledge-based control and fault detection for heating, ventilation and air conditioning (HVAC)

systems. Portions of this work have been published in conference proceedings [9, 37–40, 43–45] and archival journals [41, 42], and can be highlighted as follows:

1.6.1 Contribution 1: Framework for Data-Driven Semantic-based Reasoning

In state-of-the-art development of semantic models, a common strategy is to provide classes and data properties for all possible configurations within a domain, as well as linkage to related domains. For example, in the integrated model-centric engineering ontologies (IMCE) developed at JPL (Jet Propulsion Laboratory) during the 2000-2010 era [15, 126], the electrical engineering ontology (i.e., electrical.owl) imports the mechanical engineering ontology (i.e., mechanical.owl). Both the electrical and mechanical engineering ontologies import a multitude of foundation ontologies (e.g., analysis.owl, mission.owl, base.owl, project.owl, time.owl) and make extensive use of multiple inheritance mechanisms in the development of new classes. However, multiple inheritance can cause ambiguity in several scenarios.

In a step toward mitigating this problem, the first contribution of this dissertation is development of semantic-modeling framework (see Figure 1.2) that supports: (1) concurrent data-driven development of domain models, ontologies and rules, and (2) executable processing of events. Instead of creating ontologies and then developing a few rules for validation of model properties, the goal is to put the development of data, ontologies and rules on an equal footing. A key advantage of this approach is that it forces designers to provide semantic representations for

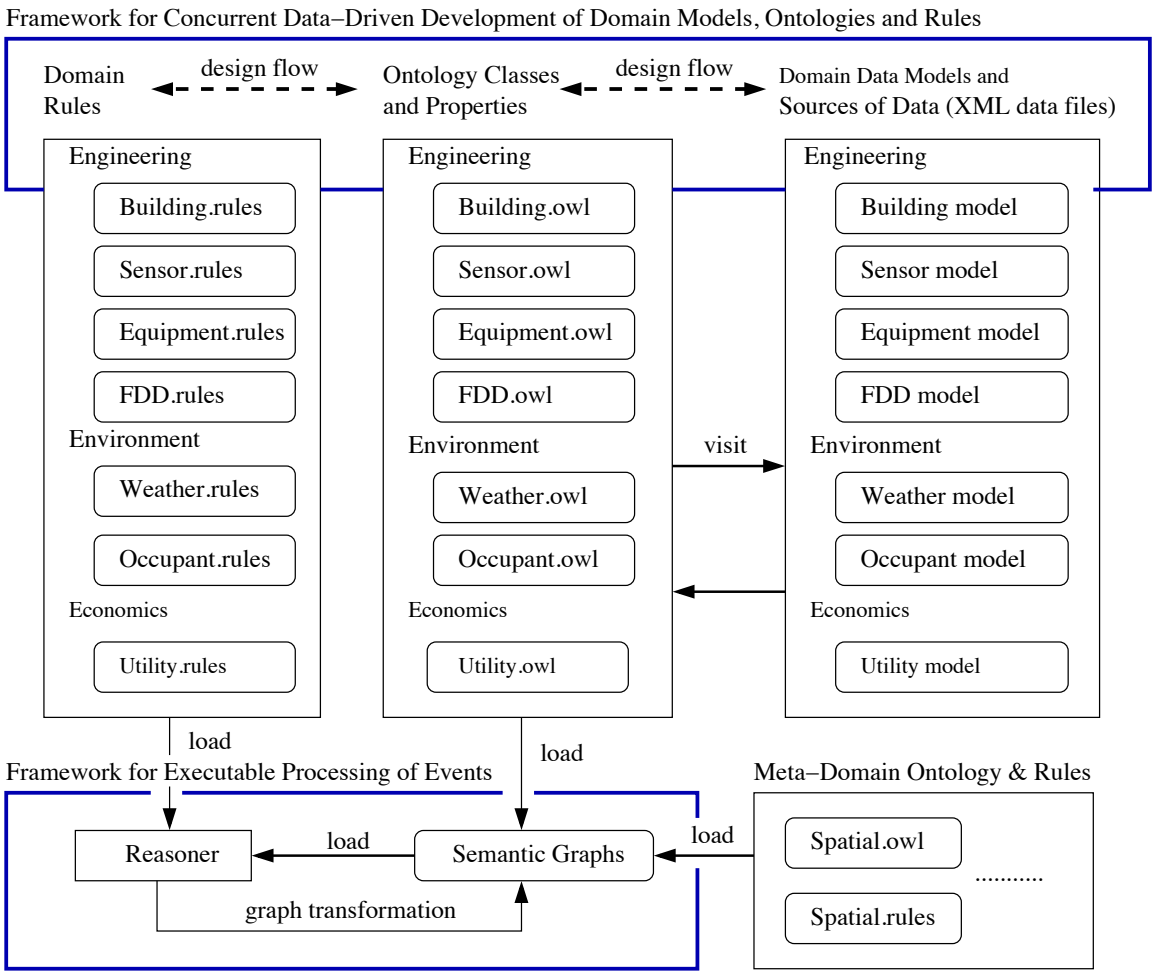


Figure 1.2: Contribution 1: Systems architecture supporting: (1) concurrent data-driven development of domain models, ontologies and rules, and (2) executable processing of events.

data that are needed in decision making, and increases the likelihood that data not needed for decision making will be left out. Rules will be developed for verification of domain properties and processing of faults through reasoning with data sources, possibly from multiple domains. Implementation of the latter goal leads to semantic graphs that will dynamically adapt to the consequences of incoming data and events (e.g., changing occupant locations and weather events) acting on the system.

The second strategy is to minimize the use of multiple inheritance in the specification of OWL ontologies and, instead, explore opportunities for replacing inheritance relationships by object property relations and rules that reach across disciplinary boundaries. In order for the architectural framework to be both scalable and adaptable to changing external conditions, the ontologies will need to be modular, and the rules will need to act both within a domain and across domains.

1.6.2 Contribution 2: Framework for Integration of Semantic and Physics-based Modeling

The second contribution involves integration of semantic models (ontologies) with advanced control strategies, such as those provided by MPC, and system simulation data, for example, as generated by Modelica models. The key benefit of this contribution is a means for MPC to use inferred knowledge about the system for its decision making.

Figure 1.3 depicts how semantic web technologies such as ontologies are used for knowledge-based data-driven control. These technologies offer solutions for

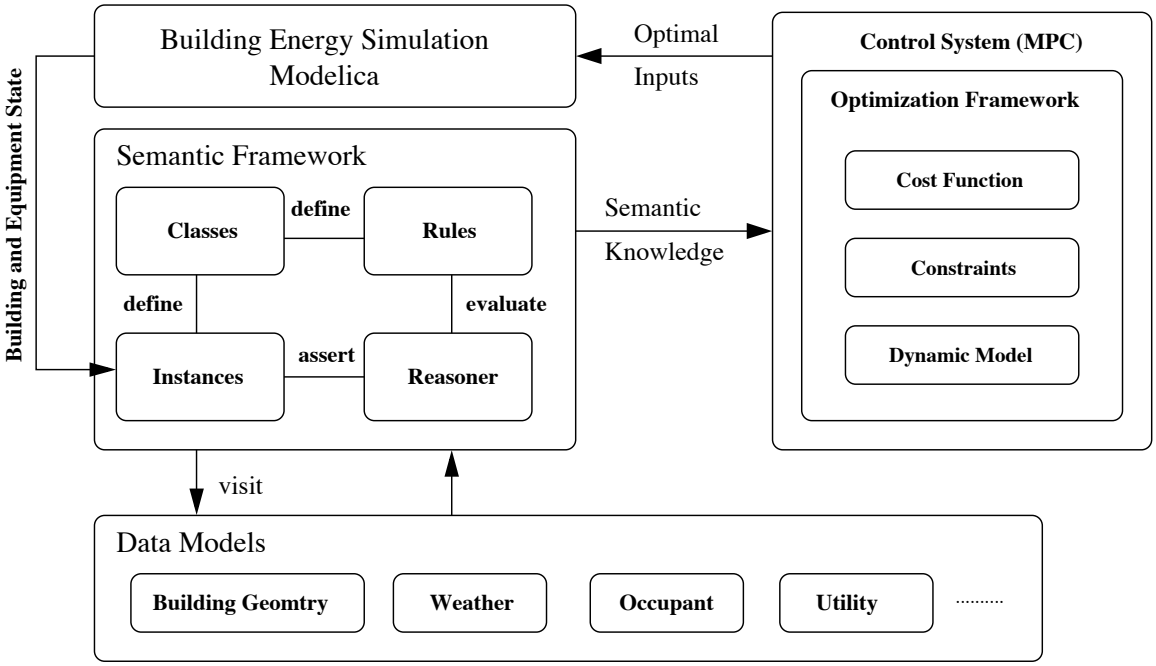


Figure 1.3: Contribution 2: Integration of semantic model with control systems and building energy simulation models.

inference-based decision making through expressive features of Descriptive Logic (DL) based on the existing data came from building energy simulations and the environment. This inferred knowledge will be set as inputs to optimization-based control methods such as MPC.

1.6.3 Contribution 3: Framework for Event-based Fault Detection and Diagnosis

The third contribution is development of a semantic framework for event-based fault detection and diagnostic framework that mimics a human's thinking in detecting a fault and diagnosing the underlying cause.

Figure 1.4 proposes an interdisciplinary approach for fault detection and diagnostics

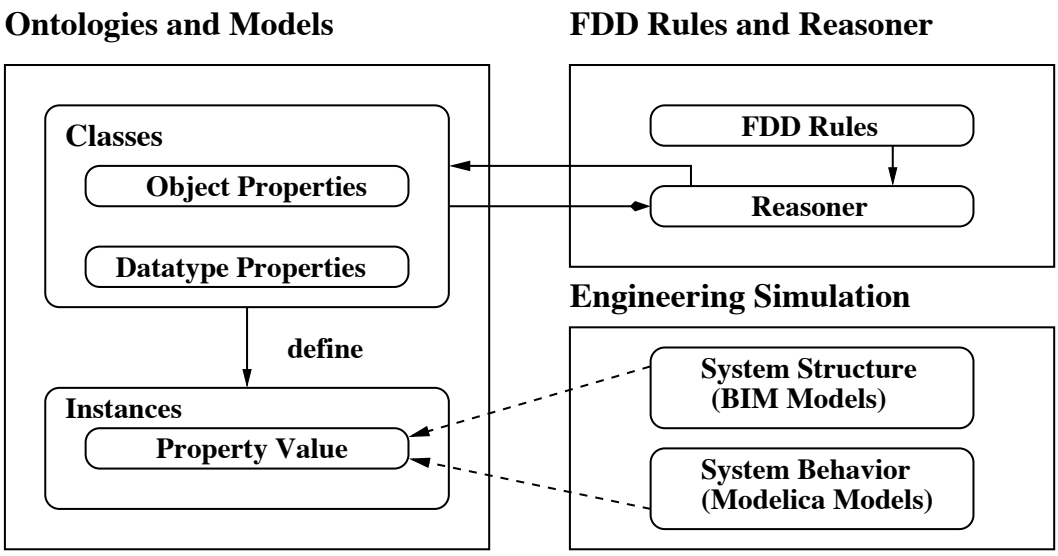


Figure 1.4: Architecture for coupled integrated semantic physical models in building simulations for Fault Detection and Diagnostics (Adapted from Delgoshaei, Austin and Pertzborn [42].

in buildings. This approach utilizes knowledge repositories, ontologies, for storing automation/simulation data and then apply inference-based reasoning techniques to obtain additional higher level of information. The rules will identify a discrepancy in an expected behavior to detect a fault. Moreover, for any specific system configuration new rules can be developed to set the evidence as the system dynamic changes over time. Finally, with semantic querying of the ontology. A list of detected faults and their associated causes can be identified.

1.6.4 Contribution 4: Improved Understanding for using Software Design Pattern to support Systems Integration

The fourth contribution of this dissertation is an improved understanding for how software design patterns can enable the system modeling and systems integration visions implied by Figures 1.1 and 1.2. While it is relatively easy to draw a

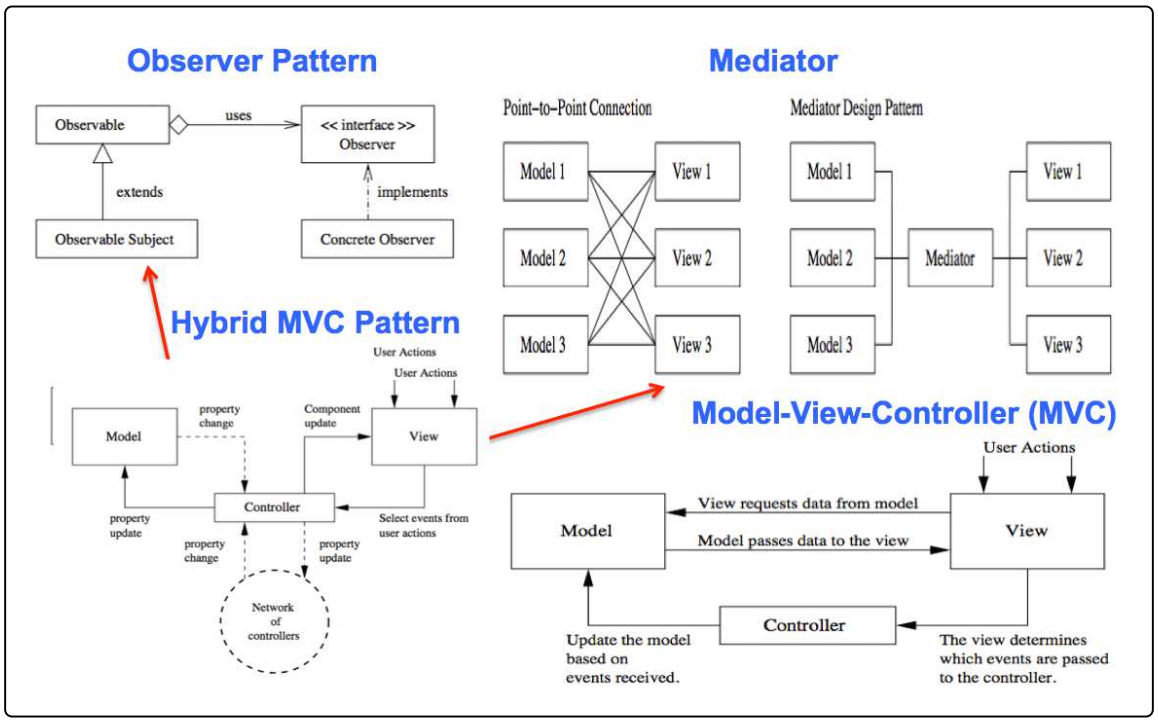


Figure 1.5: Collage of software design patterns.

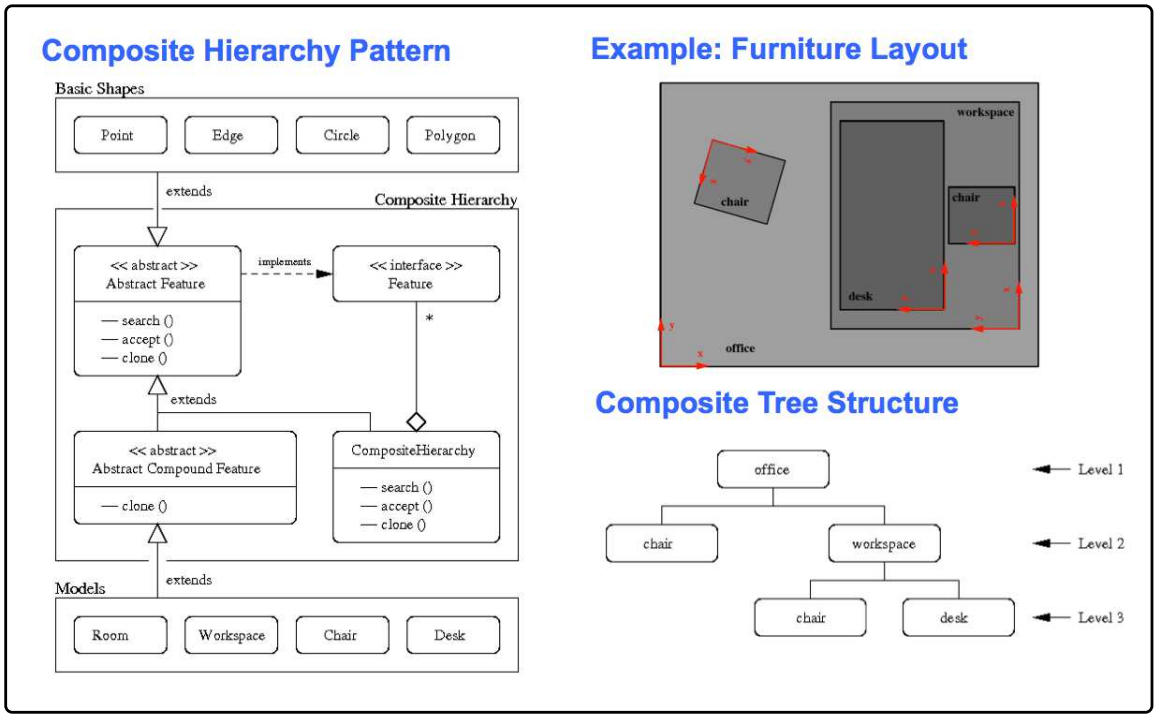


Figure 1.6: Collage of composite software design patterns and simple applications.

picture for how you think a system should work, creating a software prototype that actually works is an entirely different matter. The judicious use of software design patterns, sometimes in ways that are far from obvious, can make the difference between a prototype system that barely works and one that works so well it's clear it will scale beyond the bounds of the current study.

Figures 1.5 and 1.6 are collages of software design patterns used throughout this work. For example, Jena uses the observer design pattern to detect (and then handle) events in the semantic graph. The model-view-controller (MVC) software design pattern plays a central role in the development of software for executable statecharts (see Figure 1.1). In Figure 1.6, the visitor design pattern (not shown) is used to create a bridge between the ontology and data models. The composite hierarchy pattern is used in the modeling of building floor plans (see Case Study 3 in Chapter 5).

1.7 Organization

The dissertation is organized into six chapters and one appendix. Chapter 2 introduces concepts, languages and tools used in the Semantic Web. These tools and languages will be used extensively in semantic modeling. Chapter 3 summarizes the existing modeling, simulation and control strategies for building systems. Knowledge representations and the ontologies developed relevant for building HVAC energy systems are discussed in Chapter 4. In Chapter 5 we exercise the proposed methodologies and analysis procedures by working step by step through three case

study problems:

1. Semantic-assisted model predictive control (MPC) for detection of occupant thermal comfort,

2. Semantic-based utility description for MPC in a Chiller Plant Operation,

3. Knowledge-based fault detection and diagnostics for HVAC systems.

Chapter 6 contains the conclusions of this study along with suggestions for future work. Appendix A is a brief overview of systems integration and behavioral simulation with Whistle, a prototype scripting language developed during the course of these studies.

Chapter 2: Language and Tools for Semantic Knowledge Representation

This chapter introduces the Semantic Web vision, and the range of languages, technologies and tools found in its implementation. Basic capabilities of the resource description framework (RDF) and Web Ontology Language (OWL) are described in Sections 2.2 and 2.2.2.

Simple case study problems involving event-driven behavior modeling of component dynamics, and modeling of HVAC components in buildings with ontologies (Jena) and rules (Jena Rules) is presented. Finally, Section 2.5.4 ties out the above-mentioned concepts and describes how they will be used in the next generation of intelligent buildings.

2.1 The Semantic Web

2.1.1 Semantic Web Vision

The World Wide Web was invented in 1989 by Tim Berners-Lee, with the initial purpose to meet the demand for automatic information-sharing among members of scientific communities [17]. At that time, Berners-Lee identified two main goals

for the World Wide Web:

1. To make the Web a collaborative medium and,

2. To make the Web understandable and automatically processable by machines.

Over the past twenty years the first part of this vision has come to pass. The development of Web browsers created a means for humans to retrieve and render information, and then manually interpret and understand the meaning of the content. A second more ambitious vision for the Web is support for semantic data structures and pathways for machine-to-machine communications that carry the semantic meaning for data in addition to its values. Thus, instead of broadly searching for someone based, perhaps based on a few keywords, semantic web provides a means to search precisely for someone based on their name, plus semantic relationships to places of employment, attendance at events, age, and so forth.

2.1.2 Semantic Web Technical Structure

Figure 2.1 illustrates the technical infrastructure that supports the Semantic Web vision, and the range of languages which we will employ to build system-behavior models.

Each layer exploits and uses capabilities of the layers below. The lower layers provide capability for addressing resources on the Web, linking documents, and representing multiple languages. Specifically, the extended markup language (XML) enables the construction and management of data organized into tree structures,

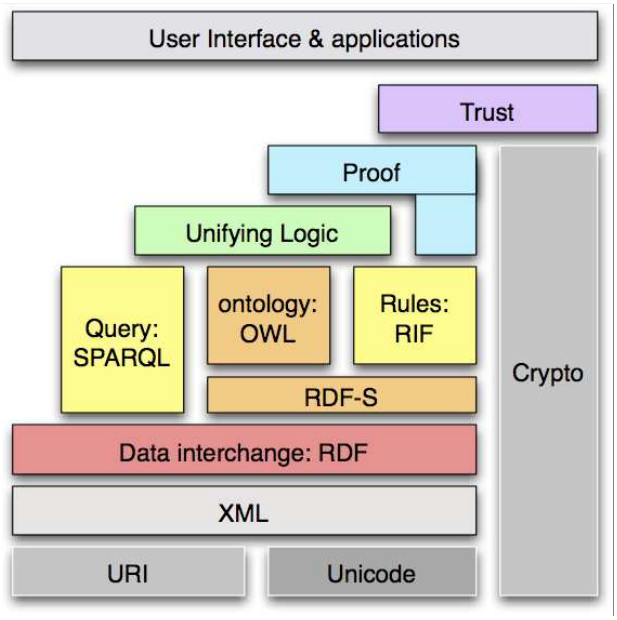


Figure 2.1: Technologies in the Semantic Web Layer Cake [51].

which is fine for storing data, but less suitable for integration of data from multiple sources. The resource description framework (RDF) takes a step toward solving this problem by allowing for the modeling of graphs of resources on the Web. An RDF Schema (RDF-S) provides the basic vocabulary for RDF statements, and the machinery to create hierarchies of classes and properties. Our semantic models make extensive use of the Web Ontology Language (OWL) and expressive features of descriptive logic (DL) formalisms. Inference-based mechanisms allow a system to infer a new statement from existing statements. Semantic Web features and language capabilities provide the foundations for representing knowledge bases (e.g., in the building, HVAC equipment and weather domains) and reasoning over that knowledge to detect faults and systematically verify hypotheses through evaluation of supporting evidence. This dissertation uses OWL for semantic modeling.

2.2 Languages for Semantic Modeling

The following are the most common ontology description languages.

2.2.1 Resource Description Framework (RDF)

XML is a mark-up language that supports portable encoding of data. That is, it is limited to represent information that can be organized within hierarchical relationships. Consequently, RDF was introduced as a graph-based assertional standard model for data interchange on the Web. RDF extends the linking structure of the Web to include the relationship between the resources on the web represented as a triple structure. This approach allows structured and semi-structured data to be mixed and shared across different applications. In the Semantic Web, RDF identifies recourses by their Web identifiers (URIs). RDF can represent circular references between resources, graph-based models, and resolve the inherent problem of the hierarchical structure of XML. Triples are described by assertions, the smallest expression of information, referred to as facts. RDF captures assertions made in simple format of a "triple" by connecting a subject to an object through a predicate (verb), shown in Figure 2.2.

RDF is a simple representation of domain facts, focused on describing the instances and the mapping to their types (rdf:type). In this representation, however, the semantic is missing it is possible to assert facts that are not semantically sound. In essence, RDF transforms English statements to machine processable format.

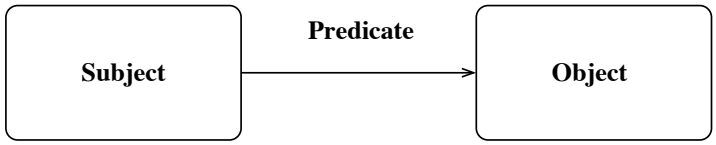


Figure 2.2: Example of RDF triple where node A is a subject, "predicate" is a verb, and node B is an object.

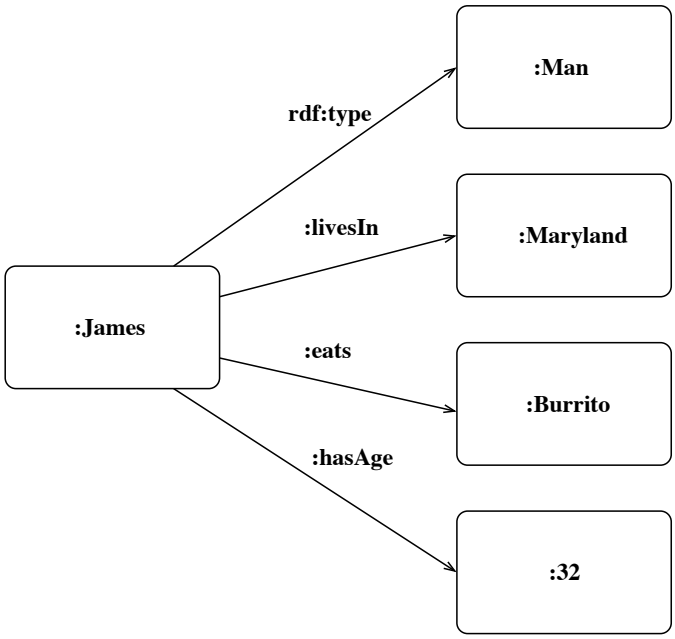


Figure 2.3: A simple RDF graph showing information about James.

These RDF triples are consisting of a subject (this is the entity the statement is about), a predicate (this is the named attribute, or property, of the subject) and an object (the value of the named attribute). They are all denoted by unique URIs. Each property will have a specific meaning and may define its permitted values, the types of resources it can describe, and its relationship with other properties. Objects are denoted by a datatype or URI. Figure 2.3 illustrates, for example, a graph model of relationships relevant to a person named James. Here are the associated triples serialized in Turtle syntax:

```
@prefix : <http://www.example.org/> .
```

```
:James    rdf:type           :Man .
:James    :livesIn  ‘‘Mayland" .
:James    :eats  ‘‘Burritos" .
:James    :hasAge  ‘‘32" .
:livesIn    rdf:type    rdf:Property .
:eats    rdf:type    rdf:Property .
:hasAge    rdf:type    rdf:Property .
```

Limitations of RDF. One limitation of RDF is that it is not expressive enough to capture knowledge attributes such as existence and cardinality, transitivity, inverse or symmetrical properties [57]. This makes this framework weaker to describe resources in sufficient detail. Also, RDF is not based on a mathematical logic and as a result does not support reasoning mechanisms. Moreover, in RDF representation has no semantic restrictions on data. As a case, it is possible to assert a statement like : $Burritos : eats“James”$. It is a perfect an meaningful assertion in RDF. However, it is semantically wrong. The Web Ontology Language (OWL) was developed to address the weaknesses of RDF.

2.2.2 Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a description logic based (DL-based), mathematical theory, knowledge representation language with the highest level of expressivity for constructing ontologies. OWL is based on the basic features of RDF, but extends and improves the basic features of RDF by enhancing the expressiveness and support for richer property definitions (e.g., transitivity), class property restrictions (e.g., someValuesFrom), quality cardinality restrictions (e.g., Qualified-Cardinality “3) equality between classes (e.g., sameAs) and relations between classes (complementOf). These additional capabilities allow ontological systems to use rea-

soning structures and infrastructure to infer new facts (triples) from existing ones with first-order logic (FOL) as a mathematical foundation.

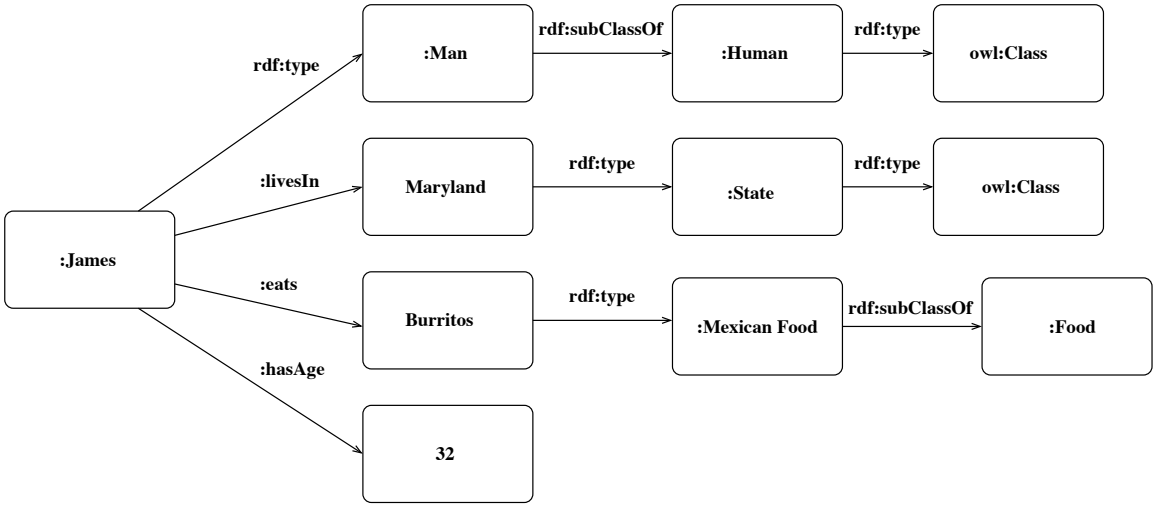


Figure 2.4: An OWL graph of relationships describing the semantics about a Human.

To explain some of the powerful features of OWL previous example is revisited. Figure 2.4, and the formal representation in RDF/XML syntax in Figure 2.5 describe an ontology about a person and the associated properties. The class Man, Human, State, Food, Mexican Food defined. James, Maryland, Buritos are now individuals of classes Man, State and Mexican Food. A class may have a datatype property such as :hasAge. These properties are simple datatypes, e.g., String, double, boolean. Moreover, classes can have object properties that define the relationship between two classes. In this example property :eats is the object property that defines the relationship between the individuals in class Human and the individuals in class Food. This feature will prevent assertions that are semantically wrong. For example $: Burritos : eats : James.$ is not a valid assertion in OWL.

The rdfs:domain and rdfs:range properties are used to specify the domain and

```
<!--  Classes -->

<owl:Class rdf:about="http://example.org/person#Human">
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<owl:Class rdf:about="http://example.org/person#Food">
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<owl:Class rdf:about="http://example.org/person#MexicanFood">
<rdfs:subClassOf rdf:resource="http://example.org/person#Food"/>
</owl:Class>

<owl:Class rdf:about="http://example.org/person#Man">
<rdfs:subClassOf rdf:resource="http://example.org/person#Human"/>
</owl:Class>

<!-- Defining Individuals-->

<owl:NamedIndividual rdf:about="http://example.org/person#James">
        <rdf:type rdf:resource="http://example.org/person#Man"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://example.org/person#Tacos">
        <rdf:type rdf:resource="http://example.org/person#MexicanFood"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://example.org/person#Maryland">
<rdf:type rdf:resource="http://example.org/person#State"/>
</owl:NamedIndividual>

<!-- Defining Datatype and Object Property-->

<owl:ObjectProperty rdf:about="http://example.org/person#eats">
    <rdfs:domain rdf:resource="http://example.org/person#Human"/>
<rdfs:range rdf:resource="http://example.org/person#Food"/>
</owl:ObjectProperty>

<!--  Property Assertion -->

<owl:NegativePropertyAssertion>
     <owl:sourceIndividual rdf:about="http://example.org/person#James">
     <owl:assertionProperty rdf:about="http://example.org/person#eats">
    <owl:targetIndividual rdf:about="http://example.org/person#Tacos">
</owl:NegativePropertyAssertion>
</rdf:RDF>
```

Figure 2.5: Formal definition of a Person and Properties in OWL.

range of a property. The rdfs:domain of a property specifies that the subject of any statement using the property is a member of the class it specifies. Similarly, the rdfs:range of a property specifies that the object of any statement using the property is a member of the class or datatype it specifies.

The English-like syntax of OWL makes it a much easier to understand and use as a logical language such as $\mathcal{SHOIN}$ DL. There are two versions of OWL used for ontology description, OWL1 and OWL2. For example, using DL feature of OWL, triple : $Jamesrdf$: $type$: $Human$ is inferred from triples : $Jamesrdf$: $type$: man and : $manowl$: $subClassOf$: $Human$.

Today, ontologies are developed in OWL for a diverse range of application areas spanning engineering, medicine, biology, geography, and defense. As a case in point, NASAs SWEET ontologies [115] and BioPAX [18], are the ontologies used in engineering and biology respectively.

Versioning of OWL. OWL1 encompasses language variants, species, of *OWL Lite*, *OWL DL*, and *OWL Full* distinguished by their increasing level of expressiveness. *OWL Lite* allows the expression of simple syntax and constraints but little support for inferencing. *OWL DL* has a user-friendly abstract syntax, inferencing is decidable and the language is computationally complete. *OWL Full* is the most expressive version of OWL1 that is compatible with RDF and RDFS languages. OWL2 adds more features to OWL1 but does not change expressiveness, semantics, complexity of the language. Moreover, it makes some patterns easier to write which allows for more efficient processing in reasoners. New features include of OWL2 are the support

for properties like DisjointUnion, DisjointClasses, NegativeObjectPropertyAssertion and NegativeDataPropertyAssertion.

OWL2 comprises three independent profiles (or sub-languages) that restrict its structure in different ways : expressiveness (OWL2 - EL), Querying (OWL2 - QL) and Reasoning (OWL2 - RL). These profiles follow the trends with OWL1 spices. i.e., OWL Lite (limited expressiveness), OWL1 DL (decidable with support for DL) and OWL1 Full (full expressiveness) [57].

Syntax. OWL 2 ontology consists of a single set of axioms that include both conceptual and instance level statements. The syntax is easy to read and close to English. As a case, in abstract sysntax EquivalentClasses(Person Human) or rdf syntax$< Person, rdfs : subClassOf, Human >$ describes that Class People is the same as Class Human.

2.3 Semantic Modeling with Ontologies and Rules

Semantic models consist of creating graphs of individuals (specific instances), and inference-based rules in the form of $if < conditions >$ then $< consequent >$. Together all these pieces form a knowledge base for a specific domain. As a case, ontologies developed for mechanical equipment, weather, building, or occupant domains.

2.3.1 Ontologies

An ontology is a formal and explicit representation of the concepts of a domain as classes and the relations between those classes as "Object Properties" (the connection between two objects of classes). Moreover, the classes may have attributes that are stored as a simple data type "Datatype Properties". Ontologies, also, provide ways to define taxonomical (hierarchical) relationship between the classes that result in the inheritance of object and datatype properties in the subclass from the superclass.

Following is an example in mechanical equipment ontology.

- **Classes**: Valve, Cooling Coil
- **Datatype properties**: coilTemperature (double), isClosed (Boolean), coilSetpoint(double)
- **Object Property**: hasValve

2.3.2 Individuals

Individuals are instances of ontology concepts. They are the existing data in the domain.

Individuals: :ValveI, :ValveII, :Ccoil, :Hcoil

2.3.3 Axioms

Axioms are logical statements about the relationships between properties and/or classes in the domain. These statements are asserted to be true in the domain being described. One common paradigm to assert axioms is in triple-based format. <subject, predicate, object>. The subject is the Axiomatic systems are better candidates than non-axiomatic systems for representing the formal models to be used in the inferencing process [81]. Axiomatic systems are systems composed of axioms. Many logic systems fall into the axiomatic category, e.g., first-order and descriptive logic (DL) that is the logical formalism for ontologies defined in OWL.

- **Stored Axiom**: <:Coil :hasValve :Valve>
- **Stored Facts**: <:Hcoil :hasValve :ValveII>

2.3.4 Reasoning

One of the key features of using ontologies is using a reasoner to derive additional truths, facts, about the concepts being modeled. In the example above: the assertions <:Hcoil :hasValve :ValveII> , <Ccoil coilTemperature 35> and <:Ccoil :coilSetpoint 35> entails the deduction that <:ValveII :isClosed true> based on the following inference-based rule. The inference-based rules are mechanisms to derive new information based on the existing data stored in the ontology in the form of: if <conditions> then <consequent>.

```
// -- An inference-based Rule that infers the valve is closed
//    if the setpoint is equal to the temperature.
```

```
(?coil rdf:Type :coil) (?coil :setPoint ?sp)
(?coil :coilTemperature ?cp) equal(?cp,?sp)
(?coil :hasValve ?valve)  -> (?valve: isClosed true)

Stored facts : <Hcoil :hasValve :ValveII>
                     <:Ccoil :coilTemperature 35>
                     <:Ccoil :coilSetpoint  35>
Inferred facts:<:ValveII  :isClosed true>
```

There are different ontology descriptions with varying features and capabilities explained in Section 2.2. Their purpose is to define ontologies that include classes, properties and their relationships to encode the semantics of the domain in a way that is machine processable. That is, these languages provide a standard and unambiguous way for machines to effectively understand and reason about contextual information or context may refer to an existing entity of the domain such as, people, equipment, sensor. In essence, these contextual information shape the knowledge of the domain processable by machines.

2.4 Semantic Web Tools

This section provides an overview of the OWL and Jena for the Semantic Web modeling.

2.4.1 OWL Supported Tools

There are application programming interfaces APIs such as OWL API, Thea, OWLink, Jena API and development and editorial environments (e.g., Protege [103], Swoop [75]) that support OWL. The major DL reasoning systems that support

OWL syntax are Pellet [112], Racer [60], HermiT [93], FaCT++ [121]. SPARQL is the querying language for RDF [97]. However, since OWL can be serialized to RDF, SPARQL languages can be used in OWL ontologies. However, their knowledge of OWL is incomplete. A more efficient language to query OWL ontologies is SQWRL [97].

2.4.2 Working with Jena and Jena Rules

Our prototype software implementation makes extensive use of Apache Jena and Jena Rules. Apache Jena [7] is an open source Java framework for building Semantic Web and linked data applications. Jena provides APIs (application programming interfaces) for developing code that handles RDF, RDFS, OWL and SPARQL (support for query of RDF graphs).

The Jena inference features allow a range of inference engines, reasoners, to be used on semantic models. Jena Rules is one such engine. Jena Rules employs facts and assertions described in OWL to infer additional facts from instance data and class descriptions.

As illustrated in Figure 2.6, it also provides support for the development of builtin functions that can link to external software programs and streams of data sensed in the real world. For the implementation of the vision implied by Figure 1.2, particularly support for spatial and temporal reasoning, the latter turns out to be crucially important because, by default, OWL only provides builtin datatype support for numbers (i.e., float and double), booleans (i.e., to represent true and

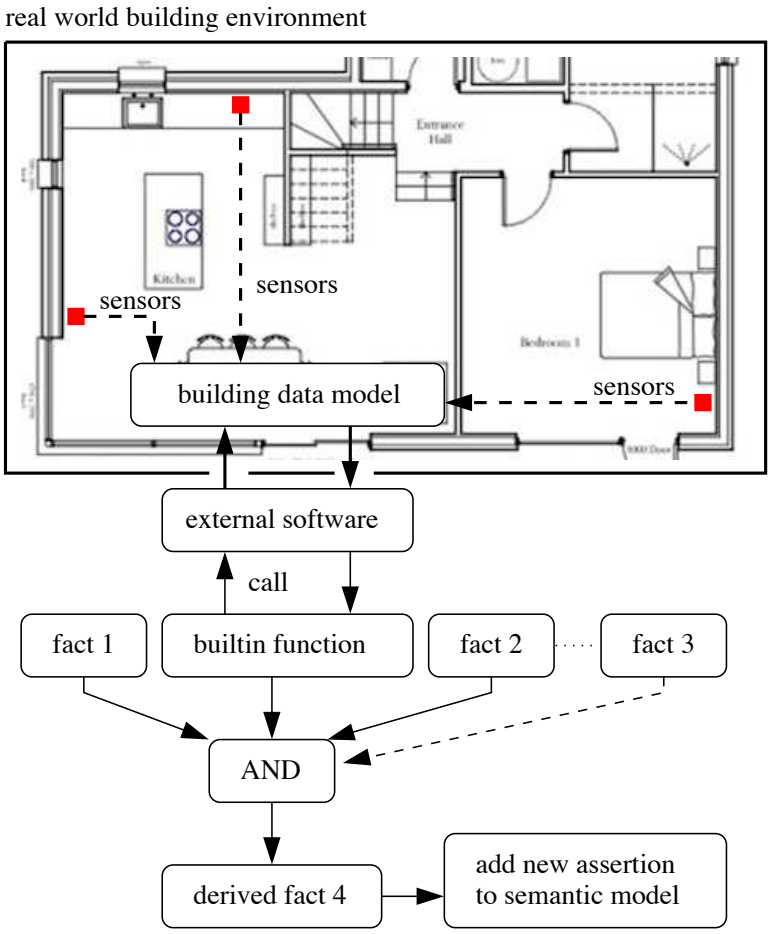


Figure 2.6: Framework for forward chaining of facts and results of builtin functions to new assertions (derived facts).

false) and character strings (i.e., string). To combat the lack of support for complex data types, such as those needed to represent data for spatial and temporal reasoning, we adopt a strategy of embedding the relevant data in character strings, and then designing built-in functions and external software that can parse the data into spatial/temporal models, and then make the reasoning computations that are required.

2.5 Simple Examples

2.5.1 A Jena Rule to Reset the Room Setpoint.

The following is an example of Jena rule that reset the room setpoint when room is not occupied. In this representation if all the assertion in the left hand side of the rule is true, the right hand side will be asserted in the ontology graph.

```
@prefix sen:  <http://umd.edu/sensor#>.
@prefix bld:  <http://umd.edu/building#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[Rule01: (?ts rdf:type sen:TemperatureSensor)
         (?ts bld:isLocated bld:?r)(?r bld:occupied false)
         (?r bld:hasSetpoint ?setPoint)
         lessThan(?setPoint,23C) -> resetSetPoint(?setPoint,20)]
```

2.5.2 Simplified Modeling of Event-Driven Component Dynamics

This example demonstrates a basic ontology- and rule-based modeling of system components, valve and coils, with Jena and Jena rules. Ontologies and rules (Jena Rules) are defined for simplified behavior modeling of coil and valve dynamics.

Once the this semantic model is assembled, the graph of individuals, and relationships will evolve in response to events.

Definition of the Component Ontology. Figure 2.7 shows a simplified component ontology, the relationship among valve and coil classes and properties.

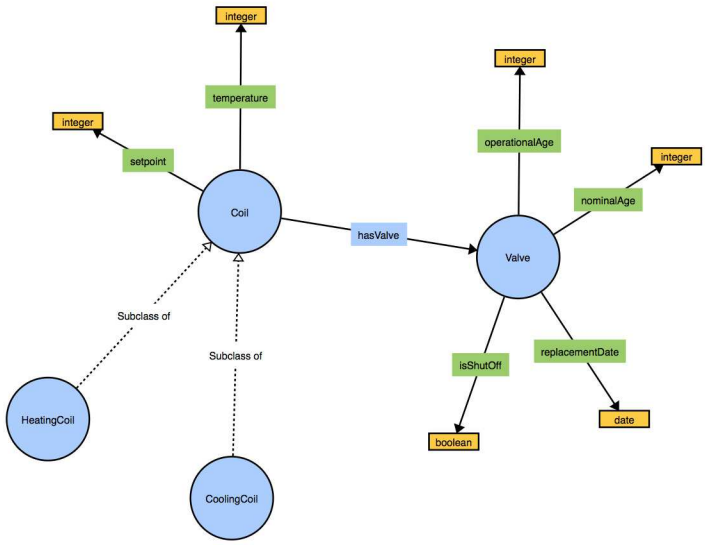


Figure 2.7: Relationship between classes and properties in the component ontology.

The ontology class Valve has properties: operationalAge, nominalAge, isShutOff, and replacementDate. They will be modeled as data types double, double, boolean and date, respectively. Class coil has datatype properties of setpoint and temperature that are double and an object property of hasValve that models the relationship between a specific coil and valve. CoolingCoil and HeatingCoil classes are a specialization of Coil.

Adding Facts and Rules.

Fact 1: ValveI was replaced on October 1, 2010.

Fact 2: CoilI hasValve ValveI.

Fact 3: CoilI has a setpoint of 21°C.

Fact 4: CoilI has a temperature of 24°C.

The following rules can be declared:

Rule 1: For a given a replacement date and a current time, a built-in function `getAge()` computes the valve's operational age. When the operational age is the same as nominal age, the flag for dueForReplacement is set to true.

Rule 2: For all the coils at anytime, if the coil temperature is the same as the coil setpoint, the coil valve will be shut off.

Figure 2.8 shows the evolution of the ontology graph defining properties of CoilI and ValveI as a function of time.

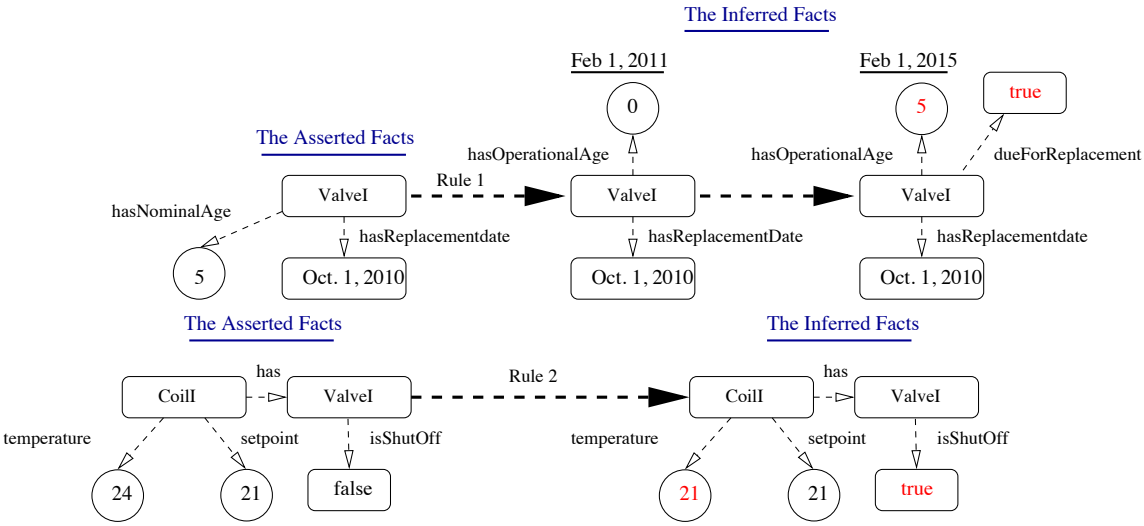


Figure 2.8: Evolution of ontology graph as a function of time.

Some of the data (e.g., valve's replacement date) remains constant over time. Other data (e.g., such as whether or not the valve is due for replacement or being shut off) is dynamic and depends on the rule execution. The rules will be triggered when specific time event happens (event-driven) or the data changes in the graph (data-driven).

Definition and Organization of Ontology Classes. The abbreviated fragment of code below demonstrates the creation of the component ontology classes, their assembly into a hierarchy, and definition of data properties for the class `Coil` in Jena API.

```
// Define classes ...

valve  = model.createClass( ns + "Valve");
coil    = model.createClass( ns + "Coil");
coolingCoil    = model.createClass( ns + "CoolingCoil");
heatingCoil    = model.createClass( ns + "HeatingCoil");

// Define relationships among classes ...

coil.addSubClass ( CoolingCoil );
coil.addSubClass ( HeatingCoil );

// Create data properties for the class Person ...

setpoint = model.createDatatypeProperty( ns + "setpoint");
setpoint.setDomain(coil);
setpoint.setRange( XSD.double );

hasValve = model.createObjectProperty( ns + "hasValve");
hasValve.setDomain( coil );
hasValve.setRange( valve );
```

The data property `temperature` is a double. The object property `hasValve` is of type Valve. Notice that since `CoolingCoil and HeatingCoil` are a subclasses of `Coil`, they automatically obtained the properties setpoint and hasValve through

class hierarchy inheritance.

Adding Individuals to the Component Model. After constructing the ontology, the next step is to define the individuals, the data associated with each individual, and the relationship of one individual to other individuals. The fragment of code below establishes a name space for the component ontology, creates a graph model for the storage of individuals and their data and object properties, and then creates a valve Individual, `ValveI`, and a data property statement for the replacement date.

```
// Namespace for the valve ontology ...

String ns = "http://building.org/valve#";

// Create ontology model (a graph) ...

OntModel model = ModelFactory.createOntologyModel();

// Add "ValveI" to the component graph model ...

Individual valveI = boy.createIndividual( ns + "ValveI" );
model.add ( valveI );

// Create statement: ValveI replacement date is 2010-10-01.

Literal rdate = model.createTypedLiteral( "2010-10-01", XSDDatatype.XSDdate );
Statement cbd = model.createStatement(valve1, hasReplacementDate, rdate );
model.add ( cbd );
```

Time Event-Driven and Data-Driven Graph Transformations (Jena Rules). Given the facts and the rules described above, graph transformations will occur as the time and data change. `ValveI` was replaced on October 1, 2010. Given a replacement date and a current time, a built-in function `getAge()` computes ValveI 's operational age. Furthermore, rule rules can be defined to identify if the valve is due

for replacement. Figure 2.8 shows the evolution of a graph defining the properties of `ValveI` as a function of time and property changes. The abbreviated fragment of code below is taken from the Jena Rules for the component ontology.

```
@prefix :  <http://austin.org/valve#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Coil Replacement Rule ....

[ rdfs01: (?valve rdf:Type :Valve) (?valve :hasNominalAge ?nage)
           (?valve :hasReplacementDate ?rdate)(?valve :hasOperationalAge ?oage)
           getAge(?rdate ?oage) greaterThan(?oage,?nage) ->
           (?valve :dueForReplacement true) ]

// Rule 02: Valve Shut off ....

[ rdfs02: (?valve rdf:Type :Valve) (?coil rdf:Type :Coil) (?coil :has ?valve)
          (?coil :temperature ?t)(?coil :setpoint ?s) equal(?t,?s) ->
          (?valve :isShutOff true) ]
```

These rule examples indicate how the graph dynamics change as the time and data change. The first rule serves two purposes. First, given a valve's replacement date, the *GetAge* function computes the operational age of the valve and inserts it into the semantic model via the `hasOperationalAge` data property. Secondly, it identifies if the valve is due for replacement if the operational age is the same as the nominal age. The second rule example indicates how the behavior of a component can be described in terms of rules. As a case, when the coil temperature reaches the setpoint temperature, the

2.5.3 Semantic Modeling of Valve Behavior

Figure 2.9 illustrates the appeal of behavior modeling with ontologies and rules. It captures the semantics behind the dynamic behavior of a mixing valve.

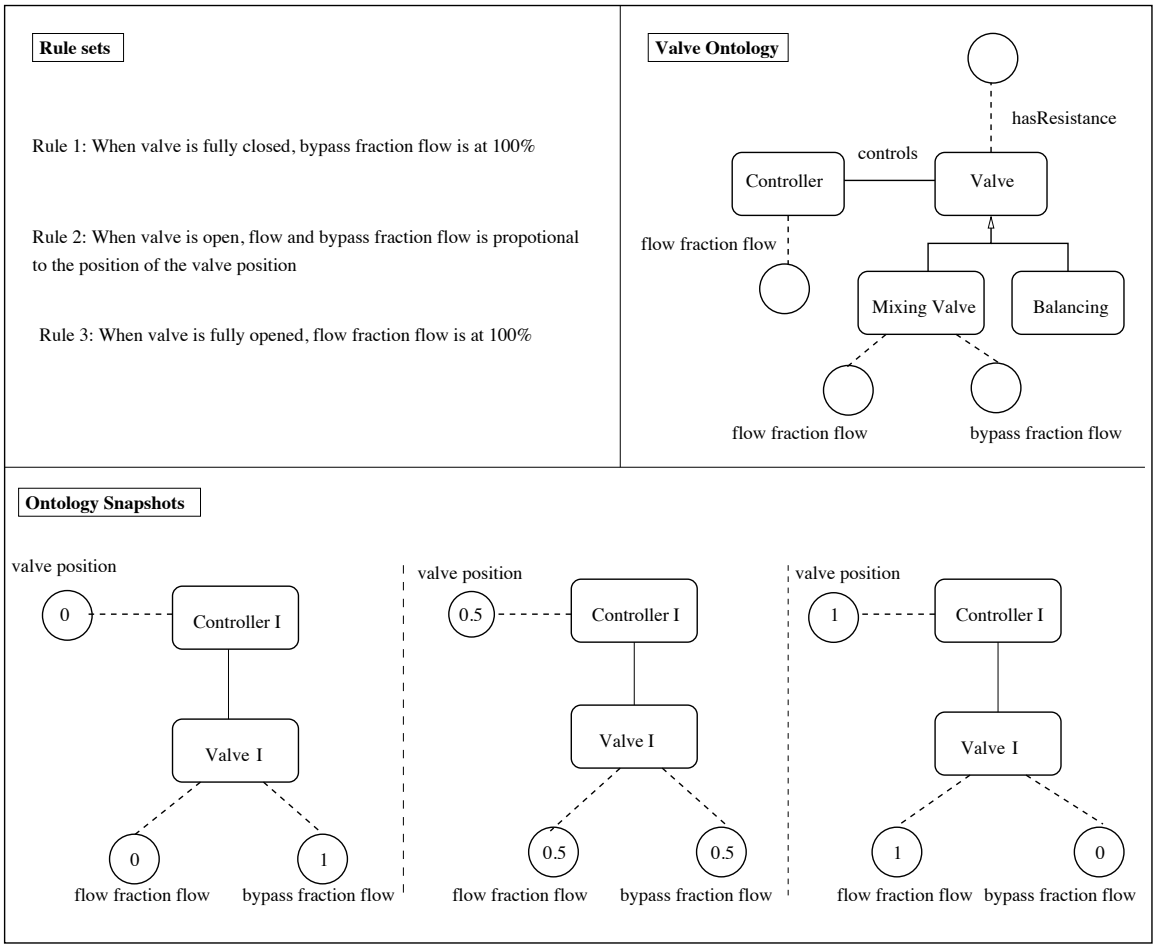


Figure 2.9: Schematic for a valve ontology and rules.

The upper right corner illustrates the system ontology that describes the interconnection between valves and valve controllers in HVAC systems. The upper left side shows some possible rule sets for directing the flow (through bypass or flow port) in a three way mixing valve. The bottom side of the figure shows the evolution

of a graph defining the properties of a valve (valve I) and a controller (controller 1) as a function of stem position.

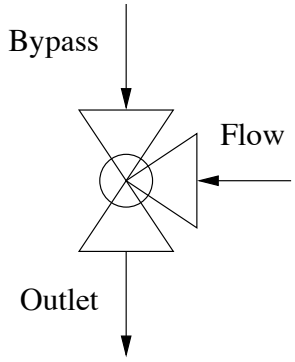


Figure 2.10: Schematic for a three-port valve.

A mixing valve is composed of a flow port, and bypass port and has properties: flow fraction flow and bypass fraction flow. These two properties range from 0-1. When the valve stem position is fully closed, the bypass fraction flow is at 100% or the bypass fraction flow property in the ontology is equal to 1. The same situation holds for flow port. However, it directs 100% of the flow when the valve is fully closed. Valve controllers can also be used to control mixing valves. If the stem position is at 0 (valve closed), flow will be going through the bypass port. If the valve stem position is 1 (valve fully opened), flow will go through flow port. Any fractions in between, will impose both ports to stay partially open. The following is the code snippet of Jena rules to mimic the described three port valve behavior.

```
@prefix val:  <http://austin.org/valve#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[ ValveClosed: (?x rdf:type val:Controller) (?x val:ValvePosition ?y)
               (?x val:controls ?v) equal(?y,0) ->
               (?v val:fractionFlow 0) (?v val:bypassFlow 1)]
```

```
[ ValveOpened: (?x rdf:type val:Controller) (?x val:ValvePosition ?y)
               (?x val:controls ?v) equal(?y,1) ->
               (?v val:fractionFlow 1) (?v val:bypassFlow 0)]
```

```
// Define classes ...

valve  = model.createClass( ns + "Valve");
mixing = model.createClass( ns + "Mixing");
balancing = model.createClass( ns + "Balancing");

// Define relationships among classes ...

valve.addSubClass (  mixing  );
valve.addSubClass (  balancing );

// Create data properties for the class Person ...

hasValvePosition = model.createDatatypeProperty( ns + "valvePosition");
hasValvePosition.setDomain(controller);
hasValvePosition.setRange( XSD.double );

hasBypassFlow = model.createDatatypeProperty( ns + "hasBypassFlow");
hasBypassFlow.setDomain(valve);
hasBypassFlow.setRange( XSD.double );
```

2.5.4 Semantic Modeling in Intelligent Buildings

Semantic Web technologies are maturing fields and continue to be appealing in many new application domains from next generation health care [48] and biology [116], to transportation [34] intelligent systems. The central idea behind Semantic Web is to enhance data on the World Wide Web by so-called metadata, which describes the meaning (semantics) of the data. This feature is the key element in processing data in intelligent systems.

Due to advancements in sensor capabilities versus their cost and data collection systems in buildings, there is too much data with too little structure for decision-making units to consume in the buildings. In state-of-the-art building con-

trol strategies, rule-based or model-based, decisions are made based on numeric value, content, of a system attribute e.g., pressure in a duct, temperature in a room. An integral element to future intelligent buildings is the capability utilize semantic information regarding the building and its surrounding domains such as, occupants, weather, and utility. This feature will empower the control logic to make decisions based on the semantic and not just numeric values. As a case, instead of responding to the room temperature value being greater than a threshold of 28°C, the controller will react when a room temperature is "warmer" than a target temperature. Ontologies and rules can provide the semantic info-structure required to achieve the following goals in intelligent building in the following ares:

- Semantic inquires of the system state,
- Reasoning capabilities to infer new knowledge from existing data,
- Intelligent information integration from heterogeneous domains.

Achieving these goals will provide mechanisms for knowledge-based control, automatic fault detection and diagnostics, building code compliance checking and utility tariff description in smart grid applications in the next generation of smart buildings.

2.6 Data-Driven Approach to Generation of Individuals in Semantic Graphs

In the proposed framework semantic models are the composition of ontologies, rules and data.

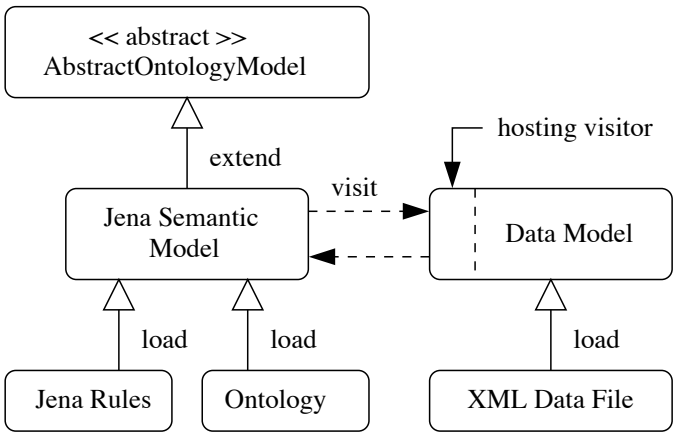


Figure 2.11: Data-driven approach to generation of individuals in semantic graphs.

Figure 2.11 illustrates a data-driven approach to the generation of individuals in semantic graphs. First, data is imported into Java Object data models using JAXB, the XML binding for Java. After the ontologies and rules have been loaded into the Jena Semantic Model, the semantic model creates instances of the relevant OWL ontologies by visiting the data model and gathering information on the individuals within a particular domain (e.g., building, sensor, occupant). Once the data has been transferred to the Jena Semantic Model and used to create an ontology instance, the rules are applied.

Chapter 3: State-of-the-Art Engineering for HVAC Analysis and Design

This chapter discusses state-of-the-art approaches to behavior modeling, simulation and co-simulation, and strategies of control for HVAC system operations. The principal objective in presenting this information is to lay the groundwork for studies that involve the integration and co-simulation of model predictive control (MPC) with Modelica, all running on the BCVTB (Building Controls Virtual Test-Bed) environment. Looking forward, this framework can be viewed as step toward an operating systems view of HVAC behaviors (see Figure 3.1), and control of operations in response to environmental processes.

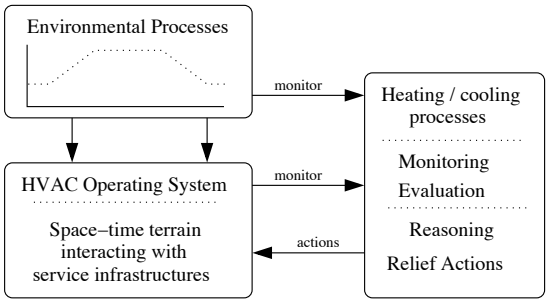


Figure 3.1: Operating systems view of HVAC behaviors, and control of operations in response to environmental processes.

The second objective of this chapter is to review state-of-the-art capabilities in fault detection and diagnostic analysis, the heart of Case Study 3 presented in

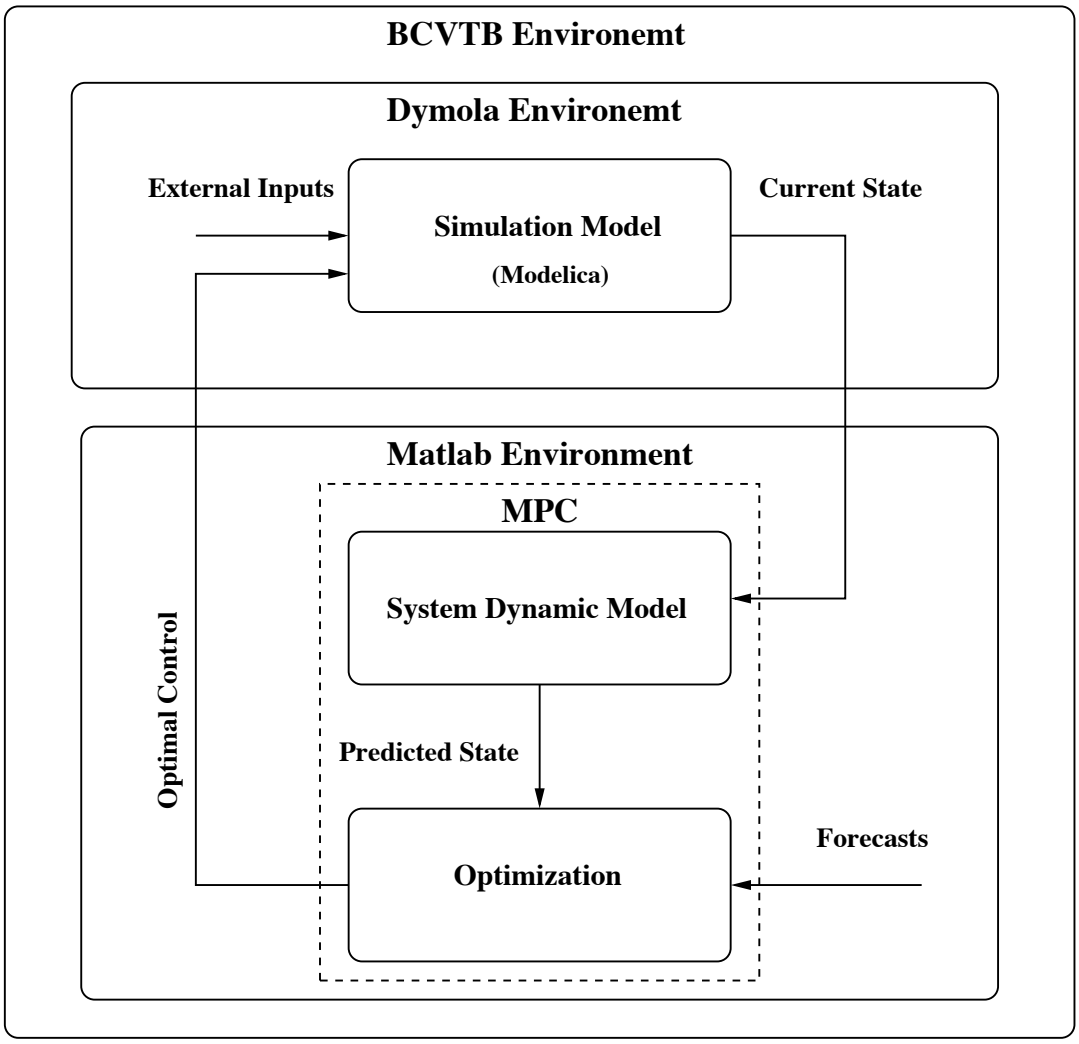


Figure 3.2: Co-simulation framework for integration of model predictive control (MPC) with physical systems simulation with Modelica (Dymola).

Chapter 5. It is noted that this dissertation is not the first attempt at introducing modern concepts of computer science to the field of building performance simulations – in fact, a number of researchers [31, 108] report that others have tried, and failed because the results were not perceived as being useful.

Section 3.1 introduces models of computation (MoC) for the development of HVAC analysis procedures. Sections 3.2 describes state-of-the-art capabilities in building performance simulation. The various tools are organized into three categories: (1) procedural-based tools, (2) equation and object-based tools, and (3) actor-based tools. Section 3.4 explains the different levels of control for HVAC system operations, and reviews the benefits of using MPC in control. Finally, related work in procedures for fault detection and diagnostic analysis for HVAC is presented in Section 3.5.

3.1 Models of Computation for Behavior Modeling

3.1.1 Introduction to Models of Computation

A model of computation is an abstract, but formal, specification of how a computation can progress. Such a specification will include the class of functions that can be computed, the associated cost (or measure of complexity in terms of time and required memory) of computation, and ease with which an algorithm or solution procedure may be expressed. Theoretical studies on models of computation are important because they create a pathway for analysis of required resources and/or an assessment of the limitations of a computer.

Models of computation are relevant to this study because modern HVAC systems are also cyber-physical systems. Challenges in the analysis of cyber-physical systems can be traced to: (1) fundamental differences in the way physical and cyber systems work, (2) the difficulty in modeling interactions (or coupling) between the cyber and physical elements of a system, and (3) the lack of mathematical framework for dealing with cyber-physical concerns in a unified way. On the physical side of the problem, continuous behaviors can be described by differential equations. When the physical system properties are not known precisely, notions of functionality and performance are not assured, and instead need to be quantified in terms of reliabilities. A physical system will usually provide some kind of warning before failure. Cyber systems, on the other hand, have behaviors that are dominated by logic (discrete), and a tiny logical errors in a cyber system can sometimes trigger system-level failures that occur without warning.

In practice, time-dependent environmental processes, such as air loops and water loops will be modeled on the physical side of the problem, and control algorithms (with ontologies and rule sets) on the cyber side of the problem. An expedient pathway forward is to decompose the space of analysis problems into a network of computations, with each node addressing an aspect of the computation in a manner that is most convenient for the specific problem at hand. Figure 3.3 shows, for example, a small network of models of computation. Models of computation can potentially include finite state machines (FSMs), communicating finite state machines (CFSMs), continuous time (ODEs), continuous time over spatial dimensions (PDEs), discrete event systems, data flow models and signal models. The physical

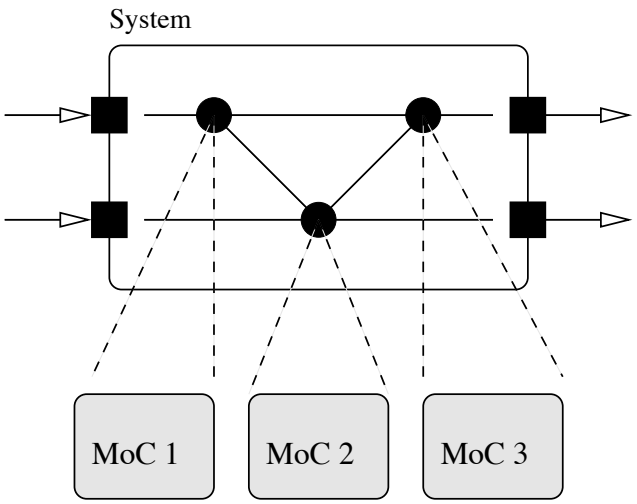


Figure 3.3: Different parts of a system are modeled with different computational models. The separate models of computation are integrated into a single framework. Models of computation can potentially include finite state machines (FSMs), communicating finite state machines (CFSMs), continuous time (ODEs), continuous time over spatial dimensions (PDEs), discrete event systems, data flow models and signal models.

side of the problem will be modeled as networks of physically connected objects. Objects at the component level will be assembled into networks of objects at the system level. Numerical procedures will compute the system state (e.g., distributions of mass flows and pressure) and time-history behavior in response to daily variations in temperature.

3.1.2 Five Approaches to System/Model Development

Figure 3.4 shows five approaches to system/model development: (1) causal modeling, (2) acausal modeling, (3) object-oriented, (4) equation-based, and (5) actor-based [83]. For each of these modeling abstractions, the key characteristics are as follows:

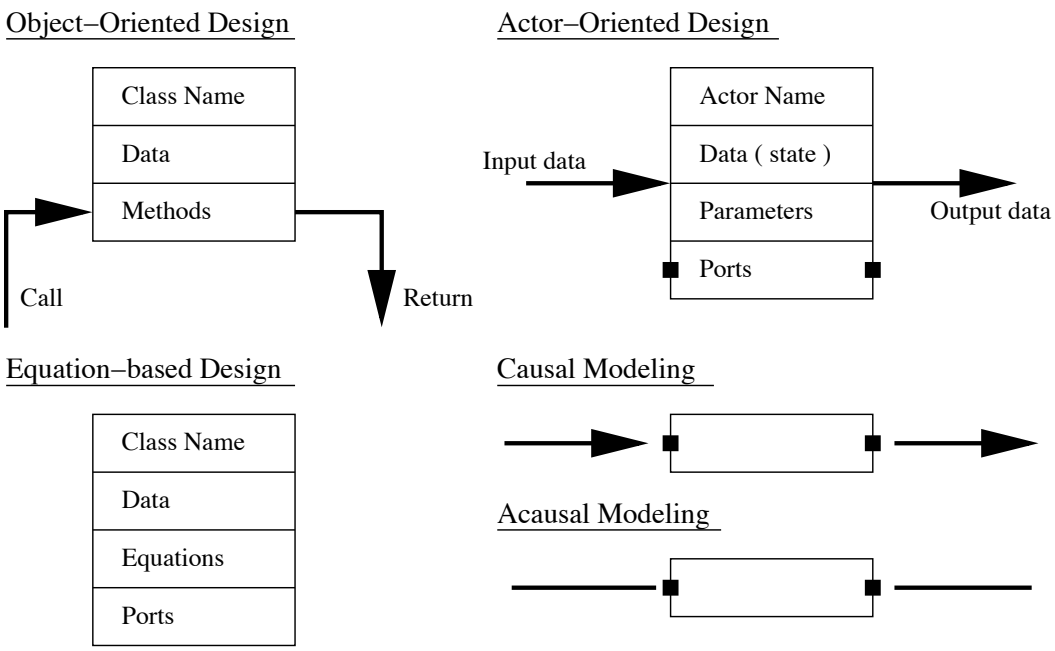


Figure 3.4: Five approaches to system/model development: (1) object-oriented, (2) actor-based, (3) equation-based, (4) causal modeling, and (5) acausal modeling (Adapted from Lee [83]).

1. **Causal Modeling.** A causal model is an abstract model that describes the causal mechanisms of a system, for example, the direction of signal/data or fluid flow throughout a networked system. Causal modeling techniques are often packaged as networks of computational blocks, with each block having ports and pre-defined notions of input and output. System components are linked through their input-output variables to form an HVAC system (or sub-system) assembly, and in such a way there exists a computational pathway to a feasible solution.

2. **Acausal Modeling.** Acausal modeling is a declarative modeling style where modeling consists of the specification of equations instead of (cause-and-effect type) assignments. Acausal approaches to modeling are ideal for system be-

haviors governed by the solution to physics equations. Most finite element and structural analysis problem solving procedures are acausal.

3. **Object-Oriented Modeling.** Object-oriented modeling procedures are concerned with the capture and processing of knowledge and data through the definition of classes, relationships among classes, mechanisms for data storage, and definition of methods to support computation on objects.

4. **Equation-based Modeling.** Equation-based modeling techniques describe a system in terms of differential-algebraic equations. Individual components are represented by a set of equations and their corresponding variables. Components may have continuous behavior or discrete behavior over time.

5. **Actor-based Modeling.** Actors operate as concurrent processes each having their own thread of execution, and will respond to streams of incoming data (e.g., a controller will respond to streams of data from sensors).

3.1.3 Co-Simulation: Coordination and Synchronization of Subsystem-Level Processes

With a steady trend toward engineering systems becoming progressively large and complex now in place, it is important that we search for new ways to keep designers productive.

One solution to this challenge is to adopt a decompositional approach to modeling of heterogeneous systems, where different parts of a problem – subsystem-level

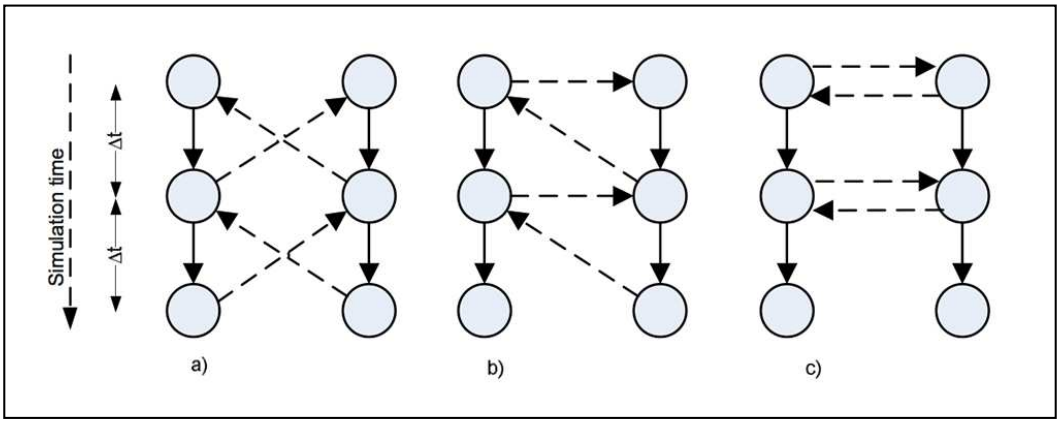


Figure 3.5: Three strategies of data exchange in energy co-simulation: (1) strong coupling of time-dependent states, (2) loose coupling with sequential simulator execution, and (3) loose coupling with parallel simulator execution.

or discipline-specific concerns – will be distributed and computed in a manner that is most convenient to their needs. The last step is to combine the results, which imply the need for computational infrastructure that is coupled, and can coordinate and synchronize the underlying concurrent processes.

Figure 3.5 shows three strategies of data exchange in co-simulation: (1) strong coupling of time-dependent states, (2) loose coupling with sequential simulator execution, and (3) loose coupling with parallel simulator execution.

3.2 State-of-the-Art Tools for Building Performance Simulation

Computational tools for simulation of HVAC behaviors and performance have been in development since the 1960s. Over the years, these tools have evolved to take advantage for new capabilities in computer hardware (e.g., available storage, processing speed) and computer science (e.g., languages, interpreters and compilers, operating systems) as they come along. From a computational standpoint, state-of-

the-art tools for building performance simulation can be classified as being either: (1) procedural-based, (2) equation- and object- based, or (3) Actor-based.

3.2.1 Procedural-Based Tools

A partial list is as follows [3]:

1. TRNSYS is a component-based tool for transient simulation of a building and assorted energy systems [35].

2. EnergyPlus is computational tool for whole building energy simulation as well as load calculations [36].

3. HVACSIM+ is a component-based tool designed to support dynamic simulation of HVAC systems, selection of components, controls and sizings [29].

4. At the system level, the engineering equation solver (EES) numerically solves systems of linear and non-linear algebraic and ordinary differential equations. EES is popular within the HVAC simulation community because it provides features for solving equations that involve thermodynamic transport.

While the majority of these tools aim to simulate building energy consumptions and entail various capabilities for the building energy simulations [35], few of them are targeted toward real-time performance assessment of HVAC systems.

Both TRNSYS and HVACSIM+ assume that component-level behaviors can be defined by causal models of behavior. System-level behaviors correspond to a

time-history of pressure-driven flows through a network of connected components. At each step in a time-history analysis, Newton-like solution strategies [113] are used to determine distributions of pressure (and corresponding pipe flows) throughout the system.

3.2.2 Equation- and Object-based Tools

A partial list of equation- and object-based simulation tools is as follows [3]:

4. At the component level, EES provides support for explicit representation of equations.

5. Modelica [55] is a declarative language for the specification of equations and mathematical models that allow acausal modeling. Compared to conventional black-box approaches to component modeling and simulation, equation-based methods provide engineers with a transparent means to understand the structure of the equations that need to be solved. The modeler can define equations symbolically, compute derivatives in numeric, symbolic, and automated fashion. The Modelica language is also object oriented; as such, it provides support to the organization of classes in to hierarchies, and the development of reusable software packages. Further efficiencies are provided by graph-theoretic algorithms to reduce the complexity of system of equations, and by built-in runtime tools to automatically generate and compile sets of equations into code for simulation computations.

The Modelica Buildings Library. The Modelica Buildings Library is a free, open source, tool for modeling building energy systems with equation-based component modeling [137]. The library is still a work-in-progress; on-going work includes, for example, the addition of new features to account for computational fluid dynamics or CONTAM airflow simulations [143].

Remark. There have been a number of studies aimed at comparing the performance of equation-based modeling with procedural modeling. Wetter and Haugstetter [135] report, for example, that the required development time for Modelica is less than that needed for implementation of a model with similar physics in C/C++. Thus, considering its strong language features and great simulation performance, this dissertation utilizes the Modelica language and the Dymola environment in the development of the case study applications.

3.2.3 Actor-based Tools

A partial list of actor-based simulation tools is as follows:

6. Ptolemy II [19] is a design platform for the component-based modeling and design of embedded systems. Models are constructed as a set of interacting components, with models of computation governing the semantics of component interaction.

7. The Building Controls Virtual Test Bed (BCVTB) acts as a middleware to manage the data exchange between different simulators, with each simulator

implemented as an actor [136]. It provides mechanisms to link to building automation systems (BAS) through the BACnet protocol, and couple simulation models that can be encapsulated in functional mock-up units (FMU) with other simulators. Building performance simulation (BPS) programs such as Energy Plus, Dymola, MATLAB and Simulink can be deployed as actors. It is based on Ptolemy II [19] and is implemented in Java. We refer the interested reader to Wetter [134] for an in-depth summary of BCVTB capabilities and the mathematics behind the data exchange during co-simulation.

BCVTB Related Work The BCVTB has been employed in a number of research studies. For example, Kwak and colleagues [82] used the BCVTB framework acted as middleware among a weather forecast program, EnergyPlus, and MPC-enabled computations for prediction of energy consumption. They used real-time building energy simulation models and daily updated weather forecasts to compute energy consumption by MPC algorithms and compared the results with measured energy consumption values. The results of this comparison were within the allowable statistical error range. Sagerschnig [107] utilized BCVTB and simulation to asses the performance results of two control strategies – rule-based and MPC – for a real building in Munich. Lastly, in a multi-domain investigation of energy consumption and the need to provide a comfortable temperature range for workers in production plants, Hafner et al. [61] utilized BCVTB for co-simulation between Dymola (for building energy simulation) and Matlab and Simulink (for control studies).

The Functional Mock-Up Interface (FMI). The functional mock-up interface

(FMI) [53] is an open-source standard designed to support exchange of dynamic models and enable links between disparate simulation programs. Each participant of the co-simulation is required to be described as a Functional Mock-up Unit (FMU), composed of of xml-files and compiled C-code, and libraries. The FMI standard has been adapted in many application cases in HVAC systems. For example, Nouidui and co-workers [96] have used FMI technique to couple EnergyPlus with other simulation programs. Nicolai [95] tested the application of the FMI co-simulation between detailed physical building models coupled to Modelica-based HVAC component and plant models.

3.3 BCVTB Software Architecture with MoC Annotations

Figure 3.6 is the same as Figure 3.2, except for annotations highlighting the models of computation used in various parts of the BCVTB architecture integrated with simulation capabilities (in Modelica) and model predictive control (implemented in Matlab).

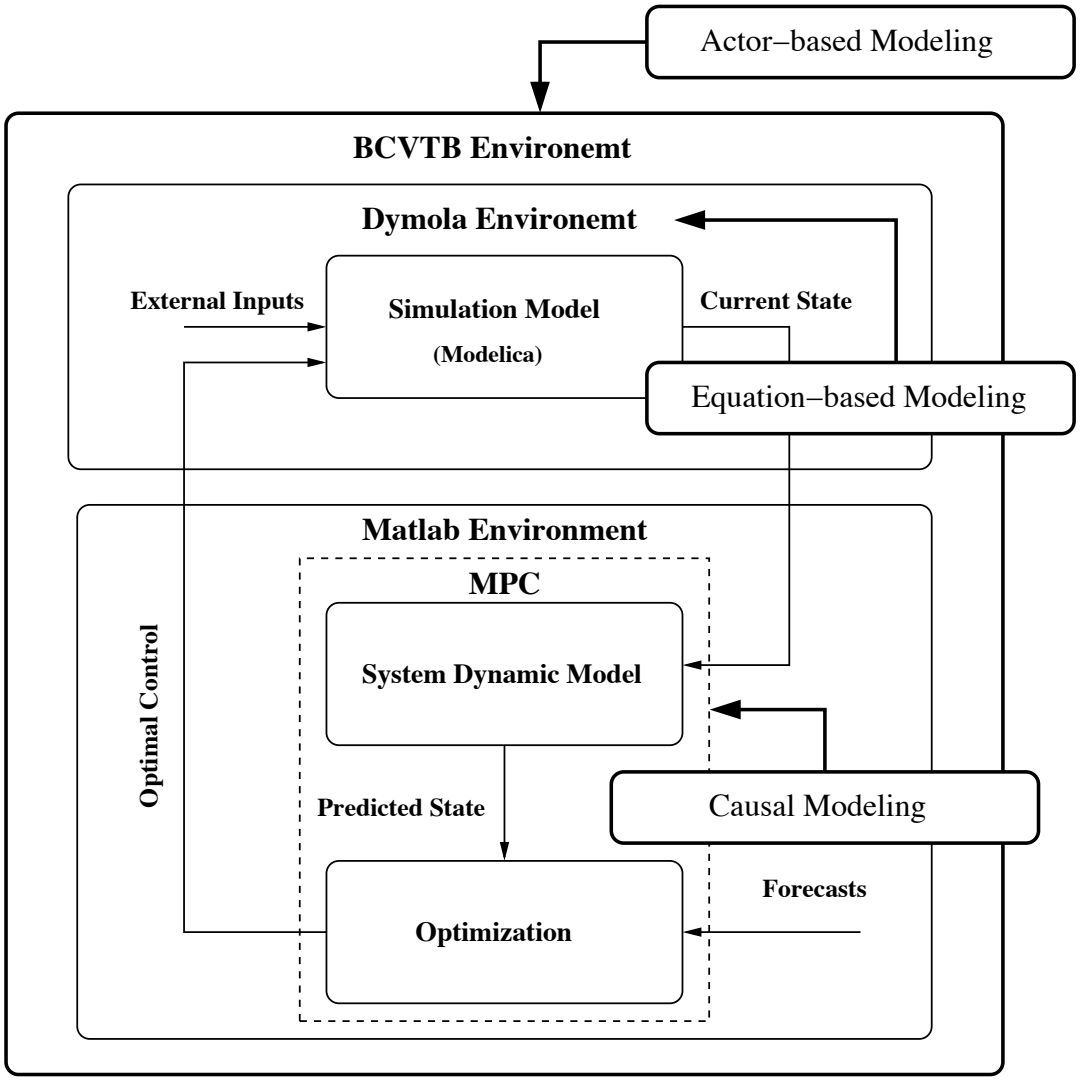


Figure 3.6: Annotated co-simulation framework for integration of model predictive control (MPC) with physical systems simulation with Modelica (Dymola).

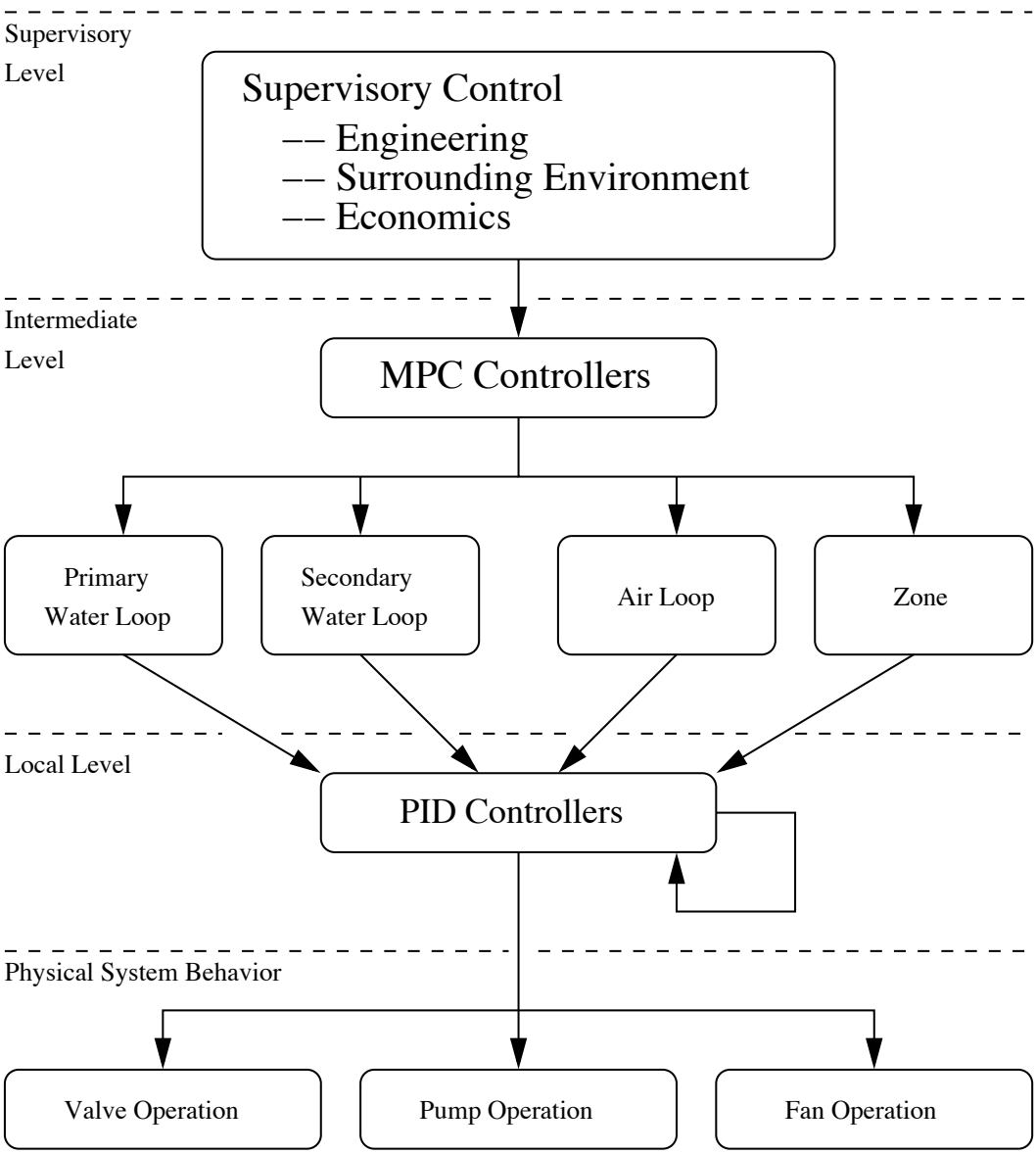


Figure 3.7: Multi-level control structure for HVAC systems.

3.4 State-of-the-Art Control for HVAC

During the last two decades, many researchers in field of HVAC have devoted considerable effort to the development and application of proper control methods. From a modeling perspective, controllers are represented by series of equations or set of formal logical rules that must be satisfied in every simulation step.

Solutions to this problem are complicated by the large scale of networks (which need to be controlled) and the prevalence of distributed systems. To deal with this, HVAC controllers can be divided into two categories as follows:

Supervisory Controllers. High-level supervisory controllers constrain the behavior of the uncontrolled system. That is, to put restriction of the behavior of the plant. They consider the system level characteristics and interactions among all components in the system and their associated variables. The operation of HVAC systems under supervisory control has been studied in [62,69,71,84,94,128,129,142].

Local Controllers. Low-level local controllers that allow HVAC systems to operate properly and to provide adequate services. Local controllers can be divided into [130]:

- Sequencing controllers define the order and conditions associated with switching equipment ON and OFF. As a case, pump sequencing controller, fan sequencing controller are examples of such category.

- Process controllers adjust the control variables to meet the required set point considering the system dynamic characteristics. The typical process controllers

used in the HVAC field are P, P-D, P-I, P-I-D controllers.

3.4.1 Supervisory Control Strategies

An overall classification of main supervisory control methods used in HVAC systems is illustrated in [130]. According to this work, supervisory control in HVAC systems could be classified into at least three categories, including model-free supervisory, model-based supervisory, or hybrid supervisory control method. It is important to note that such a classification may not be the best approach since there are no clear boundaries among some control methods. The term model-based method or a model-free method depends on whether or not numerical models are used.

Shangwei [130] classifies the control methods using physical models, gray-box models, and black-box models into the category of model-based methods. The model-free supervisory control methods do not require a mathematical model of the targeted system. Expert systems, or pure learning approaches can be grouped into the model-free category. The selection of the control methods for a supervisory control application plays a critical role in the development of an effective control strategy to the optimal operation of HVAC systems.

3.4.2 Model Predictive Control

Model-predictive control (MPC) is a method for constrained optimal control, which originally came from process industries such as oil refineries and chemical

plants, in the late seventies and early eighties. Today, it covers a wide range of applications from setpoint tracking in buildings control to trajectory tracking in autonomous vehicle and cost reduction in economical systems. MPC deals with modeling processes to optimize the control signals based on predicting how the signals will evolve in future. The main principle of MPC is to compute a control signal to minimize an objective function, which is usually a function of the system states. The state space control model is then used as a constraint in the optimization problem. The MPC algorithm is usually set up to compute the optimal control signal over some period of time (steps in the future), but only the control signal at the first time step is applied before the optimization is resolved and a new control signal is generated. An application of MPC can be found in regulating the temperature of a space by controlling the temperature of the supply air from HVAC. Since HVAC systems tend to respond very slowly, MPC algorithms can be utilized to predict the next step system behavior and apply the appropriate control inputs in advance.

One of the reasons MPC is gaining traction in the building systems research community, is due to the superior levels of performance of MPCs in optimizing building climate control while satisfying occupant thermal comfort. These emerging approaches minimize an objective function and employ dynamic models of system behavior in the evaluation of problem constraints.

Model predictive control (MPC) techniques are designed to optimize the performance of nonlinear systems (e.g., HVAC), and can easily deal with multi-input multi-output systems (e.g., multiple control variables, CV), and system constraints and nonlinearities in an intuitive way [125]. A second important aspect of using

MPC in building control stems from the method's ability to implement the satisfaction of occupant thermal comfort as a constraint. One approach to maintaining occupant comfort is to maintain the temperature within an acceptable band based on general comfort guidelines [28]. A second approach is to incorporate simplified mathematical models for thermal comfort within the control algorithms. Predicted mean vote (PMV) stands among the most recognized static thermal comfort models and predicts a mean value for the vote of a large group based on the heat balance of the human body. This index is computed based on two personal (e.g., metabolic rate) and four environmental (e.g., air temperature) factors.

In economic MPC, the goal is to optimize a user defined cost function subject to system constraints, which include simplified models of the physical behavior. The result of the optimization is a sequence of control actions for the time horizon. These actions act as inputs to either the simplified model used in the optimization or more complex dynamic models of the system that have the means to update the state of the system for the optimization in the next iteration. MPC algorithms have been adopted in various domains from plant operation to building control. Chandan et al. [25] have utilized MPC in the optimization of operating cost for a combined cooling, heating and power (CCHP) plant. Their results indicate that use of MPC leads to levels of attainable performance that are 8.5% better than what is possible with rule-based control. Faruque et al. [50] have presented a co-simulation engine GridMat; the purpose of it is to co-simulate the power systems models as well as testing different control algorithms that are modeled in Simulink to optimize power consumption of houses in a residential micro-grid problem. Their

MPC techniques reduced the power consumption compared to the baseline control and direct load control (DLC). Kolokotsa et al. [80] have combined MPC control with a Building Energy Management System (BEMS). Optimal setpoints that satisfy indoor environmental quality constraints are computed by an MPC that takes a prediction of the indoor environment conditions as an input. This approach has been tested on a real building in Hania, Greece, with satisfactory results. A similar approach has been used by Privara [102] for MPC-enabled temperature control in a real building. Estimates of potential savings range from 17-29%.

During the past decade, there have been numerous studies [24,27,54] focusing on the interactions of building control and occupant thermal comfort. They mainly focus on developing control architectures that reduce energy consumption while accounting for models of thermal comfort (i.e., PMV, DTS). For example, Castilla et al. [24] have presented a hierarchical thermal comfort PMV based control where the upper layer includes a non-linear MPC and the lower layer is a PID controller, which is in charge of reaching the setpoints set by MPC. This control approach was tested in a typical bioclimatic room. The results show a 53% savings in energy when the proposed structure was used as compared to energy savings by a classical MPC. Freirea and co-investigators [54] have implemented PMV-based model predictive control in two case studies, with the results indicating that is it is possible to simultaneously maintain thermal comfort and reduce energy consumption [30]. Cigler et al. [28] used the PMV index in the MPC optimization formulation. Their results indicate that when optimizing the PMV index in MPC, the energy is reduced 10%-15%. Another approach is to identify dynamic thermal sensation (DTS) models for

occupant thermal comfort and express the thermal sensation as a function of time and indoor temperature. A DTS-based MPC was compared to a PMV-based MPC to control the temperature of a chamber [27]. The experimental results revealed that the DTS-based MPC using occupant feedback allowed for significant energy saving while maintaining occupant thermal comfort as compared to the PMV-based MPC. In DTS-based MPC, the room and thermal sensation models are the dynamic transient models utilized in MPC.

3.5 State-of-the-Art Procedures for Fault Detection and Diagnostic Analysis in HVAC Systems

Within the building sector, degraded or poorly-maintained equipment account for 15 to 30 % of energy consumption in commercial buildings [76]. Approximately 50 to 67 % of air conditioners (residential and commercial) are either improperly charged or have airflow issues [77,111]. Faulty heating, ventilating, air conditioning, and refrigeration (HVAC&R) systems contribute to 1.5 to 2.5 % of total commercial building consumption [140]. Much of this energy usage could be prevented by utilizing automated condition-based maintenance.

3.5.1 Automated Fault Detection and Diagnostics

Automated fault detection and diagnostic (FDD) techniques provide a means of detecting unwanted conditions (i.e., "faults") in systems by recognizing deviations in real-time or recorded data values from expected values, and then diagnosing the causes leading to the faults. FDD techniques provide mechanisms for condition-based maintenance of engineered systems (e.g., buildings, health monitoring, power plants and aviation systems). Proper implementation of FDD can enable pro-active identification and remediation of faults before they become significantly deleterious to the safety, security, or efficiency of the operating system.

During the last decade, considerable research has focused on the development of FDD methods for HVAC&R systems. This work has been driven, in part,

by the historically less-than-optimal operation of many state-of-the-art HVAC systems. Yet, in spite of recent advances in building simulation, automation and control (see the arrangement of ontologies, rules, reasoning and simulation software in Figure 1.4), automatic methods for FDD of building systems remain at a relatively immature stage of development. As a result, we require more advanced FDD techniques that leverage the untapped capabilities of building automations integrated with methods in artificial intelligence and semantic modeling. These interdisciplinary FDD systems can benefit from utilizing knowledge repositories for storing automation/simulation data and the inference-based reasoning techniques to obtain additional higher information, such as sensors location, equipment service area. State-of-the-art fault detection methods are equipment and domain specific and non-comprehensive. As a result, the applicability of these methods in different domains is very limited and they can achieve significant levels of performance by having knowledge of the domain and the ability to mimic human thinking in identifying the source of a fault with a comprehensive knowledge of the system and its surroundings.

3.5.2 Procedures for Fault Detection

3.5.3 Procedures for Diagnostic Analysis of Faults

Recent advances in building automation technologies provide a means for sensing and collecting the data needed for software applications to automatically detect and diagnose faults in buildings. During the past few decades a variety of FDD

techniques have been developed in different domains, including model-based, rule-based, knowledge-based, and simulation-based approaches. Katipamula and Brambley summarize FDD research for HVAC systems [76]. Their work also describes different fundamental FDD methods under the two main categories of model-based and empirical (history-based) approaches. The major difference is in the nature of the knowledge used to formulate the diagnostics. Model-based diagnostics evaluate residuals between actual system measurements and *a priori* models (e.g., first principle models). Data-driven empirical strategies, on the other hand, do not require *a priori* models. The models used in model-based methods can be quantitative or qualitative. Quantitative models represent the requisite *a priori* knowledge of the system in terms of mathematical equations, typically as explicit descriptions of the physics underlying system components. Qualitative models, conversely, combine concepts such as descriptive "states" and "rules" into statements that are axiological instead of mathematical, expressing operational correctness or desirability through an axiology, a value system, appropriate to each physical application. As a result, the building system operation can be continuously classified as being either faulty or not faulty.

Rule-based strategies are one example of qualitative model-based FDD methods. Rules can be based on first principles or they can be inferred from historical experiments, but in either case they represent expert qualitative knowledge that no purely quantitative representation could model. The first diagnostic expert systems for technical fault diagnosis were developed at MIT by Scherer and White [109]. Since then, diagnostic systems have evolved from rule-based to model-based and

expert systems approaches. Semantic models offer a means for the representation of distributed and explicit knowledge and provide ways through inference-based rules to derive implicit knowledge. Berners-Lee and co-workers [17] points out to the benefits of ontology usage for knowledge representation, and utilizing high-level reasoning capabilities in the area of agent-based control solutions. Exploitation of semantics and ontologies in the area of agent-based engineering systems has become one of the hot topics recently. The main reason behind this trend is the success and promotion of Semantic Web technologies to enable languages that are both machine and human processable. Semantic Web-based applications have been developed in the areas of health care [48], biology [85, 116], and transportation [34]. In the area of fault detection and diagnostics, Batic [13] has developed an ontology-based fault detection and diagnosis system and tested it on airport ontologies to detect the high level irregularities in the operation of airport heating/cooling plants. Also, Schumann [110] highlights the potential impacts of artificial intelligence techniques such as ontologies on tackling the challenges in obtaining a unified diagnosis framework. The benefit of this approach is that ontologies are an essential technology guaranteeing data and information interoperability in heterogeneous and content-rich environments [91] which is at the heart of comprehensive fault detection and diagnostic methods.

3.6 Summary

This chapter has highlighted state-of-the-art methods of analysis and tools for behavior modeling with various models of computation, simulation (and co-simulation), control, and fault detection and diagnostics.

Energy simulation software packages cover a wide range of complexity, from simple (Excel-spreadsheet) to tools for complex domain-specific analysis (e.g., finite element analysis of fluid-flows around complex geometries). Most of these software tools provide computational support for domain-specific tasks, and abstract from consideration other factors that might affect system performance. Based on the features of the modeling languages, equation-based, object-oriented languages such as Modelica and supporting tools like Dymola are powerful tools for modeling building systems.

It is now evident, however, that as the drive for energy reduction in buildings intensifies [2, 105], future buildings will need to move toward the use of control strategies [20, 21, 26, 73, 123] that result in superior levels of energy performance. State-of-the-art energy control strategies are incapable of handling the multi-domain complexities (and account for semantic representations and knowledge) and interacting domains. To address these challenges there is a need for new approaches to building simulation control that employ mixtures of formal and mathematical model-based control algorithms. One solution is to exploit Semantic Web technologies in the area of HVAC control.

A common problem with verification of control algorithms in building systems

prior to deployment in real buildings is the difficulty of software implementation and systems integration. Thus, moving forward, there is a strong need for: (1) new approaches to software engineering for building control system applications, (2) publicly available tools (or methodologies) for the study of building system behavior and control with MPC, and (3) modular, extensible tools that can easily handle a multiplicity of high- and low-level building model abstractions, and a method to evaluate the effectiveness of MPC with respect to these models.

Lastly, state-of-the-art fault detection methods are equipment and domain specific and non-comprehensive. As a result, the applicability of these methods in different domains is very limited and they can achieve significant levels of performance by having knowledge of the domain and the ability to mimic human thinking in identifying the source of a fault by using that comprehensive knowledge of the system and its surroundings.

Chapter 4: Knowledge Representation in Building Domain

The scope of building systems is large; the interacting domains span across different areas from building structure and topology, to internal and external environment conditions, utility, inhabitants, and mechanical equipment. Recent studies, such as [58], explore the use of formal ontologies as a way of specifying content-specific agreements for the sharing and reuse of knowledge. State-of-the-art building simulation systems, lack the ability to utilize the knowledge of the building and its surrounding domains. This knowledge can be explicit and represent the facts, or maybe implicit by reasoning through the facts to derive new information. This chapter explains the formal information models and knowledge structures that represents the underpinning knowledge bases of a building.

4.1 Introduction

State-of-the-art building control strategies make decisions based on numerical values obtained from physical domains. They tend to overlook the semantic knowledge and the essential information to the building energy such as occupant, weather, utility, equipment and building geometry. Ambient Intelligence (AmI) is an area that is gaining attention in the application of intelligent systems. In the

early 90s, with the emergence of so-called "ubiquitous computing," it was suggested that computer and electric systems should be integrated into a physical environment and behave in an intelligent, reasonable way based on their understanding of the domain [133]. In this context, modern building automation and simulation systems should utilize an interface to understand the semantics behind the building static information and the dynamic and evolving characteristics, and to make the appropriate preliminary decisions based upon them. As a case in point, it has been shown by Braun [**?**] that in chilled water plants, storage-priority control provided near-optimal performance when there were significant differentials between on-peak and off-peak energy charges. However, without time-of-use (TOU) energy charges, chiller-priority performed better. In this case, having the knowledge of utility domain will impact the control strategy. To achieve this goal, this chapter builds upon the foundation introduced in Chapter 2 and proposes how Semantic Web technologies will act as a layer of abstraction in representing the semantic knowledge of the underpinning and surrounding domains for buildings. In this software infrastructure, ontologies are the semantic models that represent the key concepts of each domain, along with their properties and their interconnections. Moreover, inference-based rules defined on these concepts and reasoning capabilities provide mechanisms on deriving new information based on the existing data stored in the ontologies.

Figure 4.1 depicts how domain specific ontologies utilize the concepts defined in foundational, meta domain, ontologies of time and space along with spatial and temporal reasoning. As a case, the "Room" concept in Building ontology utilizes "Geometry" concept in Spatial ontology.

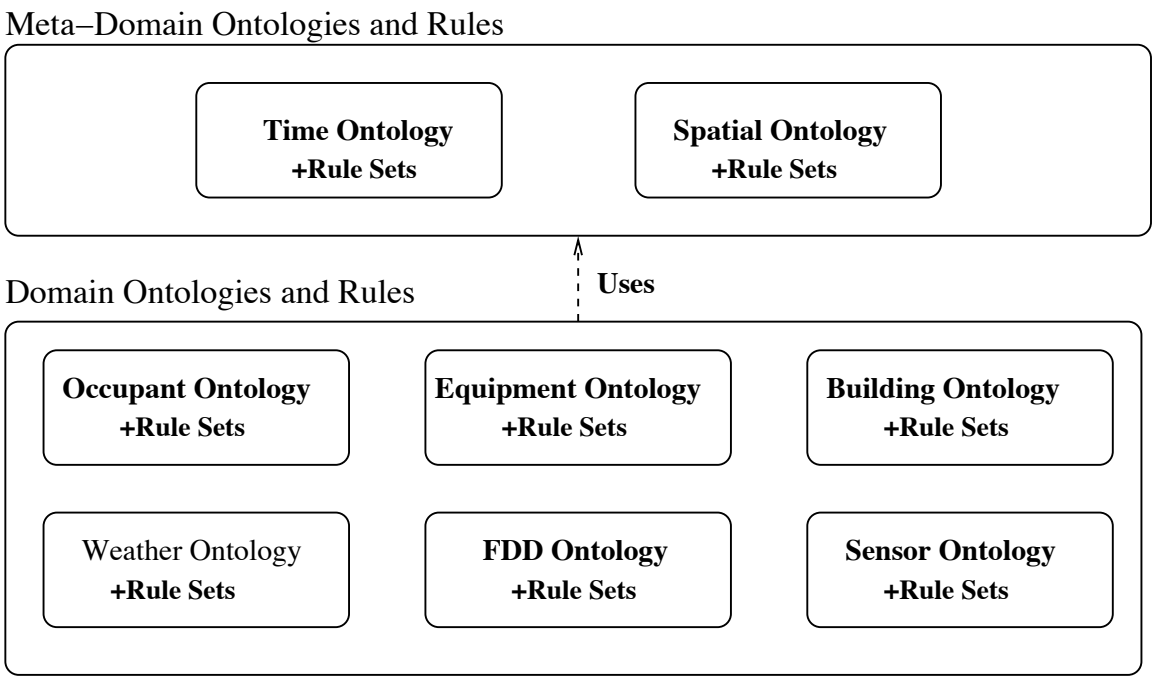


Figure 4.1: Domain specific and domain independent ontologies

The domain-specific ontologies and rules are organized into three groups: (1) engineering ontologies and rules, (2) surrounding environment ontologies and rules, and (3) economic ontologies and rules. In Figure 4.7 red rectangles with heavy dashed edges are used to highlight the important classes that participate in the presented rules.

4.2 Previous Work

While technological advances have been made to modernize building simulation and control frameworks, such solutions rarely experience widespread adoption due to the lack of a generic descriptive model that would ease the deployment of such frameworks in different buildings with minor adoption cost. Recent attempts

have sought to address this issue through data standards and metadata schemes. As a case in point, the Brick project [12] is an effort to achieve that goal. Brick provides uniform schema for representing metadata in buildings. It provides core ontology defining the fundamental concepts and their relationships. This semantic infrastructure was tested using six buildings Building Management Systems (BMSs) with sensors and equipment from different vendors. They were tested against different criteria of completeness, representing all the metadata information (such as a sensors location, type, etc. contained in a buildings BMS), expressiveness, capturing all important relationships between data points that are explicitly or implicitly mentioned in a buildings BMS. Other research projects have also, investigated and developed ontology based approaches to the building automation domain [139] have used ontologies as the generic application model facilitating an integration of heterogeneous building automation networks. They combined classical data-driven energy analysis with novel knowledge-driven energy analysis that is supported by ontology and rulesets. The analysis is performed on information collected from building automation devices and inference of an energy waste based on the state of those devices and the user behavior. Valiente [124] utilizes a semantic framework, IntelliDomo, to represent ambient intelligence. IntelliDomo allows managing the control of the building automation system itself. The state of the components that comprise a determined domotic installation is continuously obtained from the database where its values are stored and translated into instances of OntoDomo ontology. With this information, together with the SWRL rules defined by the user, IntelliDomos inference engine would fire the appropriate rules that will change the state of the system

devices. [47] focuses on the ontology development process to deliver an intelligent multi-agent software framework (OntoFM) supporting real time building monitoring. Their framework is comprised of, interrelated ontologies, including building ontology, sensor ontology and supporting ontology (mereology and topology ontologies) with the purpose of supporting the real time knowledge query of the underlying multi-agent framework. Mahdavi [88] introduced an ontology and associated data models for the representation and incorporation of multiple layers of data pertaining to inhabitants, indoor and outdoor environmental conditions, control systems and devices, equipment, and energy flows. This richly structured data representation will facilitate the collection, storage, sharing and analyses of monitored data in different applications including building automation, facility management, building diagnostics, and building performance simulation. Han [63] proposed BMS system architecture that is based on ontology and inference engine. It gathers various sensors data and equipment state data to decide the status of the building and to output control commands. Corry [33] proposed an ontology that receives data from building objects, sensors and simulation models and assessed that data in a structured way. That is, to use the ontology as a repository, or data integration tool. Han [64] used a rule-based ontology reasoning for context-aware building management to reduce energy waste. They use Jena Rules for reasoning purposes in context and policy. Moreover, the framework has been tested for a real office to estimate the effect of energy saving measures. Furthermore, energy simulation was performed with and without the rule-based ontology system. The results were more promising regarding lower energy waste when a rule-based ontology approach was used. Han [65]

utilizes ontology and inference rule sets for smart home control of appliances. Jena API was used to develop the ontology framework and the inference rule sets. Terka and co-workers [117] explain the conversion of an EXPRESS schema representing Industry Foundation Classes (IFC) into an OWL ontology. IFC is the standard used for BIM. Beetz [16] developed a converter to transform any format using an EXPRESS schema, like IFC to RDF. Baumgartel [14] study an optimization framework for green building design. They used the converted RDF from BIM models and provided input to the simulation model based on the values from the ontology. In above-mentioned literature, ontologies have been used as either a unified data-model and they dont leverage the untapped potentials of reasoning and inference offered by semantic web technologies. These capabilities should be used alongside the advanced monitoring and control techniques that operate based on mere digital values.

Our goal in building automation and control is to emulate human thinking and inferencing processes. For our purposes, this can be interpreted as event-driven decision making and control with a semantic description of domains and associated rules. To achieve this goal, it is necessary to have network/Web access and awareness of the environmental and building system state and formal systems for inferencing processes. This chapter elaborates how Semantic Web technologies can play a pivotal role in achieving this goal.

4.3 Meta-Domain Ontologies and Rules

In systems analysis, a meta model defines the languages (semantics) and processes (structure and constraints) from which models can be formed. The meta model for SysML [87] defines, for example, more than 250 entities from which SysML diagrams can be constructed. The semantic modeling counterpart of software engineering meta-models is meta-domain ontologies and rules that have universal application to the implementation of targeted domain models. Sometimes the name fundamental is used instead of meta-domain. In either case, semantic descriptions of time, space, physical units and currency can all be thought of as essential elements for describing how our world actually works.

This research employs temporal reasoning in computations to define electricity tariffs and applicable rates for specific intervals of time. And it uses spatial reasoning to determine the relationship of sensor and occupants to geometric entities such as rooms and building zones.

4.3.1 Temporal Ontology and Rules

Temporal Theories. Hayes [66] identifies six main concepts of time. Among this set, four are selected to support ontological representations of time in engineering. They are:

1. Time-interval: Pieces of time located on the temporal continuum serve as the basis for the temporal theory

2. Time-duration: A constant amount of time.

3. Time-point: The notion of a point in time supports this temporal theory; Sometimes this concept is assimilated to a "position in temporal coordinate system" which has no duration, but is useful in locating an event.

4. Time-dimension: Time is considered a physical dimension such as length, mass or voltage, with unit and physical properties.

Existing ontologies of time employ a combination of these four concepts, but are otherwise strongly influenced by the targeted need for which they were developed. In OWL-Time [118], the time ontology based upon the Web Ontology Language (OWL), Instant and Interval serve as foundational temporal entities.

Allen's Interval Algebra. Allen's temporal interval calculus [5, 6] identifies thirteen relationships between any ordered pair of convex time intervals.

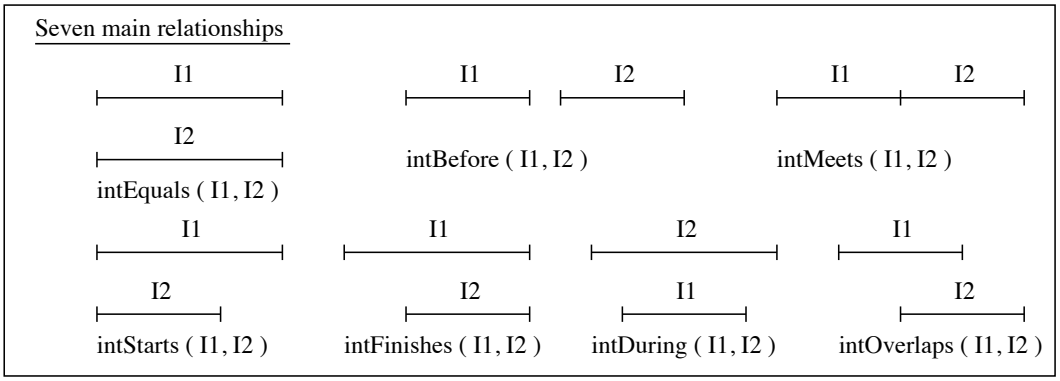


Figure 4.2: Schematic of Allen's temporal intervals. [101]

The seven main relationships are illustrated in Figure 4.2. Six inverse relations also exist.

Given two time intervals I_1 and I_2, a time-point t and a proposition ϕ, we can ask a variety of questions about the time domain, such as:

1. Does t occur within I_1?,

2. Is the interval I_1 equals to I_2?,

3. What interval represents the temporal intersection of I_1 and I_2?,

4. Does interval I_1 contains interval I_2?,

5. Does interval I_1 occur before or after interval I_2?

6. Do intervals I_1 and I_2 meet?

7. Do intervals I_1 and I_2 start and/or end at the same instants?

Logical questions include:

1. Does the proposition ϕ hold within the interval I_1?, and

2. If ϕ holds during the interval I_1, does it hold during I_2 too? Does the proposition ϕ hold before or after the interval I_1?.

Prototype Temporal Ontology and Rules. Figures 4.3 and 4.4 illustrate the classes and properties in the prototype time ontology, and sample rules for computing the relationship between intervals of time.

The time ontology is defined by four classes; TemporalEntity, OpenTimeInterval, Instant and ProperTimeInterval. The classes OpenTimeInterval, Instant and

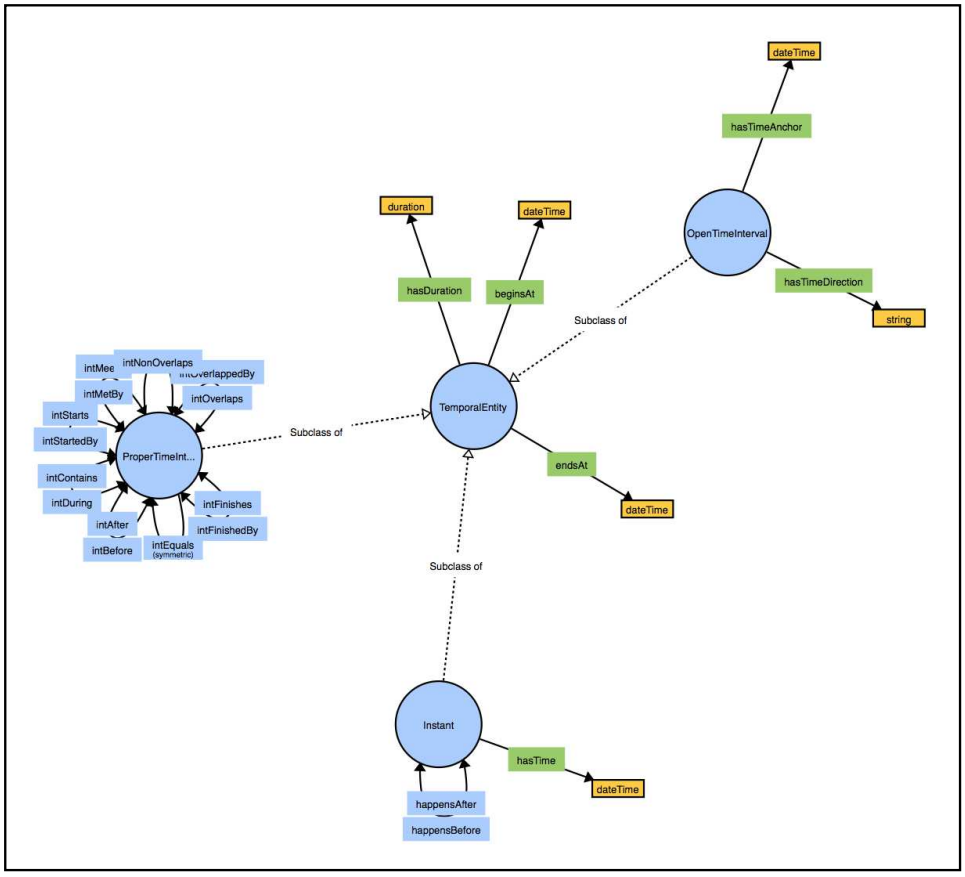


Figure 4.3: Time ontology and associated data and object properties.

Jena Rules

```
// Time Rule 1: Deduction of happensBefore time instants....

[ TimeRule01: (?x rdf:type te:Instant) (?y rdf:type te:Instant) (?x te:hasTime ?t1)
              (?y te:hasTime ?t2) lessThan(?t1,?t2) -> (?x te:happensBefore ?y) ]

// Time Rule 2: Deduce if a time instant is inside a time interval

[ TimeRule02: (?x rdf:type te:TemporalEntity) (?y rdf:type te:Instant)
              (?x te:beginsAt ?t1) (?x te:endsAt ?t2) lessThan (?t1, ?t2)
              (?y te:hasTime ?t3) lessThan(?t1,?t3) greaterThan(?t2,?t3) ->
              (?y te:isInInterval ?x) ]
```

Figure 4.4: Two rules for reasoning with time.

ProperTimeInterval are subclasses of TemporalEntity. Temporal entities are defined by three properties, hasDuration (duration), beginAt (dateTime) and endsAt (dateTime). An open time interval adds two more properties, hasTimeAnchor (dataTime) and hasDuration (string). Instances of time add the data property hasTime (dataTime), plus two object properties, happensAfter and happensBefore. The latter are enough to create ordered lists of instances of time. Proper time intervals add properties to support the results of computations that evaluate the relationship among intervals of time (e.g., intBefore, intAfter, intContains, intMeets, intFinishedBy).

Figure 4.4 shows two illustrative rules for reasoning with instances and intervals of time. The first, determines if an instance of time (?x) occurs before a second instance of time (?y). Notice how the the Jena Rules builtin function has been designed to work with dataTime arguments. The second rule deduces if a time instance (?y) lies inside the interval (?x). Rules can also be written to fill out the range of Allen's interval calculus (see Figure 4.2).

4.3.2 Spatial Ontology and Rules

Spatial logic is concerned with regions and their connectivity, allowing one to address issues of the form: what is true, and where? Formal theories for reasoning with space – points, lines, and regions – are covered by region connected calculus [106]. A robust implementation of two-dimensional spatial entities and associated reasoning procedures is provided by the Java Topology Suite (JTS) [74].

Spatial Ontology and Rules for Spatial Reasoning. Figure 4.5 shows an

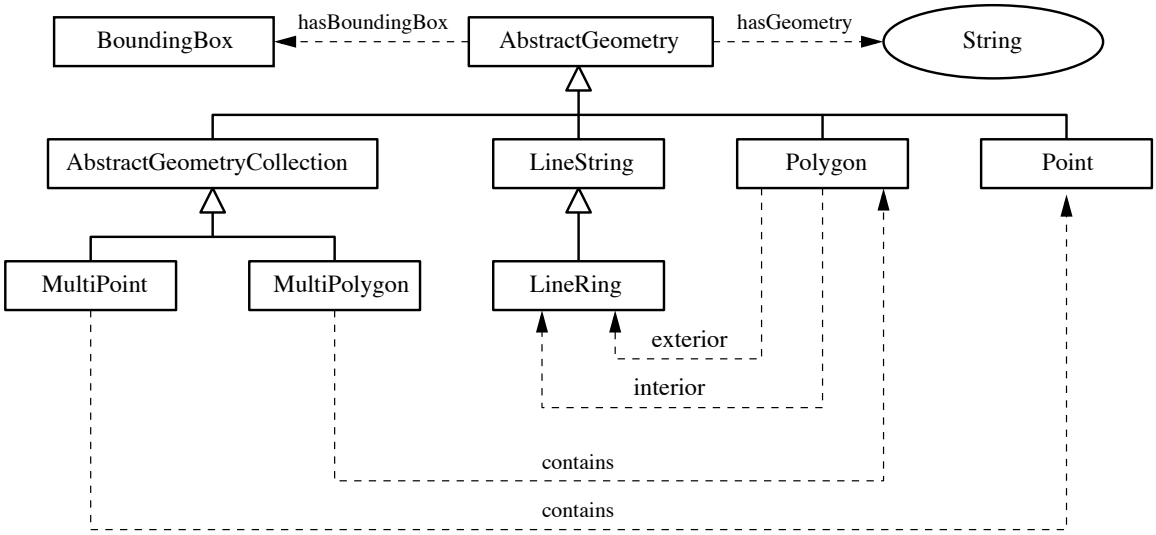


Figure 4.5: Abbreviated representation of spatial (geometry) ontology and associated data and object properties.

Jena Rules

```
// Rule to check if a sensor is inside a room ...

[ BuildingRule01: (?r rdf:type bld:Room) (?r bld:hasGeometry ?rg)
                  (?rg geom:hasGeometry ?rjts) (?s rdf:type sen:Sensor)
                  (?s sen:hasGeometry ?sg) (?sg geom:hasGeometry ?sjts)
                  getPointInPolygon(?sjts,?rjts,?t)
                  equal(?t, "true"^^xs:boolean) -> (?s bld:isInRoom ?r)]
```

Figure 4.6: Rules to determine the rooms in which sensors have been placed.

abbreviated representation of our experimental spatial (geometry) ontology and associated data and object properties. High-level classes – abstract concepts – are provided for entities that represent singular geometry (e.g., AbstractGeometry) and groups of entities (e.g., AbstractGeometryCollection). Specific types of geometry (e.g,, Polygon, MultiPoint) are organized into a hierarchy similar to the Java implementation in JTS. The high-level class AbstractGeometry contains a Datatype property, hasGeometry, which stores a string representation of the JTS geometry. For example, the abbreviated string "POLYGON ((0 0, 0 5, ... 0 0))" shows the format for pairs of (x,y) coordinates defining a two-dimensional polygon. This feature allows a semantic model to visit a domain data model, and gather a complete description of the two-dimensional geometry. Within Jena Rules, families of builtin functions can be developed to evaluate the geometric relationship between pairs of spatial entities (e.g., to determine whether or not a point is contained within a polygon). Figure 4.6 shows, for example, the Jena Rule that identifies the room in which a sensor is placed. An English translation of the rule fragments is as follows: If (?r) is a room with geometry (?rg) and string representation (?rjts), and (?s) is a sensor with geometry (?sg) and string representation (?sjts), then the builtin function getPointInPolygon(?sjts,?rjts,?t) will determine if the sensor (point geometry) is inside the room (polygon geometry) and return the result as a boolean (?t). If (?t) is true, then the sensor is inside the room and a new relationship (?s bld:isInRoom ?r) is created. A similar rule would be written to establish the relationship between sensors and HVAC zones.

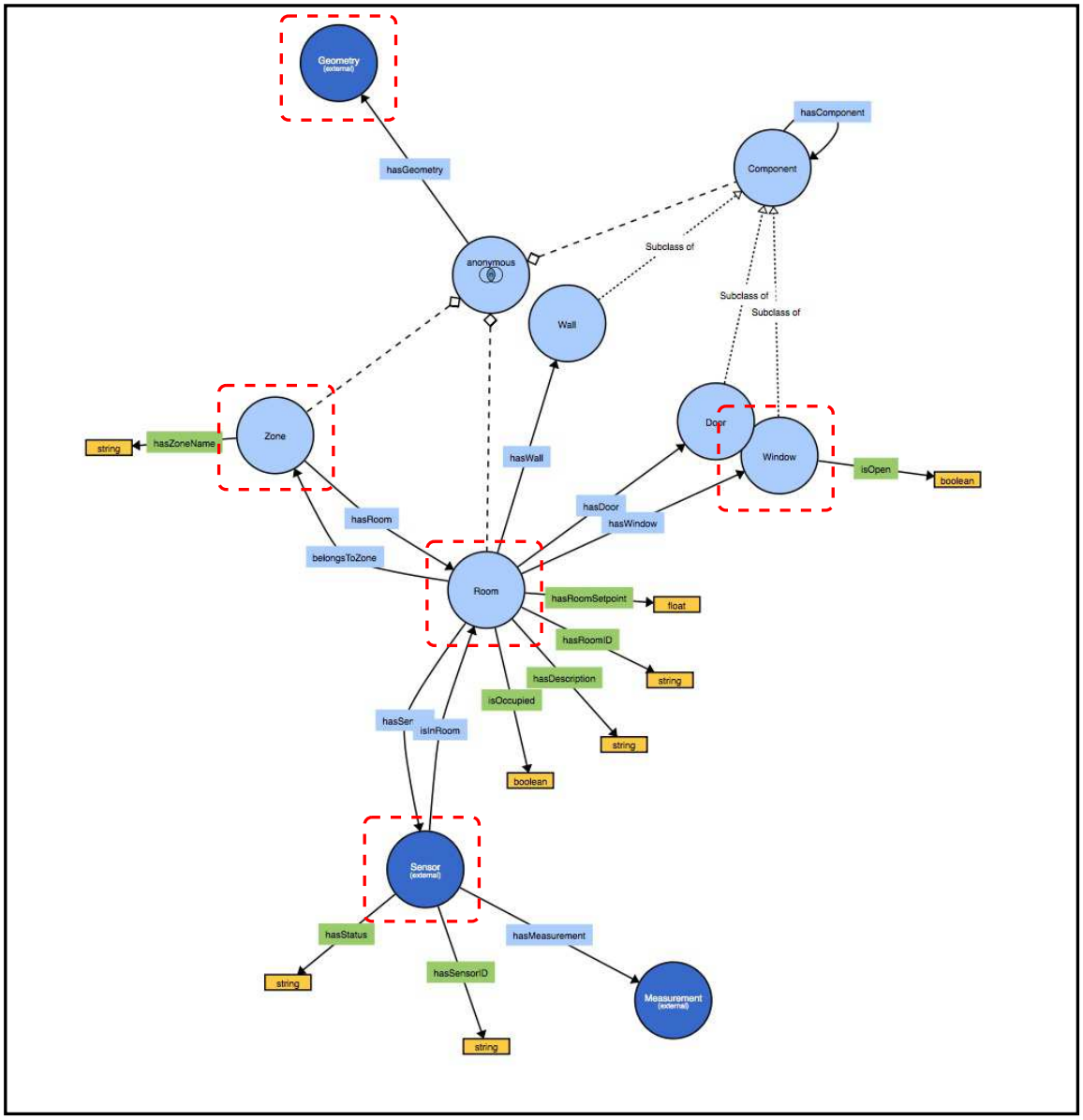


Figure 4.7: Schematic of building ontology classes and properties.

Jena Rules

```
[ BuildingRule02: (?r1 rdf:type bld:Zone) (?r1 bld:hasGeometry ?r1g)
                  (?r1g geom:hasGeometry ?r1jts)
                  (?r2 rdf:type bld:Zone) (?r2 bld:hasGeometry ?r2g)
                  (?r2g geom:hasGeometry ?r2jts)
                  notEqual( ?r1jts, ?r2jts ) getPointInPolygon( ?r1jts, ?r2jts, ?t)
                  equal(?t, "true"^^xs:boolean) -> (?r1 bld:intersects ?r2)]
```

Figure 4.8: Rule to check if two zones intersect.

4.4 Engineering Ontologies and Rules

In this dissertation, the engineering ontologies and rules cover four domains: (1) buildings, (2) mechanical equipment for HVAC systems, (3) sensors, and (4) procedures for fault detection and diagnostics.

4.4.1 Building Ontology and Rules

The overall performance of buildings depends on the building fabric consisting of the building roof, walls, windows and doors. The material, orientation, and building geometry and topology. The prototype building ontology and rules (see Figures 4.7 and 4.8) provide computational support for the representation of two-dimensional floorplan geometry, modeling relationships between elements of floorplan geometry and sensors, zones for HVAC control, and building elements such as doors, windows and walls. The latter are modeled as subclasses of a component that has geometry described by a JTS string.

Connections to the mechanical equipment and occupancy domains are achieved through data properties for the building environment state; see, for example, hasRoomSetpoint and isOccupied. Object properties record the relationship of a room to relevant HVAC zones and sensors. Windows have the boolean data property isOpen to record whether or not a particular window is open. As we will ee soon in the case study problem, this parameter plays a pivotal role in diagnostic analysis of the causes leading to a fault in mechanical equipment.

The prototype software implementation has one rule for determining the spa-

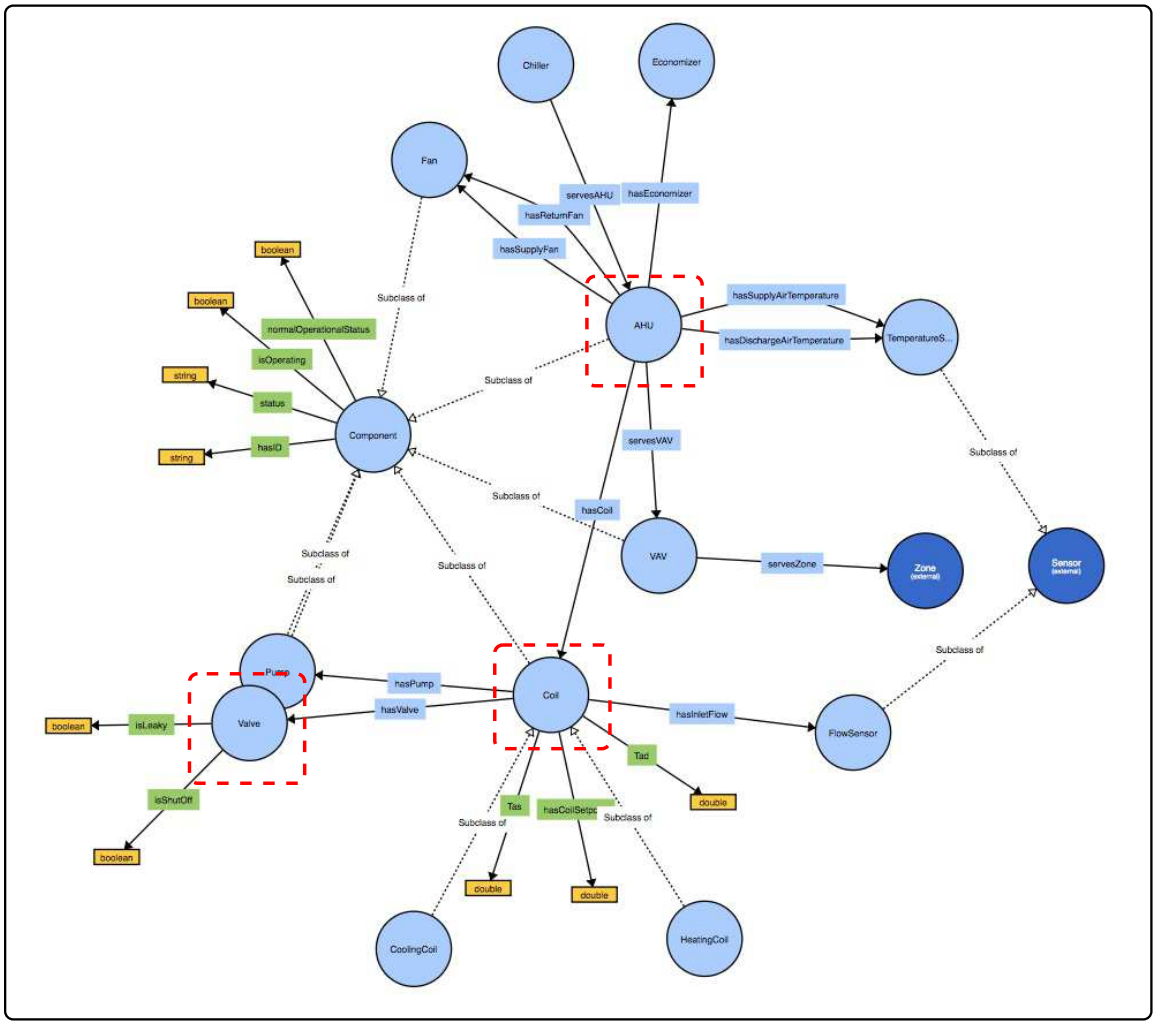


Figure 4.9: Schematic of equipment ontology classes and properties.

tial relationship among zones of the building. The rule systematically retrieves the JTS geometry of each zone, verifies they are not equal, and then uses the builtin function getPointInPolygon() to verify their geometric relationship. As previously noted, these backend computations are handled by the Java Topology Suite software [74].

Jena Rules

```
// Close the valve when the coil temperature is the same as coil setpoint.

[ EquipmentRule01: (?coil rdf:type eq:Coil) (?coil eq:hasCoilSetpoint ?sp)
                   (?coil eq:hasCoilTemperature ?cp) equal(?sp,?cp)
                   (?coil eq:hasValve ?valve) ->
                   (?valve eq:isShutOff "true"^^xs:boolean)

// If the valve is shut, the temperature of the air that passes through the coil
// has to be the same. Otherwise, the valve is leaky

[ EquipmentRule02: (?hwv rdf:type eq:Valve) (?hwv eq:isShutOff "true"^^xs:boolean)
                   (?c rdf:type eq:Coil)(?c eq:hasValve ?hwv) (?c eq:Tad ?t1)
                   (?c eq:Tas ?t2) notEqual(?t2 ?t1) ->
                   (?hwv eq:isLeaky "true"^^xs:boolean)
                   (?hwv eq:hasNormalOperationalStatus "false"^^xs:boolean)

// If the a valve fails, the AHU fails too ...

[ EquipmentRule03: (?hwv rdf:type eq:Valve) (?AHU eq:hasCoil ?c) (?c eq:hasValve ?v)
                   (?v eq:hasNormalOperationalStatus "false"^^xs:boolean) ->
                   (?AHU eq:hasNormalOperationalStatus "false"^^xs:boolean)]
```

Figure 4.10: Rules for establishing the operational status and simple operations of mechanical equipment.

4.4.2 Mechanical Equipment Ontology and Rules

Figures 4.9 and 4.10 illustrate the concepts (i.e., ontology classes), properties (i.e., data and object properties) and rules governing the operation and identification of faults in mechanical systems equipment. In practice, datatype property values

associated with the various ontologies will be set from streams of data either performed by a simulation tool (e.g. EnergyPlus, Dymola and TRNSYS) [36, 79, 120], or perhaps from measurements taken in a real building, working in conjunction with BACnet protocols [22] and a co-simulation middleware.

The semantic graph shown in Figure 4.9 is quite broad, covering concepts of HVAC systems from chillers and fans to zones. The scope of our investigation focuses on faults associated with valves, coils and air handling units. Basic rules (see Figure 4.10) are provided for: (1) controlling the flow in a coil valve, i.e., close the valve when a target setpoint is reached in the coil, (2) determining if a valve is leaky, i.e., when the temperature changes across the coil, (3) identifying situations where the normal operational status of a valve is false. Thus, we are able to determine that when a cooling coil valve is faulty, the associated air handling unit is also faulty.

4.4.3 Sensor Ontology and Rules

Figure 4.11 shows the classes and properties in our experimental sensor ontology. Our goal is to provide computational support for modeling: (1) sensor operation, including when a sensor reading might be outside an acceptable working range, and (2) determining the location of a sensor relative to the environment in which it is embedded. These objectives are achieved with three classes: Sensor, Measurement, and the external class Geometry. Support for modeling various types of sensor (e.g., temperature sensor, flow sensor, and CO2 sensor) is provided through the definition of specialized sensor classes that subclass Sensor. The class Measure-

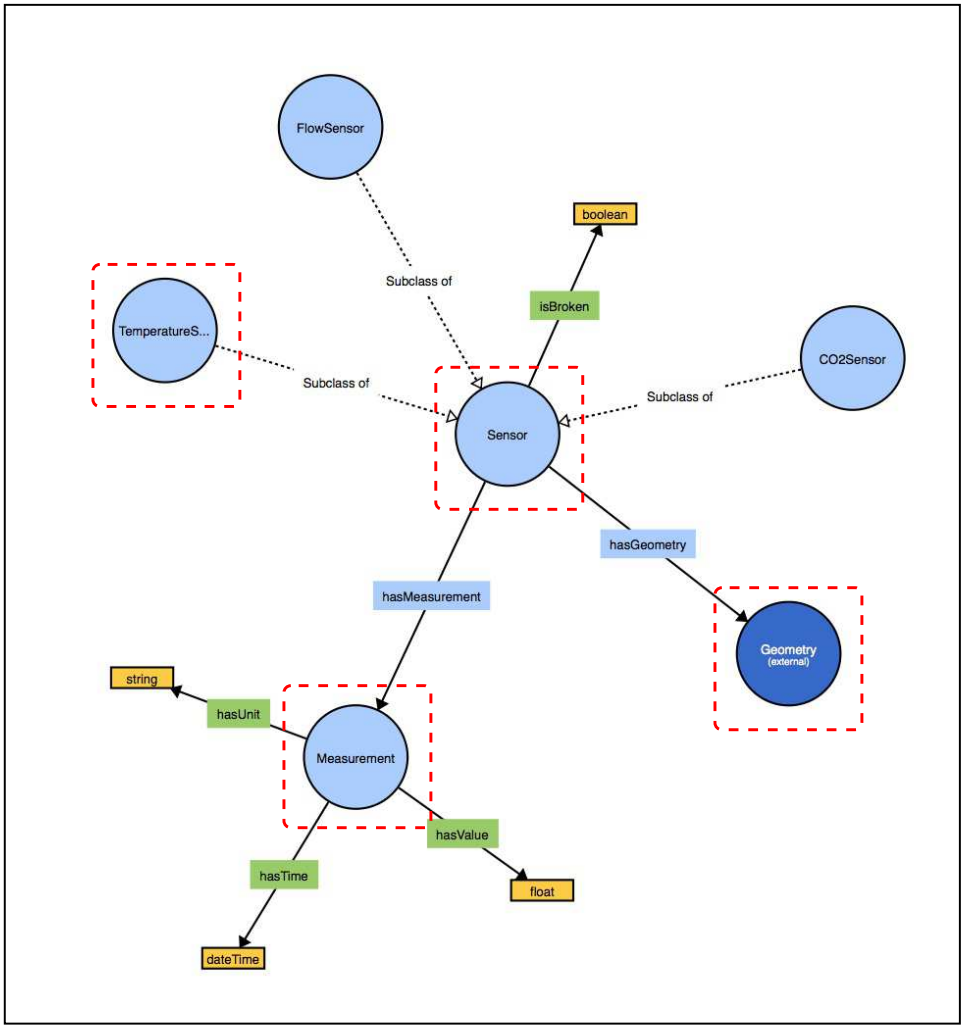


Figure 4.11: Sensor ontology classes and properties.

Jena Rules

```
// Simple rule to check if a sensor is broken ...

[ SensorRule01: (?s rdf:type sen:Sensor) (?s sen:hasMeasurement ?m)
                (?m sen:hasValue ?r) isOutOfRange(?m ?t) ->
                (?s sen:isBroken ?t) ]
```

Figure 4.12: Rule to compute intersection of zones.

ment has data properties to keep track of the current sensor value, the time, and the units associated with the measurement.

Two sensor rules (see Figure 4.12) are supported: (1) To determine if a sensor reading is beyond the acceptable range, (2) To determine the room in which the sensor is located. The first rule uses the classes Sensor and Measurement and associated properties. The second rule uses the classes Sensor and Geometry.

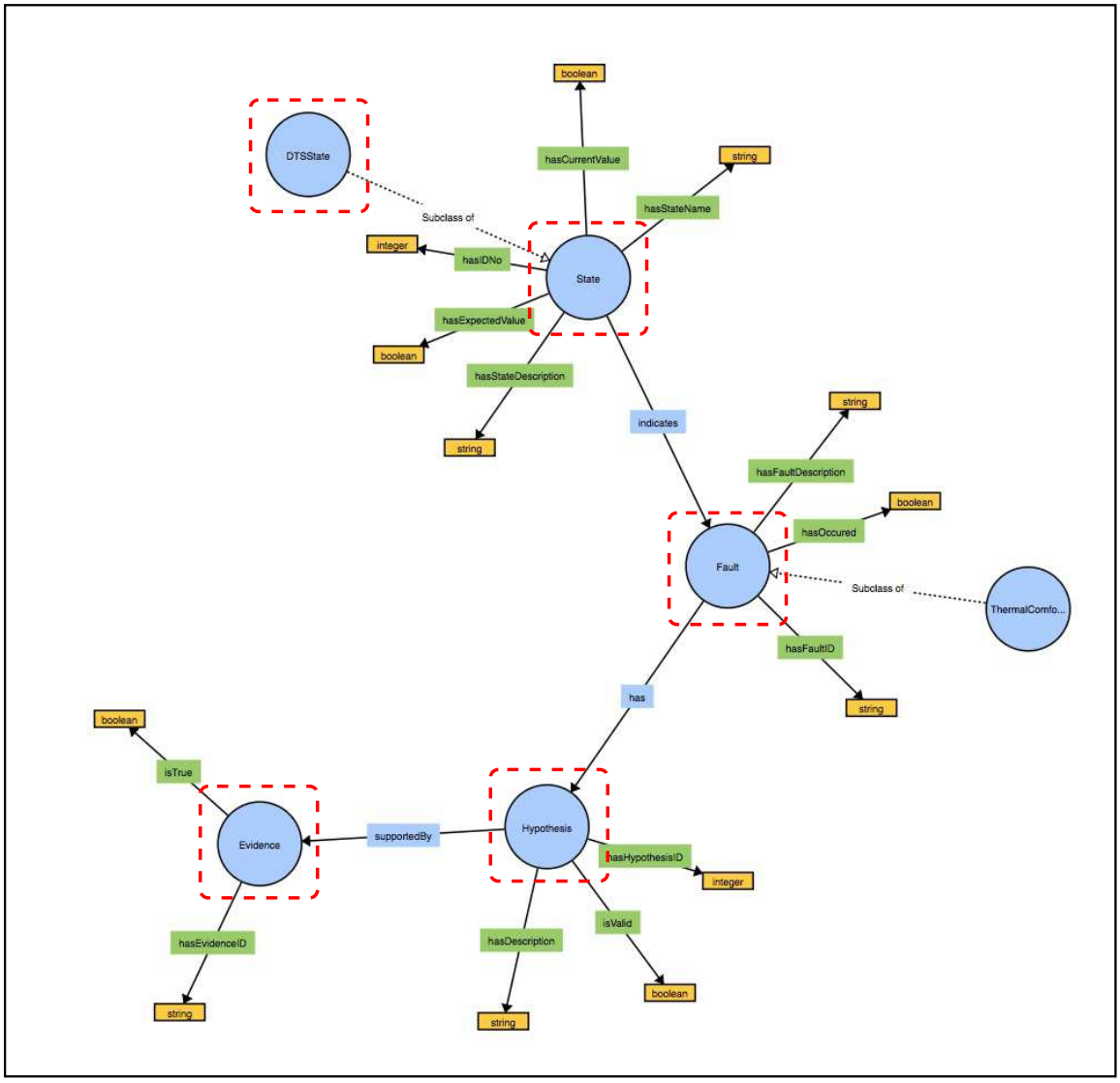


Figure 4.13: Fault detection and diagnostic ontology classes and properties.

Jena Rules

```
// General purpose rule for recording when a fault has occurred.

[FDDRule01: (?st rdf:type fdd:State) (?st fdd:hasCurrentValue ?csv)
            (?st fdd:belongsToFault ?F) (?st fdd:hasExpectedValue ?esv)
            notEqual(?csc,?esv) -> (?F fdd:hasOccured ''true'')   print('faultoccured')]
```

Figure 4.14: Rule for detecting a faulty state.

4.4.4 Fault Detection and Diagnostic Ontologies and Rules

This ontology captures the knowledge required for the process of detecting a fault in a system and identifying the root causes of the anomaly. The fault detection and diagnostic (FDD) ontology (see Figure 4.13) captures the knowledge needed for: (1) identifying that a fault exists, and (2) systematically diagnosing the fault to find the root causes. The main classes in this process are State, Fault, Hypothesis and Evidence. State is a high-level state representation that has data values – see, for example, the boolean properties hasExpectedValue and hasCurrentValue – common to many types of state representation. Our experimental FDD ontology also supports DTSState, a subclass of State, designed to represent states associated with dynamic thermal sensation (DTS).

4.4.5 Fault Detection and Diagnostic Procedures

Figure 4.15 is a flowchart for fault detection and the identification and verification of relevant hypotheses and supporting evidence. The step-by-step prcedure for detecting a fault and diagnosing its causes corresponds to a traversal through

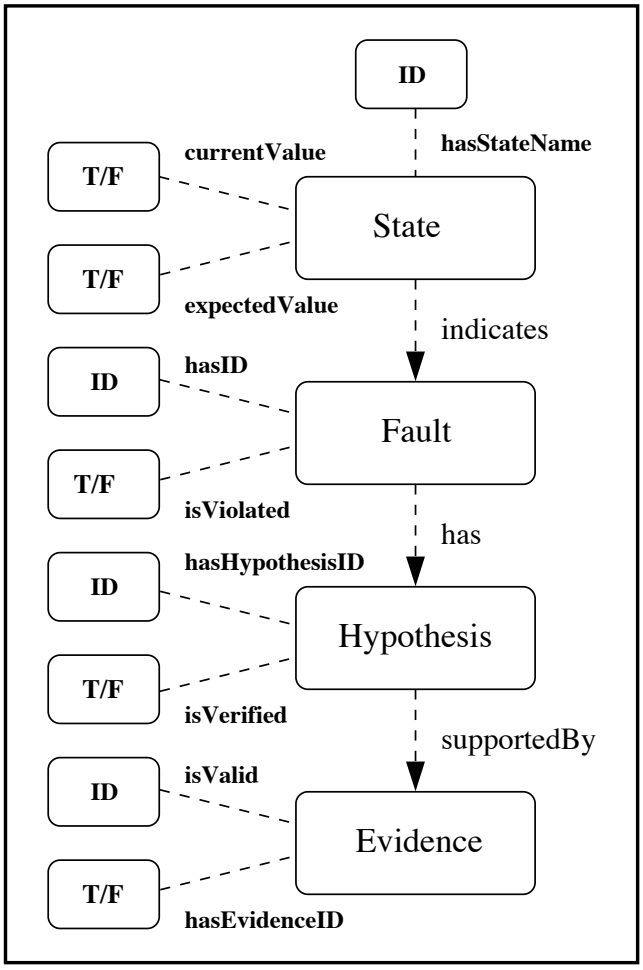


Figure 4.15: Flowchart for identification of faults, and identification and verification of hypotheses and supporting evidence.

the classes State, Fault, Hypothesis and Evidence. A fault is indicated when the current and expected values of a state are in conflict. Each fault has a hypothesis that needs to be supported by evidence. The evaluation procedure works backwards. Verification of the evidence is a prerequisite to validating a hypothesis. In an implementation of the procedure, data properties indicate whether or not a fault has been verified, whether or not a hypothesis has been verified, and whether or not supporting evidence is valid. This procedure is mirrored by a set of rules shown in Figure 4.14.

4.5 Surrounding Environment Ontologies and Rules

The surrounding environment ontologies and rules include model support for the building occupants and weather phenomena.

4.5.1 Occupant Ontology and Rules

While several studies [4, 86] have recently identified the importance of including inhabitants as an integral part of simulation and control of energy systems and indoor environments, present-day procedures rely on predetermined occupancy schedules [46] and/or empirical estimates based on sensors [78]. For fault detection and diagnostic analysis of mechanical equipment in buildings, solutions are complicated by the strong coupling of human presence, comfort and behavior, to details of the building state (e.g., whether or not a window is open) and surrounding environment (e.g., what side of the building is in the sun).

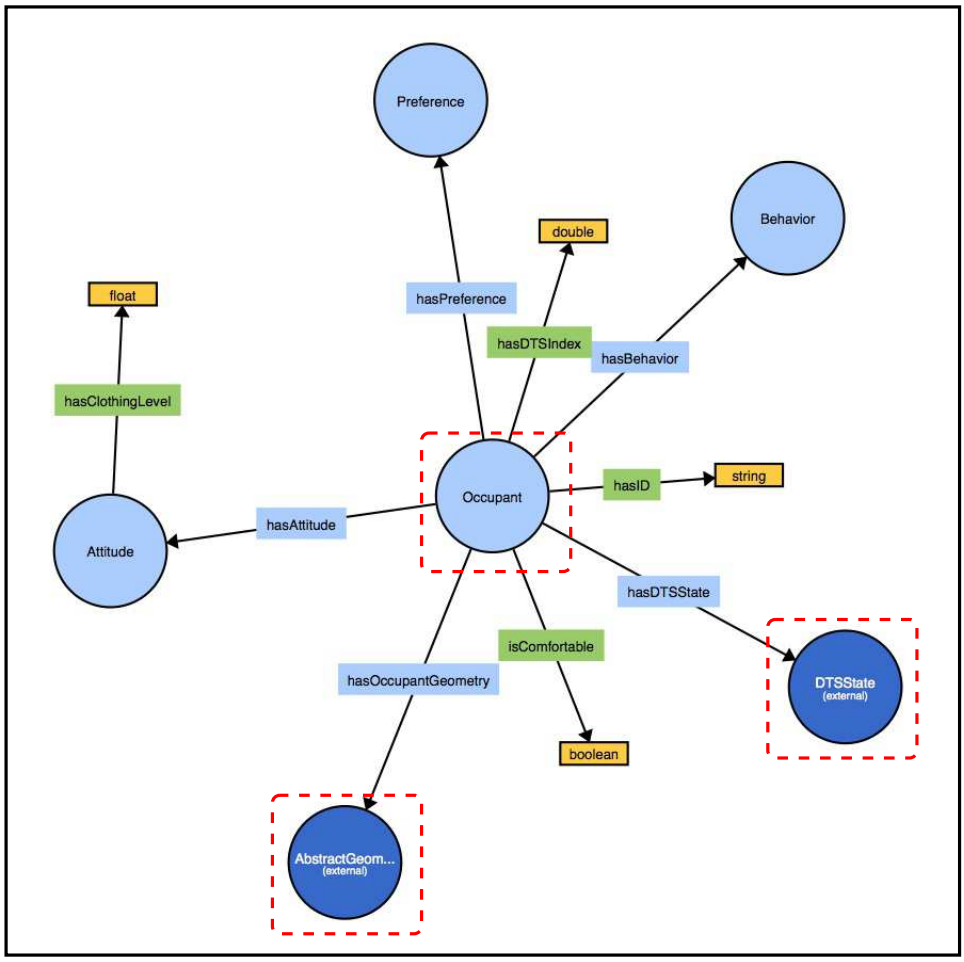


Figure 4.16: Schematic of occupant ontology classes and properties.

Jena Rules

```
// Determine romm in which an occupant is located.

[ OccupantRule01: (?r rdf:type bld:Room) (?o rdf:type occ:Occupant)
                  (?o occ:hasOccupantGeometry ?og)
                  (?og geom:hasGeometry ?ojts)
                  (?r bld:hasGeometry ?rg) (?rg geom:hasGeometry ?rjts)
                  getPointInPolygon(?ojts,?rjts,?t)
                  equal(?t, "true"^^xs:boolean) ->
                  (?r bld:hasOccupant ?o) print(?o,'OccupantisInRoom',?r,?t)]

// When positive values of DTSIndex are greater than 0.3, an occupant is not comfortable.

[ OccupantRule02: (?oc rdf:type occ:Occupant)
                  (?oc occ:hasDTSIndex ?v) greaterThan(?v,0.3)
                  (?oc occ:hasDTSState ?dts)  ->
                  print(?oc,'isComfortable' "false"^^xs:boolean)
                  (?oc occ:isComfortable "false"^^xs:boolean)
                  (?dts fdd:hasCurrentValue "false"^^xs:boolean)]
```

Figure 4.17: Rule for occupants location and thermal comfort.

Figures 4.16 and 4.17 take a first step toward the development of an ontology and rules for modeling occupant presense. The ontology expands upon the work of Mahdavi and Taheri [88], and considers four subcategory problems: (1) location, (2) actions (e.g., open/close window), (3) attitudes (e.g., thermal sensation) and (4) preferences in terms of temperature and moisture of the air. We model occupant location with a point geometry in the building, Figure 4.17 shows two rules that infer occupant's location and thermal comfort respectively.

One way to populate this ontology, in simulation, is with the occupants' behavior data obtained from the occupancy simulation frameworks such as DNAS and from models such as obXML [70]. In the case of real buildings, CO_2 and other sensor data will be used. The knowledge stored in this ontology will be used in deriving implicit knowledge. As a case, occupant's location will be used to infer occupant's

presence in a room.

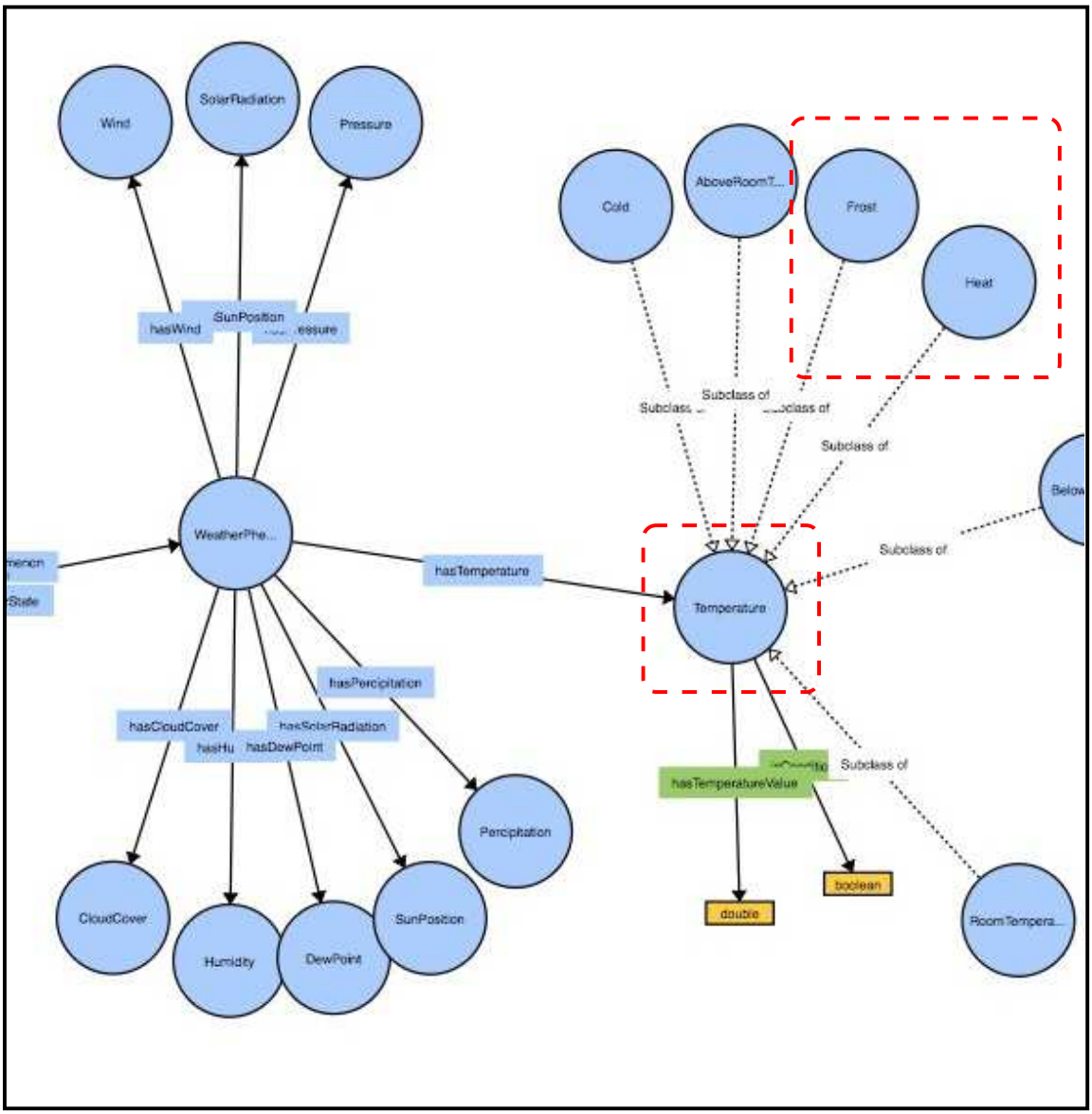


Figure 4.18: Partial view of weather ontology classes and properties (Source: Adapted from Staroch [114]).

4.5.2 Weather Ontology and Rules

Based upon the work of Staroch [114], Figure 4.18 presents the concepts that are used in Weather Ontology. The main concepts are Weather Phenomenon,

Jena Rules

```
// Use current temperature value to identify a frosty temperature condition ...

WeatherRule01: (?t rdf:type we:Temperature) (?t we:hasTemperatureValue ?tv)
               lessThan(?tv,0) ->  (?t rdf:type we:Frost)
               (?t, we:isCondition, "true"^^xs:boolean) print(?tv,'FrostCondition')]

// Use current temperature value to identify a heat temperature condition ...

WeatherRule02: (?t rdf:type we:Temperature) (?t we:hasTemperatureValue ?tv)
               greaterThan (?tv,30) -> (?t rdf:type we:Heat)
               (?t, we:isCondition, "true"^^xs:boolean) print(?tv,'Heat')
```

Figure 4.19: Rules to detect weather condition.

Weather Report, and Weather State. The weather state is composed of different Weather phenomenon class holds the physical attributes regarding the weather such as the temperature, pressure, solar radiation, wind and cloud. Weather data is obtained from [132], a free and open source API (application programming interface) that provides access to historical as well as current and future forecast weather data from an online server. A Weather report can include data about the current weather or a forecast, specified in terms of start time and duration. For example, a medium range weather report has duration of more than 3 hours, with a start time of less than 12 hours into the future.

Figure 4.19 presents two rules that use the current temperature value to identify frost and heat temperature conditions. A Frost temperature condition occurs when observed temperature is below 0°C. A Heat temperature condition occurs when observed temperature is above 30°C. Similar temperatures range can be defined for Cold, Below Room Temperature (at least 10°C and less than 20°C), and so forth.

This knowledge is very valuable for equipment operation. As a case, use thermal energy storage to make ice when there is heat advisory in the forecast.

4.6 Economic Ontologies and Rules

4.6.1 Utility Ontology and Rules

On important area of knowledge representation for building simulations is the utility domain. It is gaining more attention in building-to-grid-integration applications. One draw back of the state-of-the-art building simulation systems is the utility tariff is not modeled into the simulation in an easy to change, machine readable format. This ontology, along with its rule-sets, will provide a semantic model for defining different seasons, and time interval during a day for mid, off, and on peak rate structures. The benefit is that it is scalable and can be extended to include different utility tariffs. The building blocks of this ontology are borrowed from the time ontology used by Petgna [100], mainly the time interval and time instant.

Utility Ontology and Rule Sets. The purpose of this ontology is to capture the essential concepts involved in modeling of utility tariffs. The semantic modeling expands the temporal framework developed by Petnga and Austin [100]. It uses Jena API to create an ontology for defining electricity tariffs by extending the concepts from the time ontology. Temporal reasoning is achieved by defining rules that reason about time. For example, temporal reasoning is used to determine if a specific point in time is in an interval, or if an interval of time happens before another interval. The classes of the utility ontology are extensions from the time ontology.

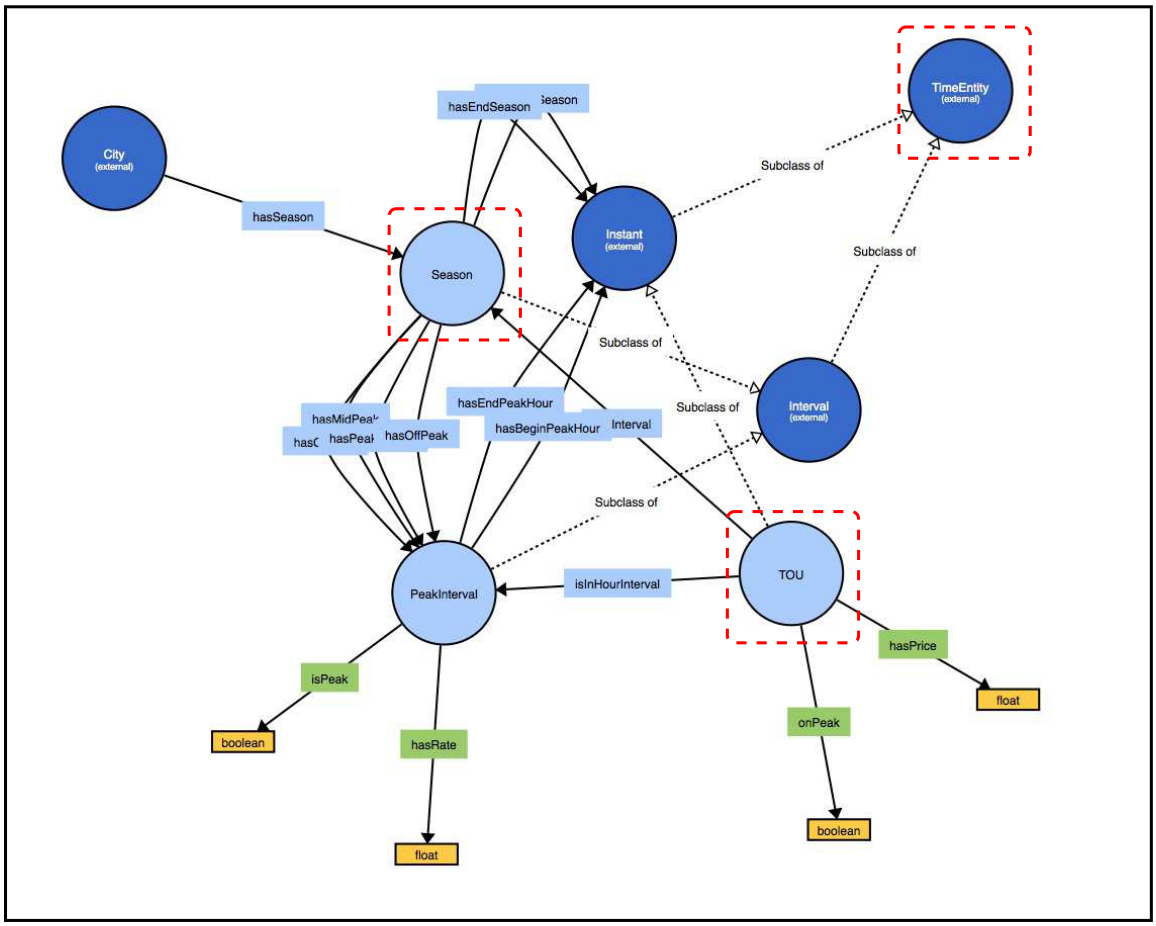


Figure 4.20: Schematic of utility ontology.

Figure 4.20 is a graphical representation of concepts, properties and relationships in the utility ontology. At the center of the ontology is the class "Season," which is a time interval with a beginning and an end date time. Depending on the rate structure, each season may have one or more, e.g., on/off/mid peak/peak, intervals. This is represented in Class "Peak Interval." A peak interval has a beginning and an end time in each day, and has an associated rate. The rules will determine if any point of time is a peak time and what the associated rate is. The rules identify which season the time of use belongs to. It then identifies which rate structure within that season applies based on the hour in a day. Finally, it checks to see if the hour is a peak hour and what the associated utility price is.

Figure 4.21 displays some sample Jena rules defined for the Utility ontology. UtilityRule01 identifies the season of operation based on "Time-of-the-Use". Following that, UtilityRule02 decides which rate category (on/off/mid) applies to a specific time. UtilityRule03 sets the flag of "onPeak" equal to the value of "isPeak" in that time interval, i.e., if time of use is in the on-peak category, the flag is set to "true". Finally, UtilityRule04 deduces the cost of electricity based on the hour and date of use.

4.7 Semantic Integration of Building Ontologies and Rules

This framework for knowledge representation will support semantic interoperability for different control algorithms to receive the data with unambiguous, shared meaning. This capability is a requirement to enable machine computable logic,

Jena Rules

```
// Utility Rule deduction if a time instant is in an regular hour interval

[ UtilityRule01: (?interval rdf:type te:TemporalEntity) (?interval te:endsAt ?end)
                 (?interval te:beginsAt ?begin) (?t rdf:type te:Instant)
                 (?t te:hasTimeValue ?time) lessThan(?begin,?time)
                 lessThan(?begin,?end) greaterThan(?end,?time) ->
                 (?t te:isInHourInterval ?interval) print(?t,?interval,'inHour')]

// Rule 16 deduction if a time instant is in before/after midnight hourly interval

[ UtilityRule02: (?interval rdf:type te:TemporalEntity) (?interval te:endsAt ?end)
                 (?interval te:beginsAt ?begin) (?t rdf:type te:Instant)
                 (?t te:hasTimeValue ?time) greaterThan(?begin,?end)
                 lessThan(?begin,?time) ge("23:59:59"^^xs:time,?time) ->
                 (?t te:isInHourInterval ?interval) print(?t,?interval,'inHour') ]

[ UtilityRule03: (?interval rdf:type te:TemporalEntity) (?interval te:endsAt ?end)
                 (?interval te:beginsAt ?begin) (?t rdf:type te:Instant)
                 (?t te:hasTimeValue ?time) greaterThan(?begin,?end)
                 lessThan("00:00:00"^^xs:time,?time) greaterThan(?end,?time) ->
                 (?t te:isInHourInterval ?interval) print(?t,?interval,'inHour')]

// Inferring the utility rate based on the season and the hourly intervals

[ UtilityRule04: (?tou rdf:type te:TimeOfUse) (?tou te:isInInterval ?season)
                 (?tou te:hasTimeValue ?time) (?tou te:isInHourInterval ?hourInterval)
                 (?interval rdf:type te:Season) (?interval te:hasPeak ?hourInterval)
                 (?hourInterval te:hasRate ?rate) (?tou te:isInInterval ?interval) ->
                 (?tou te:hasPrice ?rate)
                 print(?season,?tou,?hourInterval,?rate,'hasPrice')]

// Setting peak charge to true for OnPeak

   ... details of UtilityRule05 removed ...

// Setting peak charge to true for MidPeak

   ... details of UtilityRule06 removed ...

// Setting peak charge to false for OffPeak

   ... details of UtilityRule07 removed ...

// Determining if a specific interval has high peak (mid or on) or has low peak (off)

[ UtilityRule08: (?t te:isInHourInterval ?interval) (?interval te:isPeak ?peak) ->
                 (?t te:onPeak ?peak)]
```

Figure 4.21: Sample Jena rules for utility ontology.

inferencing, knowledge discovery, and data federation between information systems.

4.8 Summary

This chapter elaborated the ontologies required for knowledge representation in building simulation systems. This technique is highly scalable since the ontologies are decoupled from the building simulation and it can be extended to include more domains of interest. As a case, a domain that represents a building policy of operation and whether or not the building meets code requirements. This extensible approach allows users to extend new informational dimensions as with adding new ontologies with their governing rules. Finally, this section elaborates on the relevance of the domain independent ontologies in the development of domain ontologies by showing a case study in the case of building simulation domain. Utilizing semantic inference-based rules has several advantages: (1) Rules that represent policies are easily communicated and understood, (2) Rules retain a higher level of independence than logic embedded in systems, (3) Rules separate knowledge from its implementation logic, (4) Rules coupled from the model can be changed without changing source code or the underlying model, and (5) The Framework is extensible and can expand to include more constraints. An inference-based approach to problem solving is particularly beneficial when the application logic is dynamic (i.e., where a change in a domain needs to be immediately reflected throughout the application) and rules are imposed on the system by external entities. Both of these conditions apply to the simulation and control of energy systems in buildings.

Chapter 5: Case Study Applications

This chapter presents three case study applications covering knowledge representation in building monitoring and control, and fault detection and diagnostics. The applications areas are as follows:

1. Semantically-enabled control strategy for building simulation that includes model predictive control (MPC) for detection of occupant thermal comfort,
2. A semantic-based utility description for MPC in a Chiller Plant Operation,
3. Knowledge-based fault detection and diagnostics for HVAC systems.

5.1 Case Study 1: Semantically-Enabled Control Strategy for Building Simulation with MPC and Modelica (Dymola)

5.1.1 Problem Description

The case study examines the problem of conditioning a large five-zone room (see Figure 5.1) with variable air volume (VAV) boxes under a variety of control strategies.

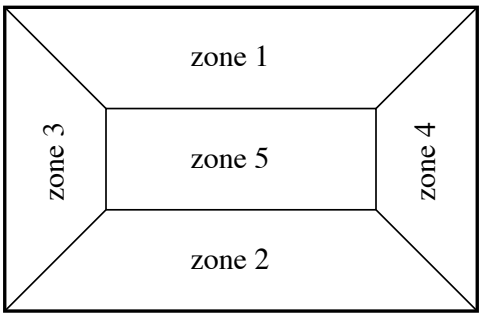


Figure 5.1: Plan view of large room with five thermal zones.

The system has one air-handling unit (to serve the five zones), and one economizer to take advantage of outdoor air if the condition permits.

5.1.2 Problem Goals

The goal of this case study is to understand how different control strategies perform with regard to room setpoint tracking and occupants thermal comfort level. Two control case studies are considered:

Control Case 1: A rule-based supervisory control strategy where the room setpoint is based on time and a fixed schedule.

Control Case 2: A co-simulation approach that incorporates optimization-based MPC control with semantic knowledge to adjust the setpoint temperature for the occupied zones so that thermal comfort requirements are not violated. The MPC computes the optimal zone temperature subject to constraints of the occupant dynamic thermal sensation model and the room physical models.

In both cases, the control objective is to keep the dynamic thermal sensation (DTS) index in the recommended range of -0.5 to 0.5 when the large room is occupied.

5.1.3 Problem Setup

Figure 5.2 is a schematic of the physical system model and Dymola environment, and a zoomed-in view of BCVTB integration with the physical model.

For the physical simulation, the room model along with the VAV boxes, air-conditioning unit, and the local PI and PID controllers are adapted from examples developed in the Dymola environment and distributed in the Modelica Buildings Library [92]. This library allows for rapid development of building systems models and has components for co-simulation purposes. The supervisory control is implemented using the Modelica State-Graph library. The library is used to represent state machines and capture how the system transitions through various states and occurrences of the events [99]. The MPC is implemented in Matlab in order to take advantage of that program's extensive library of optimization routines. Co-simulation between the MPC and Dymola is implemented in BCVTB.

Architectural Framework. State-of-the-art approaches to MPC employ models

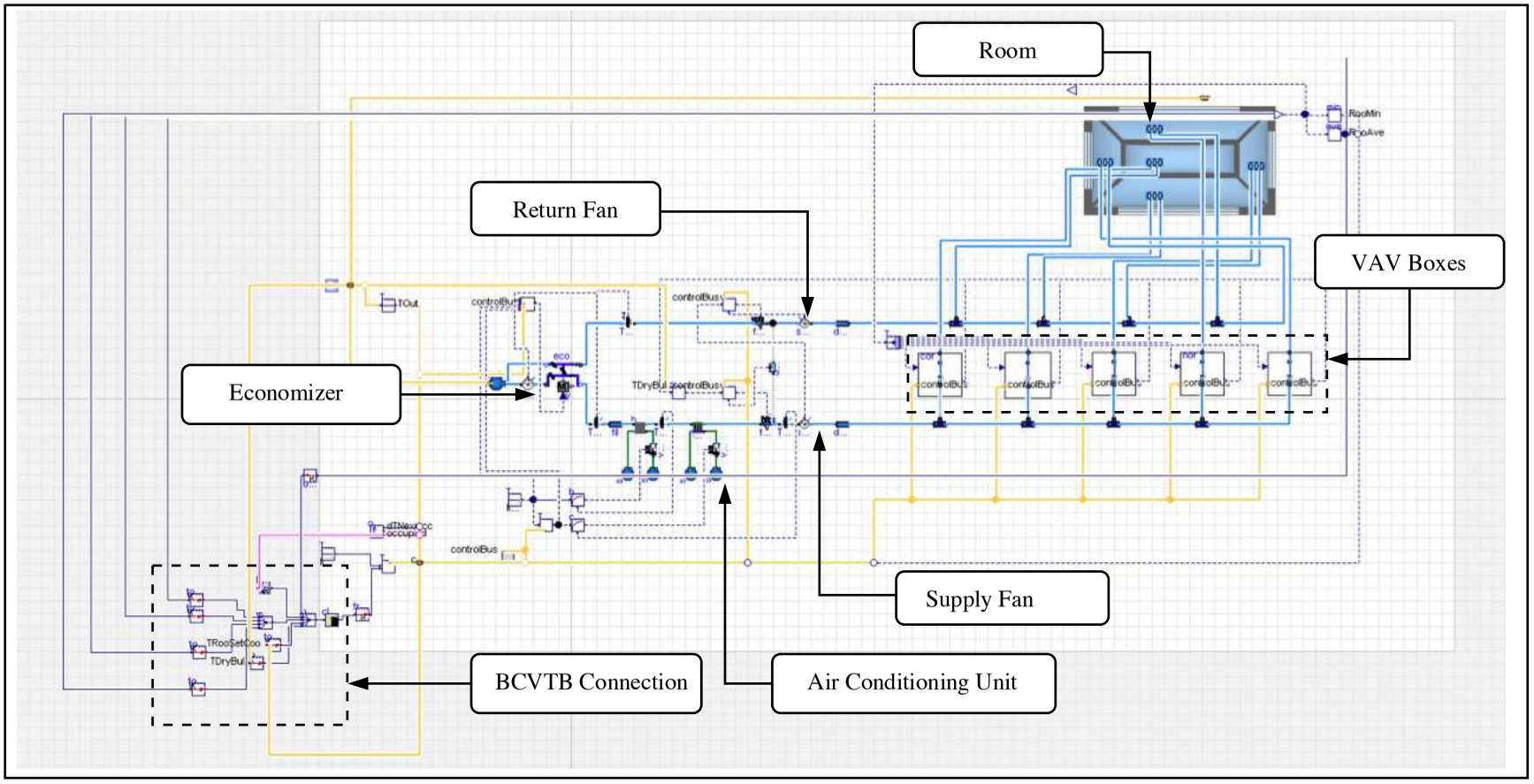


Figure 5.2: Schematic of physical simulation model and Dymola environment.

of dynamic behavior that are significantly simpler – often orders of magnitude simpler – than those used for simulation of the physical system. As such, estimates of building performance within the MPC decision making process are less accurate than those generated by full-scale building simulations. To improve upon state-of-the-art capability and close this gap, this research proposes a new architectural framework for co-simulation, Figure 3.2, that uses simplified models for DTS-based MPC decision making coupled with higher fidelity models for estimating actual building performance through BCVTB middleware. The higher fidelity models include detailed time-history simulations with tools such as Modelica, but can also include regression analyses based upon real-world data. In real building applications, the simulation model in Figure 3.2 is replaced by an interface with a real building using the BACnet protocol [22].

Data Exchange and Co-Simulation Framework. Figure 5.3 is a flowchart that represents the data exchange between the participating simulation actors in Control Case 2. The process begins with the MPC actor receiving the inputs of different zone temperatures from the physical models and weather forecasts; the MPC provides an optimal room setpoint back to the physical model. It is important to note that the supervisory control, can override the DTS-based MPC setpoint and provide the relevant control signals to the Modelica models. In the test case setup, room setpoints are only passed to the local PID controllers when the building is occupied (i.e., they will be passed to the local controllers located in and operating under the occupied mode).

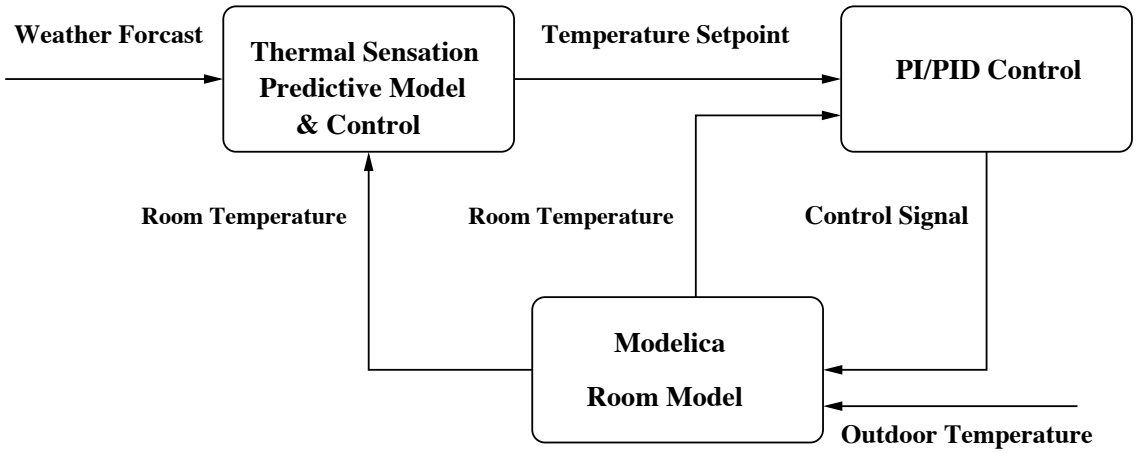


Figure 5.3: Flow diagram for data exchange in BCVTB

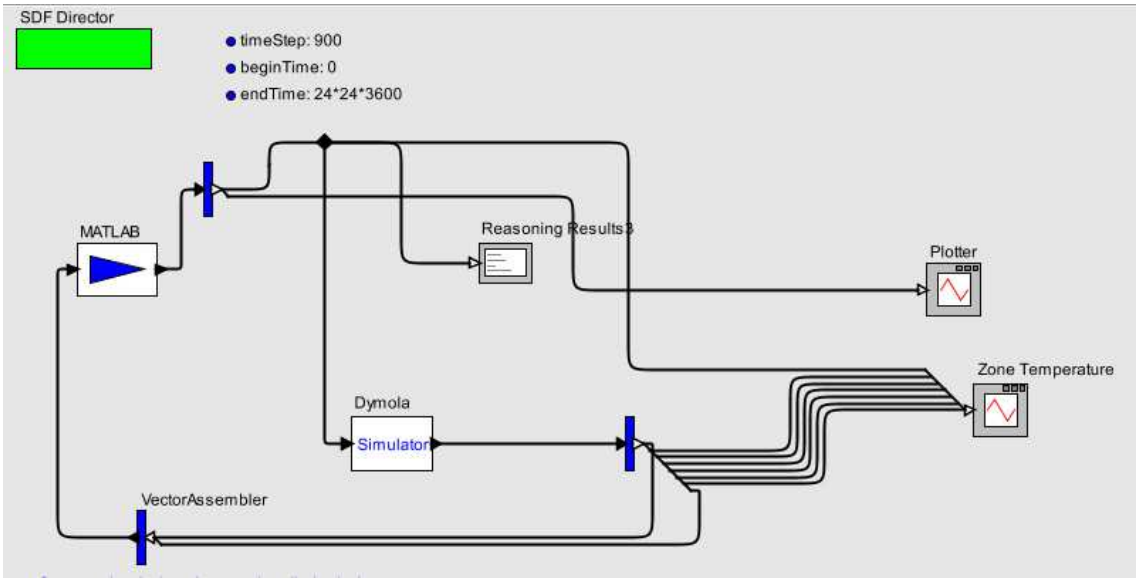


Figure 5.4: BCVTB framework DTS-based MPC and Modelica simulation model.

A schematic of the co-simulation framework and communication among the participating simulators is shown in Figure 5.4. For the dynamic sensation model (simulated in MATLAB) and the physical model in Dymola, the simulation time step is 900 seconds. Synchronization of computational results occurs at each time step – in particular, the MPC actor exchanges the setpoint temperature with the indoor zone and outdoor temperature in Dymola. Figure 5.5 shows the setpoint, room temperature, and control signal for this configuration. Finally, Figure 5.6 illustrates the representative implementation of the DTS model for this configuration.

DTS-based MPC Formulation. Based on the model developed by Chen et al. [27], the DTS-based MPC formulation optimizes the occupant comfort level. The mathematical details are as follows:

Cost function:

$$\begin{aligned} \min \quad J_t = \; & \textstyle\sum_{k=1}^{N}[T_{sply}(t+k|t) - T_{chmbr(t)}]^2 \\ & + \lambda_1 \textstyle\sum_{k=1}^{N}[T_{sply}(t+k|t) - T_{sply}(t+k-1|t)]^2 \\ & + \lambda_2 + \textstyle\sum_{k=1}^{N}[q(t+k|t)]. \end{aligned} \tag{5.1}$$

Equation 5.1 minimizes the energy consumption by decreasing the difference between the setpoint (i.e., supply) and the room temperature. Its second purpose is to minimize the difference between two consecutive indoor temperatures. The cost function is subject to the minimum and maximum possible supply temperature and the thermal sensation index range.

Here, λ_1 and λ_2 are coefficients that penalize violations of thermal comfort.

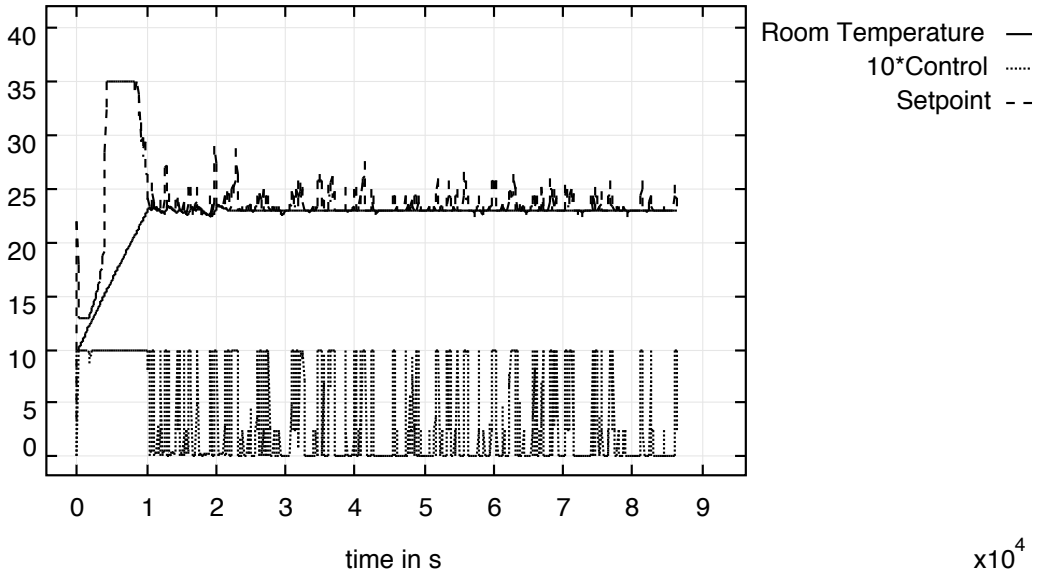


Figure 5.5: Setpoint, room temperature, and control signal.

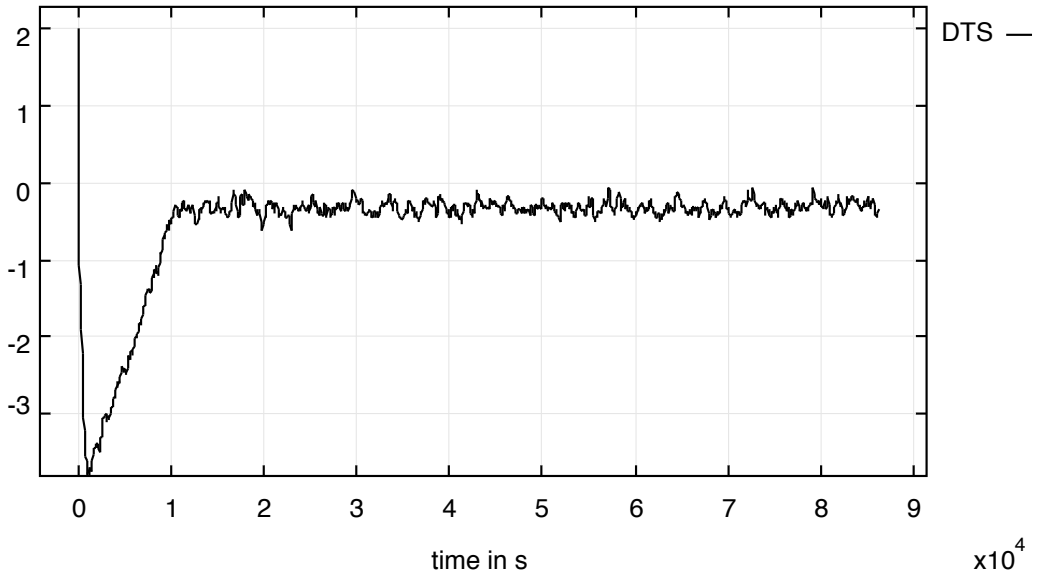


Figure 5.6: Dynamic thermal sensation model – Acceptable range is -0.5< DTS < 0.5

$q(k)$ represents the optimization slack variable. The simulation time and horizon time step are represented by t and k, respectively. In this equation, $T_{chmbr}(t)$ and $T_{sply}(t)$ denote the chamber and supply air temperature, respectively.

Subject to:

$$y_{min} - q(t+k|t) \ll TS(t+k|t) \ll y_{max} + q(t+k|t)q(t+k|t) \gg 0 \tag{5.2}$$

$$q(t+kIt) \gg 0 \tag{5.3}$$

Constraints on supply air temperature:

$$T_{sply}^{min} \ll T_{sply}(t+k|t) \ll T_{sply}^{max} \tag{5.4}$$

$$-\Delta T_{sply} \ll T_{sply}(t+1|t) - T_{sply}(t) \ll \Delta T_{sply} \tag{5.5}$$

Where the chamber dynamic model is represented by:

$$\begin{aligned} T_{chmbr}(t+k+1|t) = \quad & 0.965 * T_{chmbr}(t+k|t) \\ & +0.0286 * T_{sply}(t+k|t) + 0.0523 * T_{sply}(t+k-1|t) \\ & -0.0257 * T_{sply}(t+k-2|t) - 0.0315 * T_{sply})(t+k-3|t) \\ & +0.0133 * T_{out}(t+k|t) + 0.0232 * G_{in}(t+k|t) \end{aligned} \tag{5.6}$$

and the dynamic thermal sensation model is described as:

$$TS(t+k|t) = f(T_{chmbr}(t+k|t), T_{chmbr}(t+k-1|t), ...) \tag{5.7}$$

In these equations, T_{out} denotes the outside temperature and G_{in} represents the internal gain. Variable TS represents the thermal sensation index.

Dynamic Room Model Identification. The data-driven model of the chamber temperature is borrowed from Chen et al. [27], who performed a regression analysis on data collected from a 8.5 m*2.7 m*3.9m chamber to see how the temperature varies over time.

$$\begin{aligned} T_{chmbr}(t+1) &= 0.965 * T_{chmbr}(t) + 0.0286 * T_{sply}(t) \\ &+ 0.0523 * T_{sply}(t-1) - 0.0257 * T_{sply}(t-2) \\ &- 0.0315 * T_{sply})(t-3) + 0.0133 * T_{out}(t) + 0.0232 * G_{in}(t) \end{aligned} \tag{5.8}$$

Equation 5.8 is a discrete dynamic model of the chamber behavior. It shows how a future value of chamber temperature is computed based on previous values of the chamber temperature and the supply temperature. The sampling time of this model is 60 seconds. The investigators observed that the predicted chamber air temperature agrees with the measured chamber temperature with the coefficient of determination $R^2 = 99.14$.

Dynamic Thermal Sensation model Identification. A data-driven state-space

DTS model with Wiener structure has been developed by Chen et al. [27]. This empirical model is based on the results of survey questionnaires obtained for different indoor temperatures; it captures the occupant's thermal sensation due to changes of indoor temperature through Equations 5.9 and 5.10.

$$\begin{aligned} x(t+1) = \quad & 0.798.x(t) + 0.0610.x(t-1) + \\ & T_{chmbr}(t) - 0.883.T_{chmbr}(t-1) + e(t) \end{aligned} \tag{5.9}$$

$$y(t) = \frac{3.033}{exp[-0.558.(x(t) - 7.931) + 8.166] + 1} + d(t) + v(t) \tag{5.10}$$

Equation 5.9 represents how the future value of thermal sensation is computed based on chamber temperature. Equation 5.10 describes how the observed mean vote is accounted for in the thermal sensation model. Here, x denotes the thermal sensation state (7-point scale) and y denotes the observed mean vote of thermal sensation. Parameters e and v represent the process noise and measurement noise respectively. The parameter d is an offset parameter and its nominal value is 0.994.

5.1.4 Results

The computational experiments extend over a five-day period during the hot season. The results are displayed in Figures 5.7 through 5.10. The negative and positive values of the DTS index indicate perceptions of temperature that are cold and warm, respectively. For a comfortable person the DTS index value is between -0.5 to 0.5. A stringent consideration is between -0.3 to 0.3. Note that based on the ASHRAE scale for thermal sensation, a value of 1 is considered slightly warm. Figure

5.7 shows the DTS index versus time for Control Case 1. In this computational strategy, the setpoint values are based on occupancy schedule, but do not vary as a function of time. It tries to increase the setpoint before the occupants enter the spaces and decrease the setpoint around the time occupants are scheduled to leave. The simulation results indicate that the DTS index goes beyond the acceptable range of -0.5 to 0.5 to 0.8.

Figure 5.8 shows the DTS index versus time for Control Case 2. Notice that in this case the DTS index is bounded by -0.6 to 0.6 – an almost acceptable range – which is slightly higher than the recommended range. It is important to note that DTS stays in the acceptable range during the occupied time and goes beyond the comfortable range when transitioning from night setbacks to setpoints. It is speculated that a limitation of this approach stems from steady-state assumptions used in the formulation of the DTS model – as such, the model may be incapable of representing accurate values of perceived sensation in a fluctuating indoor temperature (i.e., the zone temperature rises dramatically during scheduled night setbacks when the zones are assumed to be unoccupied).

Figure 5.9 shows the temperature trend over time when MPC is not implemented and only the rule-based strategy is in effect (Control Case 1). It shows that even during occupied hours, the zone temperatures do not follow the setpoint closely. Figure 5.10 depicts the zone temperature under Control Case 2. Notice that during the occupied time, the zone temperatures track the MPC setpoint closely. However, during the night setback, the rule-based supervisory control will increase the zone temperature to a level that is not accepted by MPC thermal comfort constraints.

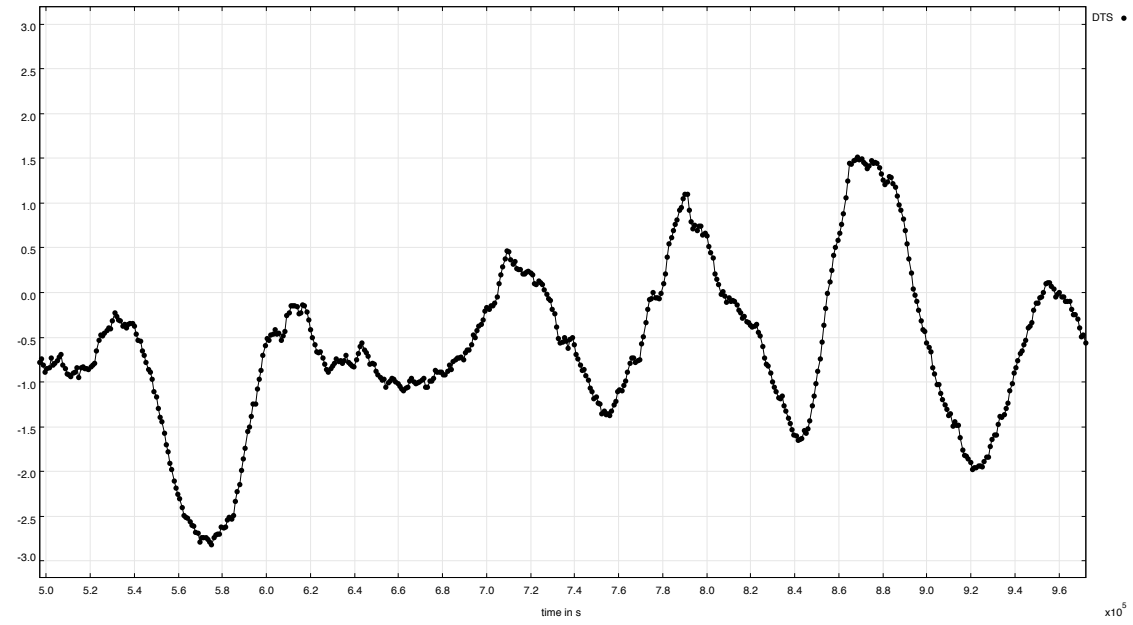


Figure 5.7: Dynamic thermal sensation index versus time (sec) for Control Case 1.

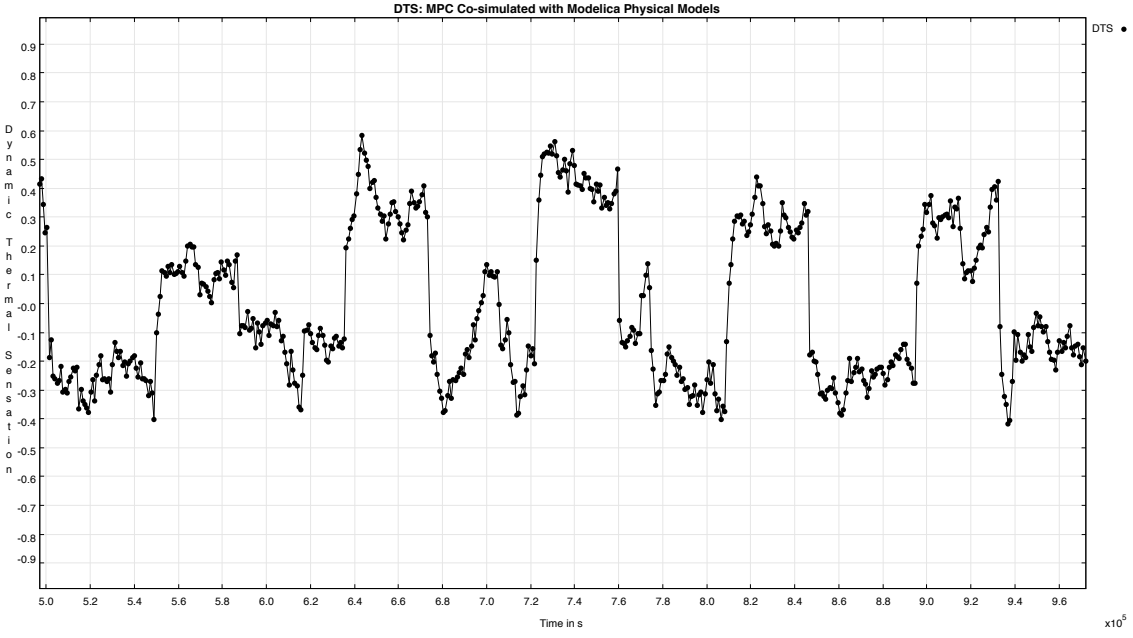


Figure 5.8: Dynamic thermal sensation index versus time (sec) for Control Case 2.

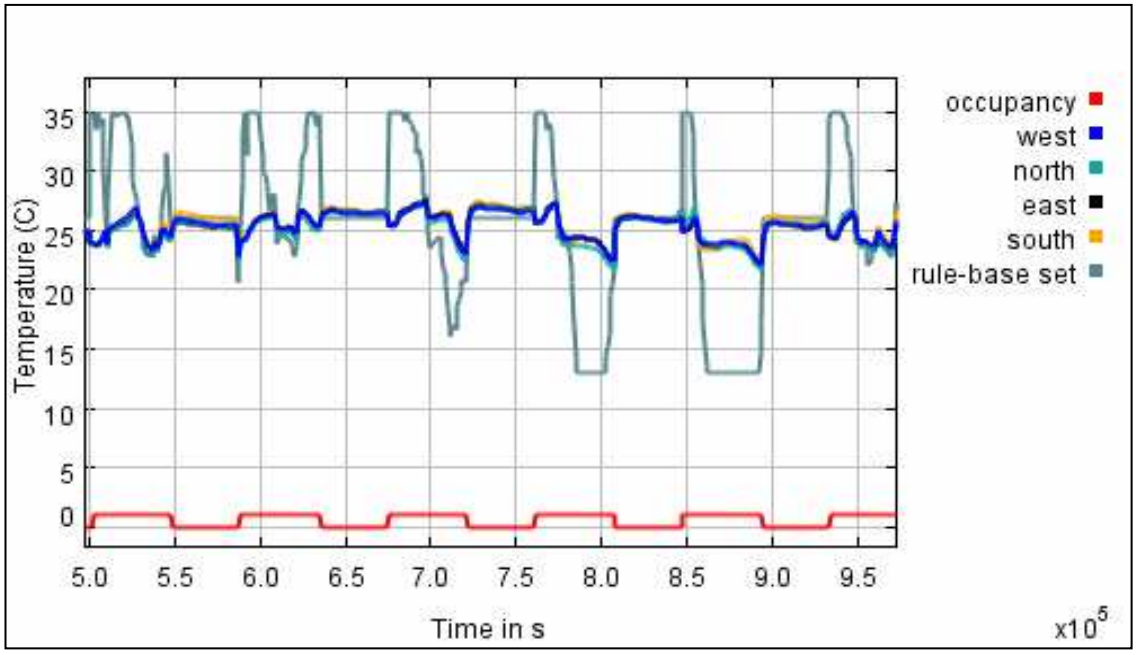


Figure 5.9: Profiles of room occupancy, rule-based setpoint, and temperature versus time (sec) for Control Case 1.

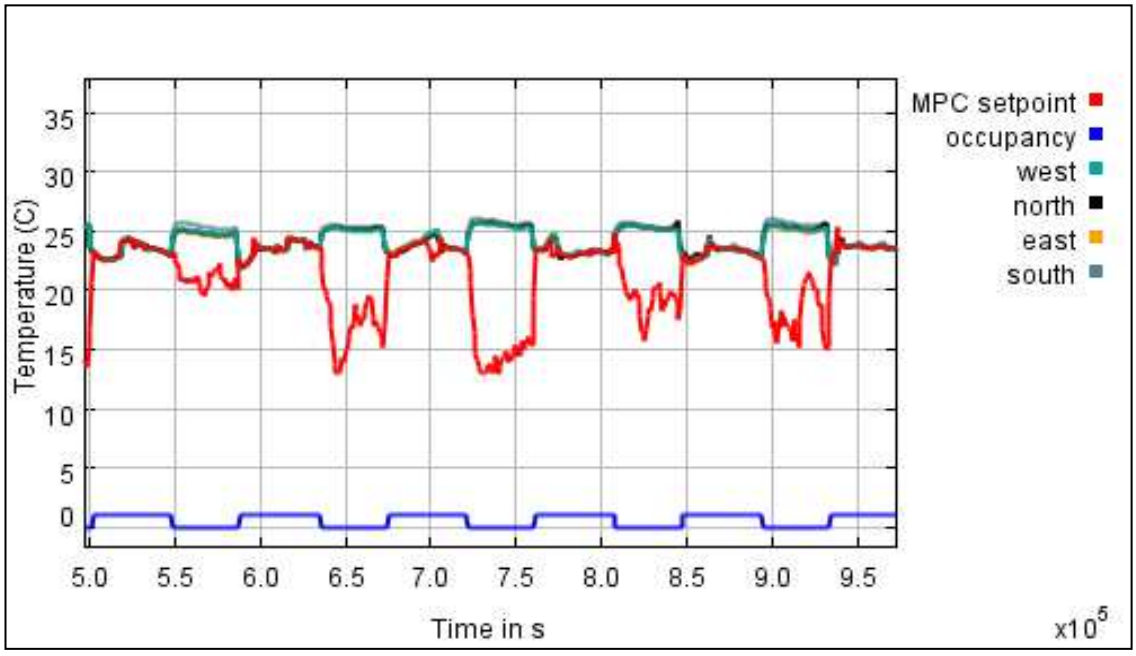


Figure 5.10: Profiles of room occupancy, MPC setpoint, and temperature versus time (sec) for Control Case 2.

This causes MPC to fail.

5.1.5 Findings

The results include identification of limitations on HVAC systems simulation and control performance, which, in turn, point to opportunities for our future development. First, it was identified that low DTS values are achieved through the co-simulation of DTS-based MPC and physical building models with the knowledge about occupant presence in the zone. Second, current thermal comfort research reports extensively on various aspects of the human thermal response to stable chamber conditions where the indoor operative temperature is at steady state [**?**]. It follows that dynamic thermal sensation models do not faithfully represent true thermal sensation when there is a considerable change in the chamber temperature. The current DTS model does not account for transient environmental conditions caused by variations in room setpoint tracking occupant thermal sensation. Thus, we propose that future programs of work should mitigate these limitations through the use of models that account for fluctuations in indoor temperature [52]. A second need involves enhancing the current air side rule-based (i.e., state machine) supervisory control with MPC based chilled water plant control. In such problem setups, the results of plant MPC will set optimal setpoints for cooling and heating coils. Finally, after testing different control approaches on the simulation models, we will move on to integrating the tested control strategies with real buildings through BACnet protocols.

The co-simulation defines a pathway toward the systematic evaluation of different control such as MPC control and other (i.e., rule-based) strategies in real building systems, with the scope of this study linking DTS-based MPC control to building Modelica models and thermal properties of the inhabitants. Not all of the strategies explored in this study led to good (or even satisfactory) levels of system performance. For example, strategies of rule-based control that do not include a thermal sensation model result in levels of room temperature extending beyond accepted ranges of thermal comfort. However, when a co-simulation of DTS-based MPC and Modelica physical models of building and HVAC is employed alongside BCVTB (i.e., Control Case2), it is possible to achieve levels of comfort for the occupants (i.e., DTS value between -0.5 and 0.5) that are superior to values obtained by the use of rule-based approaches (i.e., Control Case1). Having this co-simulation framework will be an important part of the groundwork for evaluating new approaches to simulation and optimization of building systems performance, supported by combinations of regression models of measured data, complex physical simulations and different MPC methods.

This framework can be extended to include the ontologies that represent the knowledge of building. The results of building energy simulations will be stored in those ontologies for further processing and knowledge management to be used in semantic-assisted MPC control.

5.2 Case Study 2: Knowledge-Assisted MPC for Utility Representation

State-of-the-art building simulation control methods incorporate physical constraints into their mathematical models, but omit implicit constraints associated with policies of operation and regulation. To overcome these shortcomings, one solution is to exploit Semantic Web technologies in building simulation control. Such approaches provide the tools for semantic modeling of domains, and the ability to describe the policy and regulations in terms of rules in those domains.

5.2.1 Problem Description

In a step toward enabling this capability, this application case tests a semantics-assisted control strategy for building simulations that integrates ontologies and reasoning mechanisms into a Model Predictive Control (MPC) formulation. This integrated control strategy was tested for MPC involving the operation of a cooling, heating and power plant equipped with a thermal energy storage (TES) unit that is optimized for utility rates. The study investigated three different electricity tariff structures associated with cities of Austin, New York, and San Francisco and their impact of the plant operation [67].

5.2.2 Problem Goals

The goal of this prototype implementation is to integrate domain specific ontologies, such as Utility domain, and the associated rules for capturing utility rate constraints in system simulations. The main advantage is that these models are decoupled from control strategies such as MPC and are scalable and easy to adapt. For example, if the utility tariff changes.

5.2.3 Problem Setup

The predictive control approach will exploit dynamic models, thermal energy storage, and predictions of zone loads, utility rates to minimize energy cost while meeting equipment and thermal comfort constraints. At each time step of the prediction horizon, the ontology is queried by the MPC unit to determine (via temporal reasoning) the applicable electricity rate tariff. The rules in the ontology support time-variant electricity pricing (TOU).

Architecture for Coupled Semantic/MPC HVAC Control. Figure 5.11 shows a simplified architecture for simulation and control.

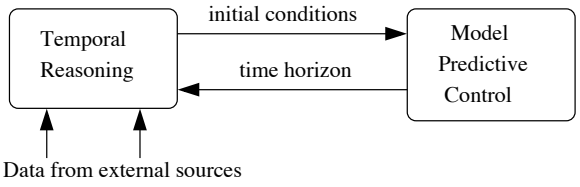


Figure 5.11: Architecture for coupled semantic/MPC HVAC system control.

It is composed of two parts of the semantic model and MPC control. Electricity rate

predictions for the MPC stem from the semantic model and associated reasoning processes described in Section 4.6.1.

Figure 5.12 is a schematic of the multi-level HVAC control structure used in this study.

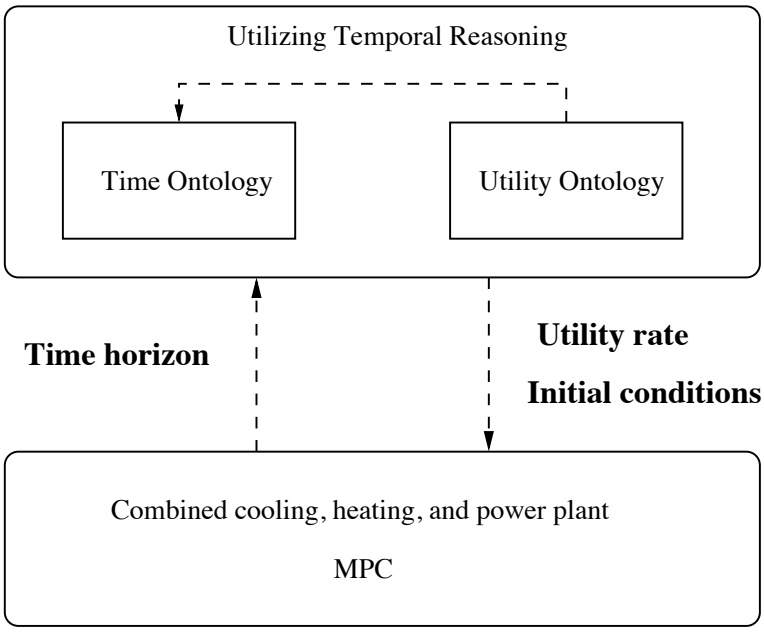


Figure 5.12: Multi-level control structure for HVAC systems.

At the beginning of the time horizon, the MPC optimization routine acquires the predicted utility rate and initial conditions for the decision variables from the ontology.

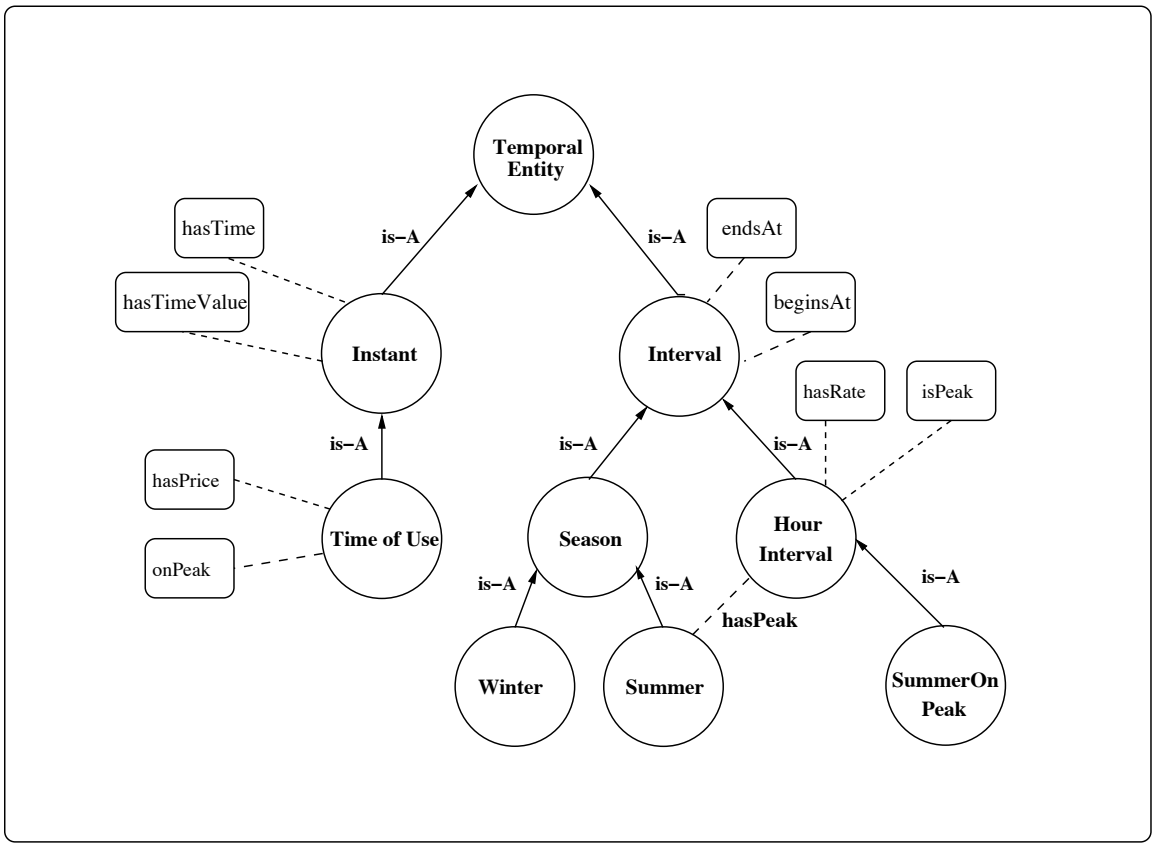
Figure 5.13 shows that the city of Austin has a summer season that begins on 05/01 and ends on 10/30. During this season from 2 p.m. to 8 p.m. are on-peak hours. The electricity rate tariff is based on time-of-use (TOU) which breaks up the day into two or three time intervals, i.e., off-peak, on-peak, mid-peak. In addition, months are categorized as either the heating or cooling season. This approach encourages

customers to shift the load away from the times of the day that demand and rates are higher. However, it does not necessarily lead to less energy consumption during critical peak periods, such as heat waves.

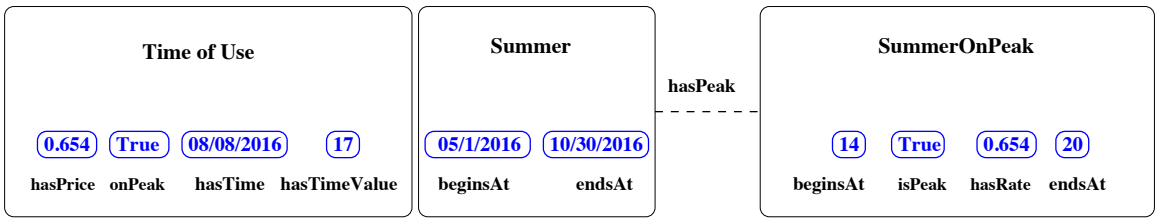
Formulation of Model Predictive Control Problem. The study adapts the MPC algorithm developed by Chandan and co-workers [25] for modeling and cost optimization of a combined cooling, heating and power (CCHP) plant. The plant consists of three electric chillers that can provide chilled water to a campus for cooling, a stratified thermal energy storage unit (TES), two generators, a gas turbine, a steam turbine, and a heat recovery unit. The plant supports co-generation, where the heat recovered from generators is utilized for production of thermal energy and electricity. TES is used to reshape the cooling demand during the course of a day by reducing the cooling load met by the chiller banks. The inputs to MPC are the cost of electricity and the building cooling load. The decision variables are the chiller mass flow rates, mass flow rate supplied to the building, the chiller supply temperature, the return temperature from the building, power supplied by the gas turbine, and power purchased from the grid.

Objective function

$$J = \sum_{k=1}^{24} (1000 c_{grid}(k) W_{grid}(k) + c_f(k) m_f(k)) \tag{5.11}$$



(a) Utility Ontology



(b) Section of the Utility ontology for City of Austin, Texas

Figure 5.13: Utility tariff ontology and snapshot of semantic graph values for City of Austin, Texas

Cooling demand constraint: For all $k = 1, 2, ..., 24$

$$\sum_{i=1}^{n_{Chiller}} Q_{CHW,i}(k) = 1000 Q_{Cooling}(k) \tag{5.12}$$

Electricity Demand Constraint: For all $k = 1, 2, ..., 24$

$$\begin{aligned} W_{grid}(k) + W_{GT}(k) + W_{ST}(k) = \; & W_{Elec}(k) + \tfrac{1}{1000}(W_{P1}(k) + W_{P2}(k)) + \\ & \tfrac{1}{1000}(W_{CHWP}(k) + W_{CWP}(k) + W_{CTF}(k) + \\ & \sum_{i=1} n_{Chiller} W_{COMP,i}(k)) \end{aligned} \tag{5.13}$$

Equation 5.11 is the objective function to be minimized by MPC with the prediction horizon of 24 hours. The first term represents the cost of electricity and the second term is the cost of fuel (gas). Equation 5.12 captures the constraint on meeting the campus cooling demand with the chillers. Equation 5.13 shows the balance between the electricity purchased, produced and consumed. The left hand side represents the total electricity purchased from the grid and generated on campus. The right hand side shows the campus electricity demand, pumps, chilled water plant, cooling tower fan, and chiller electricity consumption. Equations 5.12 and 5.13 are the system constraints for the objective function.

Thermal Energy Storage Dynamics (TES). Figure 5.14 is a schematic of the thermal energy storage system used in this study.

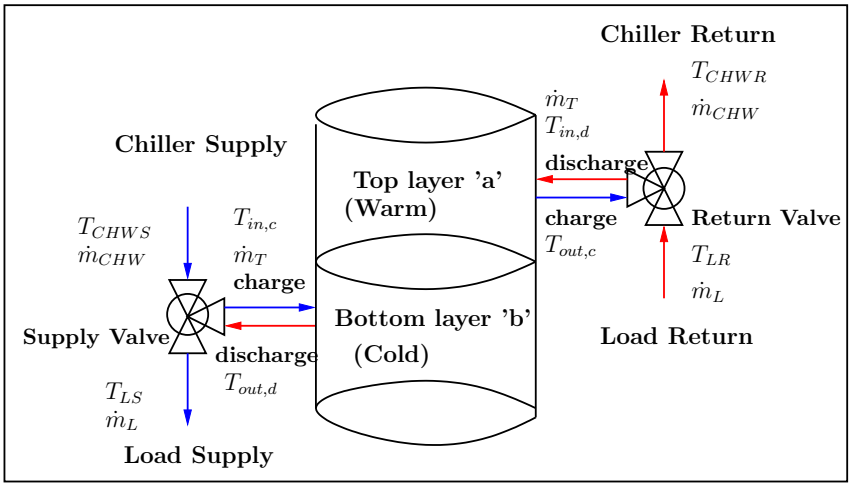


Figure 5.14: Schematic of the thermal energy storage.

The model employs a stratified two layer TES, where T_a and T_b denote the top and bottom layer water temperatures, respectively. The TES is operated in two modes. In charging mode the chiller bank will provide chilled water to the load and the TES. In discharging mode, chilled water from the TES and chiller bank are supplied to the load.

The time variation in thermal energy storage temperature in both charging and discharging modes is given by the following set of equations:

(a) Charging Mode Equations:

Overall Mass Flow Balance

$$\dot{m}_T = \dot{m}_{CHW} - \dot{m}_L \tag{5.14}$$

Top Layer Energy Balance

$$\rho c_{pw} \frac{dT_a}{dt} = f_{a,c} \dot{m}_T c_{pw} (T_b - T_a) + U_c A (T_b - T_a) \tag{5.15}$$

Bottom Layer Energy Balance

$$\rho c_{pw} \frac{dT_b}{dt} = f_{b,c} \dot{m}_T c_{pw} (T_{in,c} - T_b) + U_c A (T_a - T_b) \tag{5.16}$$

Supply Valve Temperatures

$$T_{in,c} = T_{LS} = T_{CHWS} \tag{5.17}$$

Return Valve Temperatures

$$\dot{m}_T T_{out,c} + \dot{m}_L T_{LR} = \dot{m}_{CHW} T_{CHWR} \tag{5.18}$$

(b) Discharging Mode Equations:

Overall Mass Flow Balance

$$\dot{m}_T = \dot{m}_L - \dot{m}_{CHW} \tag{5.19}$$

Top Layer Energy Balance

$$\rho c_{pw} \frac{dT_a}{dt} = f_{a,d} \dot{m}_T c_{pw} (T_{in,d} - T_a) + U_d A (T_b - T_a) \tag{5.20}$$

Bottom Layer Energy Balance

$$\rho c_{pw} \frac{dT_b}{dt} = f_{b,d} \dot{m}_T c_{pw} (T_a - T_b) + U_d A (T_a - T_b) \tag{5.21}$$

Supply Valve Energy Balance

$$\dot{m}_T T_{out,d} + \dot{m}_{CHW} T_{CHWS} = \dot{m}_L T_{LS} \tag{5.22}$$

Return Valve Temperatures

$$T_{in,d} = T_{CHWR} = T_{LR} \tag{5.23}$$

Equation 5.24 illustrates the heat transfer rates in charging and discharging modes. Here, Q_{CHW}, Q_L, Q_T, and δ are chilled water heat transfer, campus demand, thermal storage heat transfer, and the thermal storage control signal, respectively.

$$Q_{CHW} = \delta(Q_L + Q_T) + (1 - \delta)(Q_L - Q_T) \tag{5.24}$$

These equations also serve as constraints in the MPC formulation.

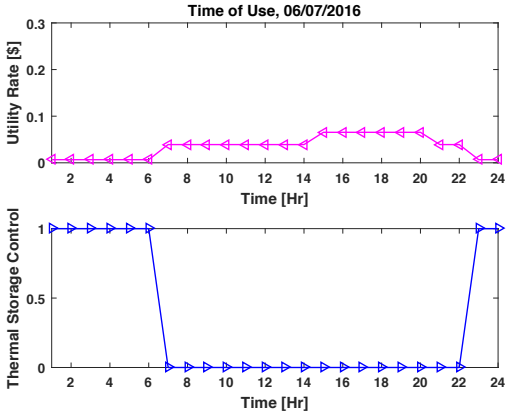
5.2.4 Simulation Results

Figure 5.15 shows the results of the simulation with integrated control. The MPC control method was tested under three different rate tariffs structures associated with Austin, New York City, and San Francisco. The benefit of defining the

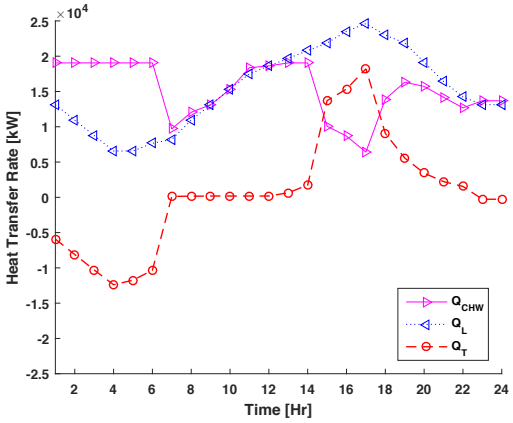
rate tariff in a semantic model and inference-based rules is that the MPC method is unchanged even as the electricity tariff structure is changed. It is important to note that all three case studies use the same temporal logic (rule sets), however, different ontologies are created for each city to represent the semantics of the electricity tariff for that specific city.

Figure 5.15a, bottom, shows the thermal storage control signal (i.e., 1 charge and 0 discharge) based on the inferred electricity rate for the city of Austin. Note that the discharging process begins when the electricity rate increases (depicted on the top), during the on-peak and mid-peak periods. Figures 5.15c and e depict the rate structure and the TES control signal during a specific TOU in NYC and San Francisco, respectively. The impact of the TES control strategy on chiller cooling loads for the city of Austin is shown in Figure 5.15b. Figures 5.15d and f illustrate the chiller, TES and campus heat transfer rates for New York City and San Francisco, respectively. Note the difference between TES heat transfer rates between these three cities. NYC and Austin benefit more from TES during peak periods as compared to San Francisco due to the small deviations between on- and off-peak rates (a flat rate structure).

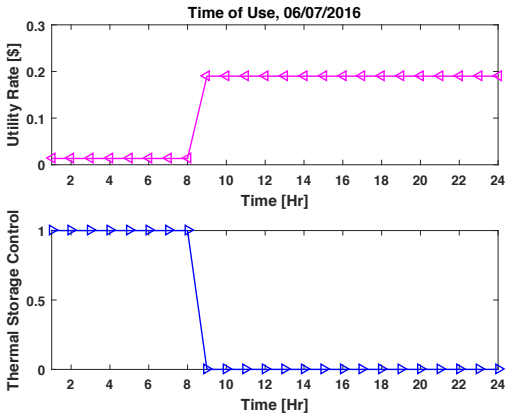
The operational cost of the plant on the simulated day is $26,654, $32,900, $20,700 for NYC, San Francisco, and Austin, respectively. In terms of the simulation time, NYC, with three utility rate variations during a day, requires less computational time than Austin and San Francisco which each have five utility rate variations. The elapsed time using a personal desktop with Core i7-4470 3.4GHz CPU and 32 GB RAM was 263.3, 437.5, and 314.2 seconds for NYC, San Francisco,



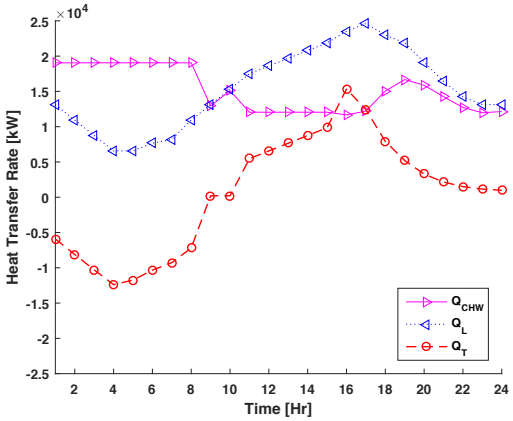
(a) Austin summer TES control



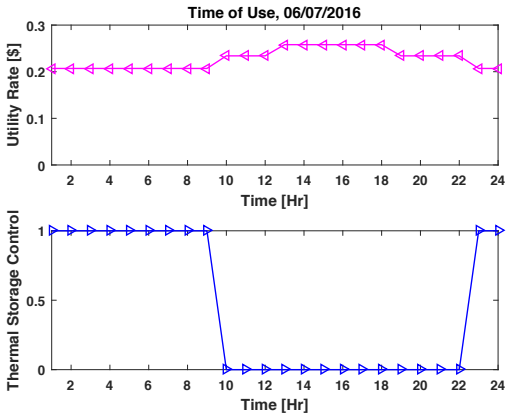
(b) Austin summer Q



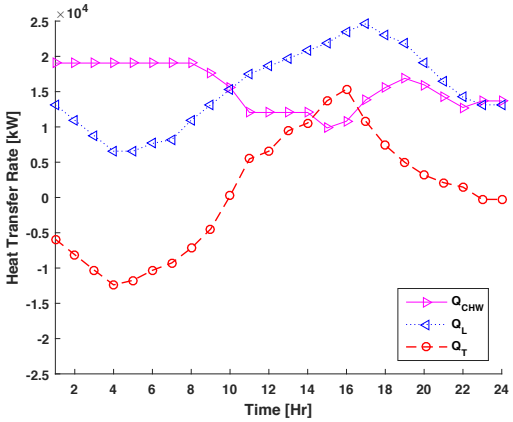
(c) New York City summer TES control



(d) New York City summer Q



(e) San Francisco summer TES control



(f) San Francisco summer Q

Figure 5.15: Simulation results for Austin, New York City and San Francisco; For TES 1=charge and 0=discharge

and Austin, respectively.

5.2.5 Findings

The integrated approach provides a pathway toward robust strategies of control that take into account not only the physical constraints, but also the domain specific constraints and regulations of the operating environment. The numerical experiments indicate that MPC converges faster if inputs and initial conditions for the decision variables are obtained based on inferred results of the semantic rules. In this case application, the predicted electricity rate as well as initial conditions for chiller mass flow rates are obtained from the semantic model.

5.3 Case Study 3: Knowledge-Based Fault Detection and Diagnostics

The third case study application exercises the framework for knowledge-based fault detection and diagnostic analysis (proposed in Chapter 4), by working step-by-step through a scenario triggered by occupant discomfort in a conditioned space. The case study shows how heterogeneous data and knowledge from a variety of sources and domains can be integrated into a single semantic graph, how ontologies and rules can work together to detect the existence of a fault, and then diagnose the causes by systematically considering hypotheses and the supporting evidence.

5.3.1 Problem Description

Figure 5.16 is a plan view of the case study problem setup, consisting a small two-room building architecture, three sensors and three building occupants. Not shown is the mechanical equipment responsible for conditioning the room temperature and achieving acceptable levels of occupant comfort. The mechanical equipment consists of an air handling unit (AHU). The AHU has a coil (i.e., for heating and cooling). The water temperature that flow to the coil is managed by a valve.

Three rules are responsible for the operation and classification of faults in the mechanical equipment:

- Close the valve when the coil temperature is the same as coil setpoint.
- If the valve is shut, the temperature of the air that passes through the coil has to be the same. Otherwise, the valve is leaky

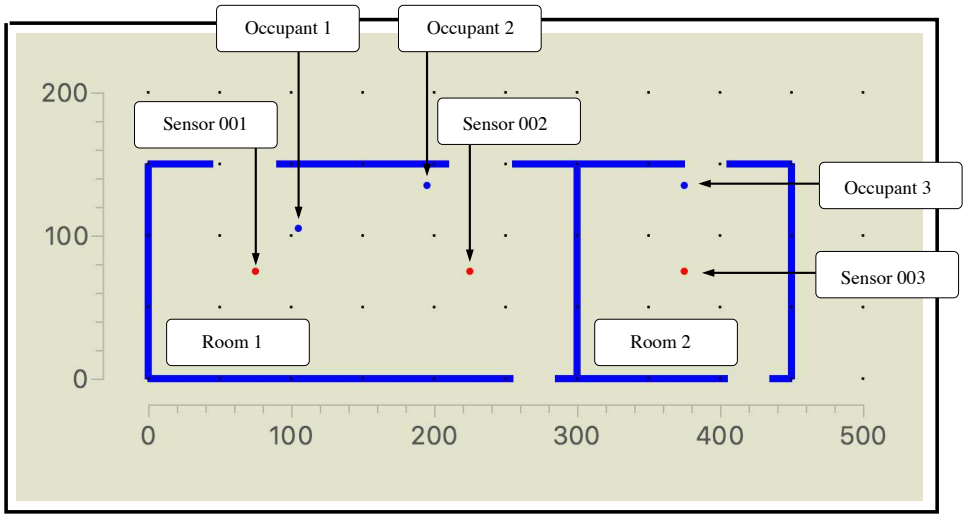


Figure 5.16: Plan view of two-room building architecture, sensors, and building occupants.

- If the a valve fails, the AHU fails too.

One measure to evaluate thermal comfort for the occupants is through computing the thermal sensation as a function of environmental factors such as outdoor and indoor temperature and some personal factors such as clothing levels. A dynamic model to compute thermal sensation (DTS) index to was introduced by Chen and co-workers [27]. According to thermal sensation scale suggested by ASHRAE [1], an acceptable range for occupancy comfort is the interval $[-0.3, 0.3]$. By comparing the current and expected values in a DTS state, the rules in Figure 4.14 will infer the existence of a faulty state, and then systematically examine the evidence associated with each hypothesis to find a root cause.

5.3.2 Snapshot of Semantic Graph Model Assembly

Figure 5.17 shows a snapshot of the building, equipment, sensor, weather, and FDD ontologies integrated together, and populated with system data. The seman-

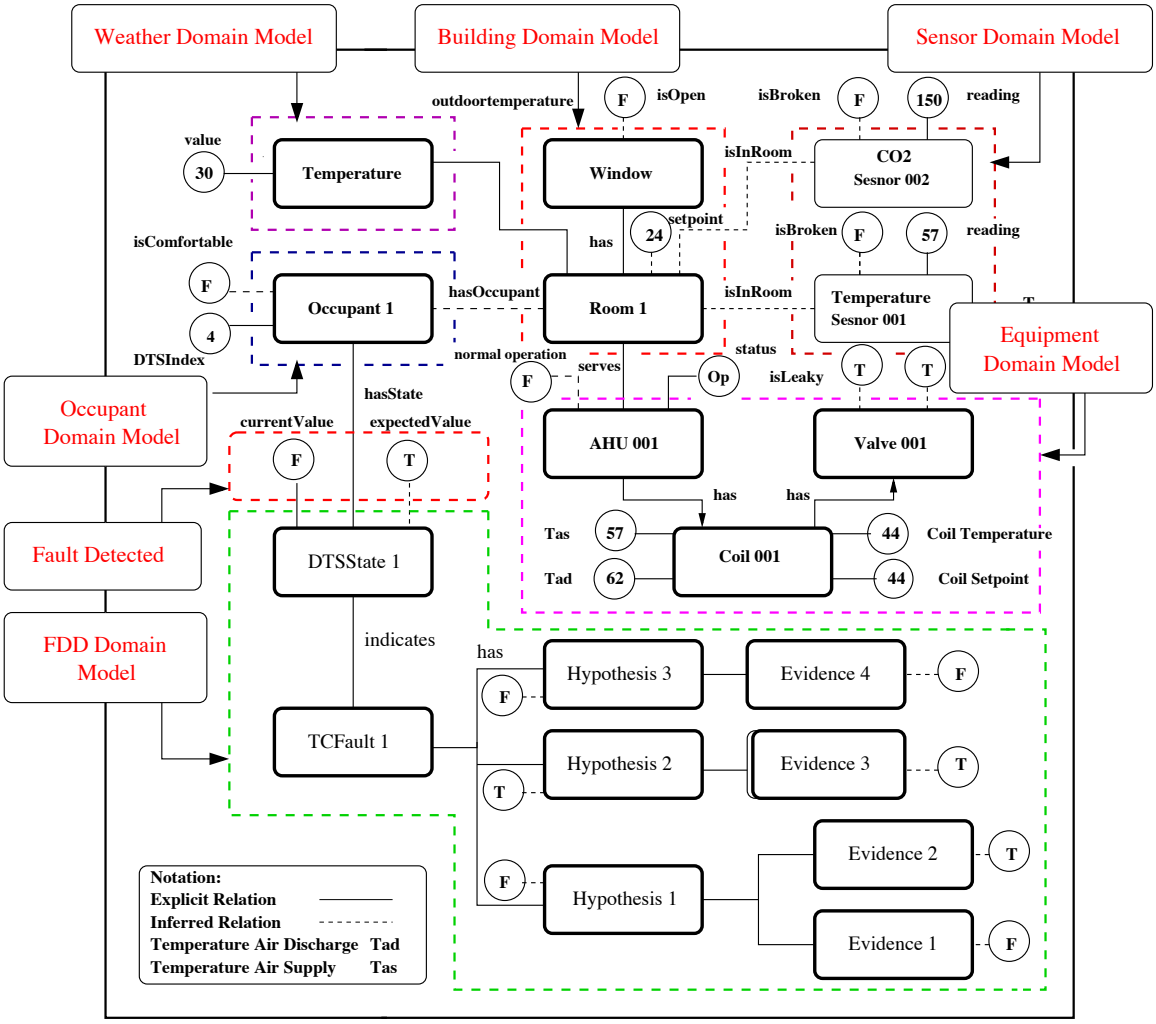


Figure 5.17: Snapshot of fully assembled semantic graph model.

Table 5.1: Instances of states, hypotheses, and evidence for identifying the cause for abnormal occupant thermal comfort value.

Class	**Individual**	**Description**
State	DTSState 1	The DTS index in between $[-0.3, 0.3]$.
Fault	TCFault 1	The DTS index lies outside the interval $[-0.3, 0.3]$ when the air-handling unit is operating.
Evidence	Evidence 1	The CO2 sensor reading is above the normal range the and that shows the window is open.
	Evidence 2	The outdoor temperature is greater than room setpoint.
	Evidence 3	A sensor's reading is outside the range that indicates the sensor is broken.
	Evidence 4	A component is AHU is malfunctioning that results in an abnormal operation of AHU.
Hypothesis	Hypothesis 1	Warm outside air is leaking into the room through an open window –> Supported by Evidence 1 and Evidence 2.
	Hypothesis 2	The serving air-handling unit has abnormal operation. –> Supported by Evidence 4.
	Hypothesis 3	The room sensor that provides feed-back to AHU reaching its target setpoint is broken –> Supported by Evidence 3.

Jena Rules

```
// Evidence Rule 01: A window is open base on C02 concentration in the room.
// ------------------------------------------------------------------------

[ EvidenceRule01: (?cs rdf:type sen:CO2Sensor) (?cs bld:isInRoom ?room)
        (?r bld:hasWindow ?w)(?cs bld:hasReading ?m) lessThan(?m,600)
        greaterThan(?m,400) (?e fdd:hasEvidenceID ?n)
        equal("1"^^xs:integer,?n) ->
        (?w building:isOpen "true"^^xs:boolean) (?e fdd:isTrue "true"^^xs:boolean)  ]

// Evidence Rule 02: Outside temperature is warmer than the setpoint.
// ------------------------------------------------------------------------

[ EvidenceRule02: (?r rdf:type bld:Room) (?r bld:hasSetpoint ?sp)
        (?t rdf:type we:Temperature) (?t we:hasTemperatureValue ?tv)
        greaterThan(?tv,?sp) equal("2"^^xs:integer,?n) (?e rdf:type fdd:Evidence)
        (?e fdd:hasEvidenceID ?n) -> (?e fdd:isTrue "true"^^xs:boolean)  ]

// Evidence Rule 03: Temperature sensor in a room is broken.
// ------------------------------------------------------------------------

[EvidenceRule03: (?ts rdf:type sen:TemperatureSensor) (?ts bld:isInRoom ?room)
        (?ts bld:isBroken ?t)  equal(?t, "true"^^xs:boolean)
        equal("3"^^xs:integer,?n) (?e rdf:type fdd:Evidence)
        (?e fdd:hasEvidenceID ?n ->(?e fdd:isTrue "true"^^xs:boolean) ]

// Evidence Rule 04: Malfunction is in the Air Handling Unit.
// ------------------------------------------------------------------------

[EvidenceRule04: (?AHU rdf:type eq:AHU)
        (?v eq:hasNormalOperationalStatus "false"^^xs:boolean)
        equal(?t, "true"^^xs:boolean)  equal("4"^^xs:integer,?n)
        (?e rdf:type fdd:Evidence)->  (?e fdd:isTrue "true"^^xs:boolean) ]

// FDD Rule 02: Indicate when thermal comfort in a conditioned room has expected value.
// ------------------------------------------------------------------------------------

[FDDRule02: (?AHU rdf:type eq:AHU)(?AHU eq:servesRoom ?r)(?r bld:hasOccupant ?oc)
        (?oc occ:hasDTSState ?dts) (?AHU eq:status ?s) equal(?s "Operating") ->
        print('Expected DTS',?oc)(?dts fdd:hasExpectedValue "true"^^xs:boolean)]
```

Figure 5.18: Fault detection diagnostic rules for operation of a heating coil and for checking evidence 3 and evidence 4.

tic graph model contains instances of ontologies (individuals), relationships among individuals (often spanning domains), and data values associated with various individuals.

From a fault detection and diagnostics standpoint, the main points to note are as follows:

- Occupant 1 is located in Room 1.
- Room 1 has window, a temperature sensor (Sensor 001), and a carbon dioxide sensor (Sensor 002). HVAC services are provided to Room 1 by air handling unit AHU 001. AHU 001 has a coil (Coil 001); Coil 001 has a valve (Valve 001).
- The datatype property for AHU001 “normal Operation” is set to false. This setting is based on the system data and the result of equipment rules 01 through 03 being triggered.
- The setpoint temperature for Room 1 is 24 C, but the current temperature reading for Sensor 001 is 57 C.
- OccupantRule02 sets the ”isComfortable” datatype property for Occupant1 to “false” as the result of a DTSindex value of 4.
- Occupant 1 has dynamic thermal sensation (DTS) state DTSState 1. DTSState 1 indicates a thermal comfort fault (TCFault1), which will be diagnosed by looking at three hypotheses and their supporting evidence.

- The relationship between Hypotheses 1 through 3 and supporting evidence is shown along the bottom of Figure 5.17. Users may query the semantic graph to find the correct hypotheses and valid supporting evidence.

5.3.3 Test Problem Scenario and Hypothesis Evaluation Procedure

The test problem scenario assumes that the numerical value of occupant thermal comfort in a conditioned room has fallen outside the acceptable range. This is detected by FDD Rule 01. With this scenario in place, any one of three hypotheses could potentially be true. To identify the correct hypothesis, the system reasons among the facts and identifies the evidence existing in different domains,

- The outdoor temperature is higher than the setpoint (weather) and the window in the room is open (building, sensor, weather).
- The air-handling unit is malfunctioning (mechanical equipment),
- The room sensor providing feed-back to the air-handling unit to reach its target setpoint is broken (sensor).

As a result, this task will require comprehensive reasoning over multiple domains and identifying the supporting evidence to the most probable hypothesis. To achieve this, we used the proposed framework and implemented ontologies for weather, building, occupant, sensor and equipment domains. The ontologies are populated with data. In general this data will be obtained from simulations or real buildings.

5.3.4 Synthesis of Multi-domain Rules

Table 5.1 describes the instances for key concepts of FDD ontology as they apply to the test case problem, and explains details of the individuals for FDD ontology. For the case study problem, the chain of dependency relationships between hypotheses and supporting evidence is as follows:

- Hypothesis 1 is that warm outside air is leaking into the room through an open window. Evaluation of this hypothesis is supported by execution of two evidence rules, EvidenceRul01 and EvidenceRule02.
- Hypothesis 2 is that the serving air-handling unit has abnormal operation. Evaluation of this hypothesis is supported execution of EvidenceRule04.
- Hypothesis 3 states that the room sensor that provides feedback to AHU reaching its target setpoint is broken. Supporting evidence is provided by the execution of EvidenceRule03.

Figure 5.18 presents the fault detection diagnostic rules for: (1) Operation of a heating coil, (2) Checking evidence 3 and evidence 4, and (3) Detecting when the thermal comfort in a conditioned room matches its expected value.

5.3.5 Multi-domain Rule Evaluation

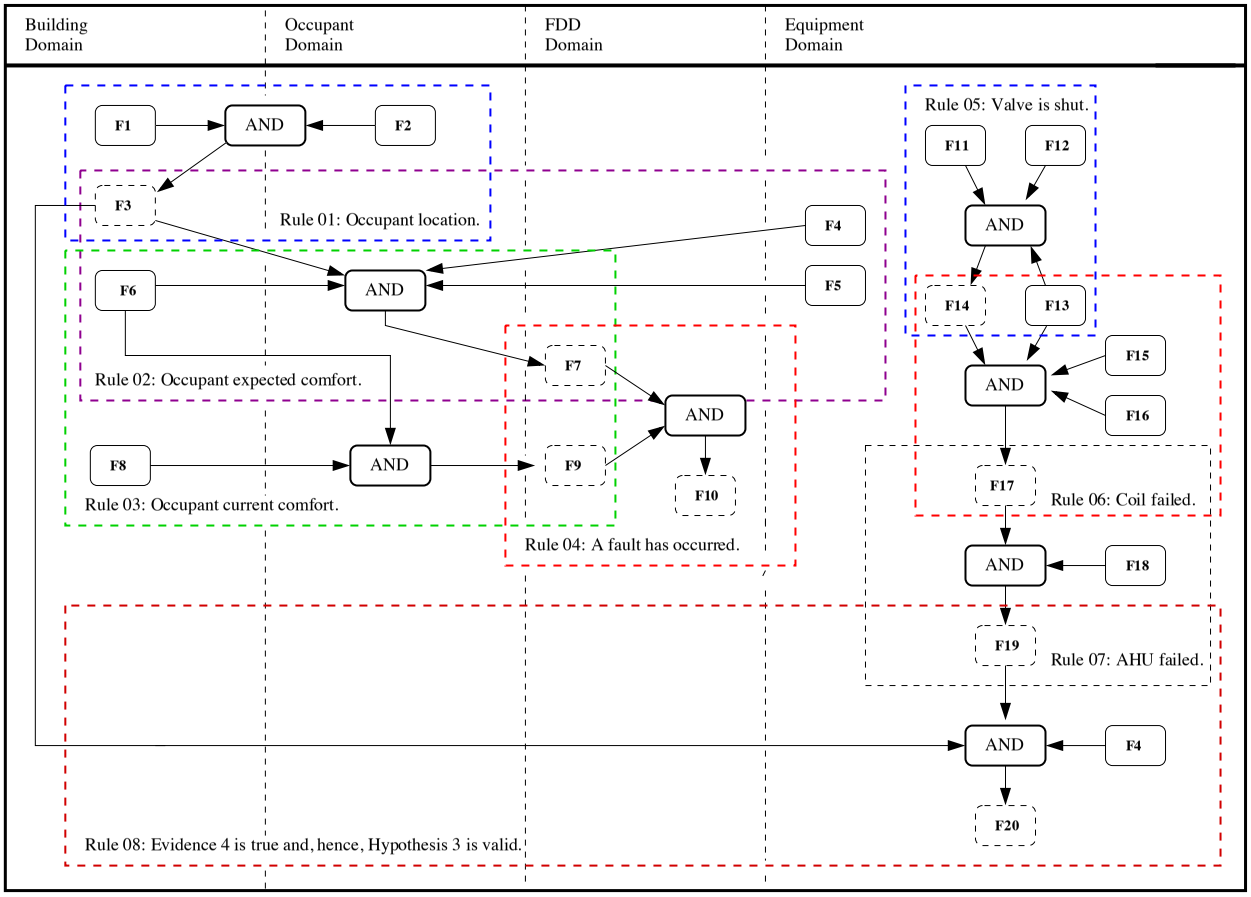
Figure 5.19 shows a snapshot of multi-domain evaluation and forward chaining of rules. From an evaluation standpoint, the eight rules can be clustered into two

pathways, the first focusing on fault detection and the second focusing on diagnostic investigation of probable causes, represented as hypotheses and supporting evidence.

Fault Detection: The first pathway identifies the existence of a fault and is covered by rules 1 through 4:

- Rule 01: Use OccupantRule01 (see Figure 4.17) to determine when an occupant is located in a room.
- Rule 02: Use FDDRule02 (see Figure 5.18) to determine the expected comfort of an occupant.
- Rule 03: Use OccupantRule02 (see Figure 4.17) to determine the current comfort of an occupant.
- Rule 04: Use OccupantRule02 (see Figure 4.17) to compute when a fault has occurred.

determine in which room an occupant is located and whether or not the current value of occupant comfort matches the expected value of comfort. In the snapshot, activation of Rule 01 determines that: Occupant1 is located in Room1. A separate execution would also determine that Occupant2 is also located in Room1. Activation of Rule 02 is based upon the output of Rule 01, state data from the building domain, the relationship of the air handling unit to Room1. In the snapshot trace, the output of Rule 02 states that DTSState for Occupant1 is true and that Occupant1 has a DTSIndex of 4. A fault occurs when there is a discrepancy between the current and



Legend:

F1 = Room hasGeometry
F2 = Occupant hasGeometry
F3 = Room1 has Occupant1
F4 = AHU001 serves Room1
F5 = AHU001 status operating
F6 = Occupant1 hasState DTSState
F7 = DTSState expectedValue true
F8 = Occpant1 hasDTSIndex 4
F9 = DTSState currentValue false
F10 = DTSState indicates DTSFault
F11 = Coil001 CoilSetpoint 44
F12 = Coil001 CoilTemperature 44
F13 = Coil001 hasValve Valve001
F14 = Valve001 isShut true
F15 = Coil001 Tas 62
F16 = Coil001 Tad 57
F17 = Valve001 isShut normalOperation false
F18 = AHU001 hasCoil Coil001
F19 = AHU001 normalOperation false
F20 = Evidence1 isValid true

Figure 5.19: Snapshot of multi-domain evaluation and forward chaining of rules.

expected values of comfort (see F7 and F9), as indicated by the values of current and expected values of DTSState.

Fault Diagnostics: By systematically examining hypotheses and supporting evidence, the second pathway diagnoses the causes of a fault. For the scenario outlined in Figure 5.19, this procedure is covered by rules 5 through 8:

- Rule 05: Use EquipmentRule01 (see Figure 4.10) to determine if a valve is shut.
- Rule 06: Use EquipmentRule02 (see Figure 4.10) to determine if the coil has failed.
- Rule 07: Use EquipmentRule03 (see Figure 4.10) to determine whether or not the air handling unit has failed.
- Rule 08: If EvidenceRule04 (see Figure 5.18) evaluates to true then Hypothesis 3 is true.

The rule for determining whether or not the valve is shut takes input values from the Coil001 CoilSetpoint (44) and CoilTemperature (44) (see F12 and F13), and checks to verify that the coil has a valve. In our scenario, the rule output (F14) is true, indicating that Valve001 is shut, and hence in Rule 06 normal operation evaluates to false. A simple check to verify that the coil belongs to air handling unit AHU001 generates the conclusion that normal operation of the AHU is false (see F19). Finally, input from the room occupancy test and a test to verify that AHU001 is connected to Room1, leads to the conclusion Evidence 4 is supported

and Hypothesis 2 is valid. Finally, we note that except for the room occupancy information feeding into Rule 08, the fault detection and diagnostics pathways operate independently.

5.3.6 Findings

This application served as an example to demonstrate knowledge-based framework for fault detection and diagnostics. The underlying process closely mimics the "thinking process" that humans follow in identifying and diagnosing the causes of a fault. Thus, the steps of gathering data for the participating domains, populating ontologies with individuals, and using rules to detect and diagnose faults and their causes is easy for humans to understand and generally applicable to other domains (e.g., building energy, automotive, health care) for FDD purposes. Capabilities of the prototype implementation have been demonstrated by working step by step through the procedure of detecting and diagnosing the source of faults in an HVAC system.

Key advantages of this approach include: (1) it is decoupled from the system simulation, (2) it is comprehensive, and (3) it is scalable. In fact, the process for expanding an application to include new domains as they come along is very straight forward. The inference-based rules are guaranteed to check at anytime if a changed occurred in an ontology resulting in event-driven fault detection and diagnostic. Finally, inference-based rules provide mechanisms in capturing chain effects that exist in the nature of system failure – for example, if a valve is not operational, the

evidence that AHU is not operating properly also holds true.

Chapter 6: Conclusion and Future Work

6.1 Conclusions

Summary of Work. This dissertation lays the groundwork for a new capability in engineering analysis, where semantic web technologies, languages, and rule sets, are integrated with procedures for knowledge representation and reasoning in real buildings. This knowledge representation will have applications in supervisory control and fault detection and diagnostics in buildings. The scope of investigation included development of ontologies and rule sets to study the potentials offered by these technologies in HVAC systems simulation settings. The ontologies were developed to represent knowledge about the domains essential to building energy such as utility, weather, occupant, equipment, building structure and sensor. Ontologies map data from these domains to familiar concepts that are related to each other and the rule sets provide mechanisms for knowledge expansion in the ontologies. This approach integrates sources of data that are semantically heterogeneous to produce cross-domain information required for decision making and fault detection in HVAC systems. The ontologies have been developed in OWL2, which provides strong support of reasoning in DL. Rules are defined as Jena Rules. Event-based

graph transformations lead to the creation of new knowledge.

Contributions. The contributions of this research are three-fold:

- Developed a semantics-based framework where domain ontologies and rule-sets are created and populated with system data. This framework will transform simulation data to semantic knowledge.
- Leveraged integrative semantic knowledge in MPC controller for supervisory decision making. The knowledge-assisted MPC will integrate descriptive logic with optimization techniques for optimized context-aware control. This control can respond to time and changes in the data as the graph ontologies have the time and data change listeners.
- Developed semantic fault detection and diagnostics framework that mimics a human's thinking to detect a fault and identify the cause of it.

Discussion. The semantic framework is used to enhance building control strategies to make decisions based on a comprehensive semantic knowledge of different domains and implement semantic fault detection and diagnostics techniques that are based descriptive logic formalisms. The rule sets provide mechanisms to integrate the semantic constraints of a certain domain (regulations) with physical constraints described as mathematical equations. This technique is highly scalable since the ontologies are decoupled from the building simulation and it can be extended to include other ontologies. This semantic framework can be utilized in areas such as compliance management for building codes and building-to-grid applications.

6.2 Future Work

The future work will expand current co-simulation strategies to include FMI techniques that are supported by many building simulation tools. Moreover, the semantic framework will utilize new and faster, compared to XML, data format such as JSON (Javascript Object Notation) to extract semantic information of simulation results. The semantic framework will be deployed in environmental chambers and integrated with on-line building control strategies to test the efficacy of the proposed approaches both in building control and fault detection and diagnostics.

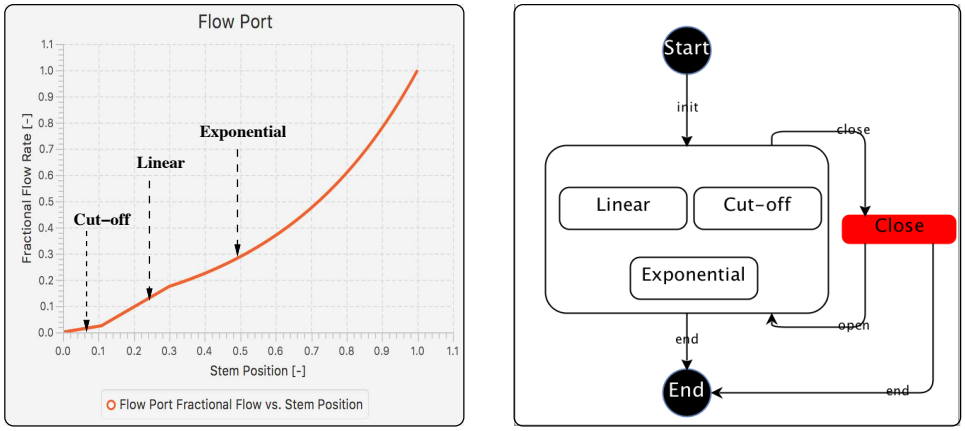


Figure 6.1: Hybrid behavior of a valve.

Further work is needed to improve our capabilities for modeling of components having hybrid behaviors – see, for example, the hybrid behavior model for valve shown in Figure 6.1, and Figure A.7 – including the development of XML markup languages for the model components (e.g., states, events, transitions) and the visual layout of executable statecharts. Our present-day capabilities in this area are slow and tedious.

Moreover, future work will address the uncertainty in automated reasoning. They are different methodologies such as, bayesian inference, for constructing probabilistic arguments about model-based information and knowledge. This capability is very important in fault detection and diagnostics applications.

Ultimately, this approach will be tested in real building application. The challenge in this path is that to ensure the sequence of adding and removing in the rules. The other challenge is that the obtained data may not be a high quality data useful for decision making tasks. In general, ontologies are perfect models for working with data from the Internet of Things (IoT). The semantic framework can receive data from web-based resources. In the future generation of buildings with IoT, these models and web services will be used to unlock value from the vast quantity of data being generated by smart devices.

Chapter A: Systems Integration and Simulation with Whistle

This appendix describes our present-day capabilities for physics-based discrete and continuous behavior modeling of systems with Whistle, an object oriented scripting language used for modeling cyber-physical systems (CPS). Whistle was developed with one observation and one simple idea in mind. First the observation: From the standpoint of CPS design, behaviors in the physical world are constrained by physics (e.g., Newton's laws). Designers have much more freedom to design the cyber world. It follows that if we want to do a better job at CPS simulation and design, then a practical pathway forward is to provide cyber with the mechanisms to be informed about the processes happening in the physical world.

If we were able to design computer languages that understand notions time and space and physical units, then the cyber would certainly be better positioned for decision making which, in turn, would improve correctness of system functionality and performance. We will see how existing modules developed in Java code can be imported into Whistle environment and become part of the admissible syntax. Executable statecharts were integrated with Whistle for modeling the discrete behavior and differential equations were used to capture the transient behavior of the system components.

A.1 Whistle Scripting Language Design

Whistle Scripting Language Design. Scripting languages [98, 127] are designed for rapid, high-level solutions to software problems, ease of use, and flexibility in gluing application components together. Whistle departs from standard scriting languages in that physical units are embedded within the basic data types, matrices, and method interfaces to external object-oriented software packages. Figure A.1 shows, for example, how units are derived in Whistle.

Whistle uses a small number of data types (e.g., physical quantities, matrices of physical quantities, booleans and strings). Features of the language that facilitate the specification of problem solutions include: (1) liberal use of comment statements (as with C and Java, c-style and in-line comment statements are supported), (2) consistent use of function names and function arguments, (3) use of physical units in the problem description, and (4) consistent use of variables, matrices, and looping and branching structures to control the flow of program logic.

Whistle is implemented entirely in Java. It uses the tools JFlex (the Fast Scanner Generator for Java) [72] and BYACC/J (an extension of Berkeley YACC for Java) [23] to handle the parsing and lexical analysis of tokens and statements, Java Collections for the symbol table, and a variety of tree structure representations of the abstract syntax tree. A good introduction to symbol tables and abstract syntax tree representations can be found in the compilers and interpreters text by Mak [89]. Whistle builds upon ideas prototyped in Aladdin [8, 10, 11] a scripting environment

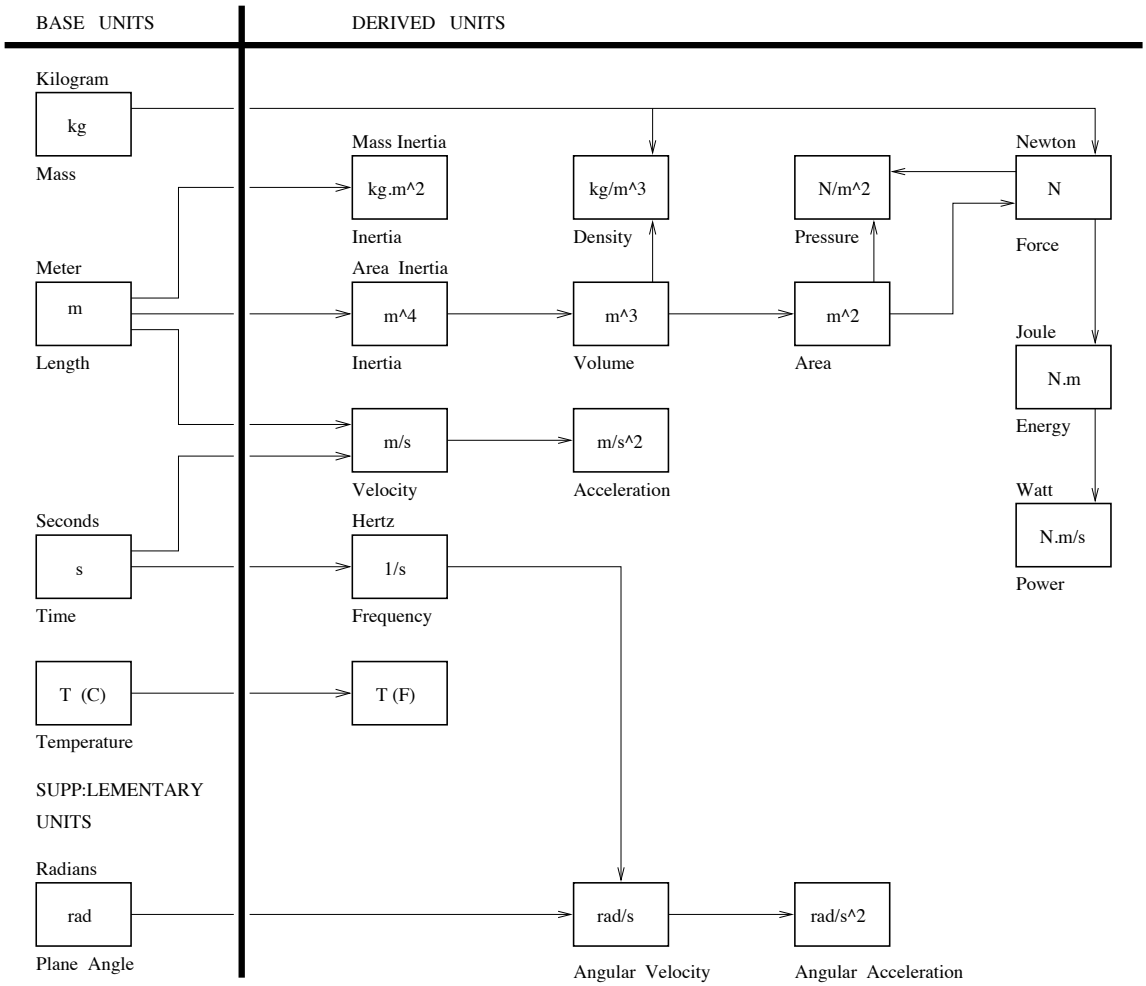


Figure A.1: Primary base and derived units commonly found in engineering mechanics.

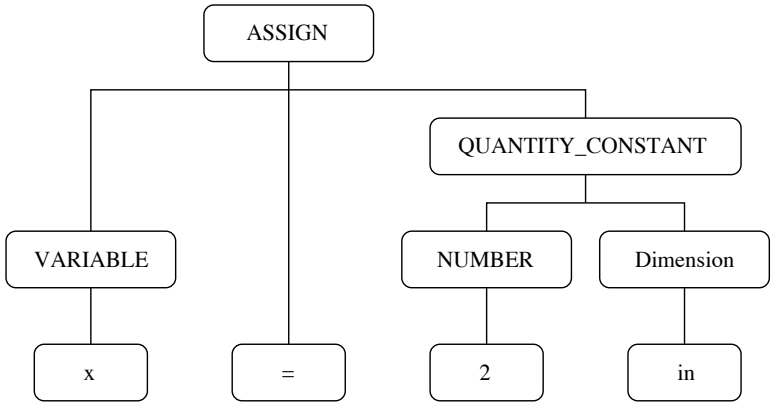


Figure A.2: Parse tree for x = 2 in.

for the matrix and finite element analysis of engineering systems.

A.2 Example 1. Parsing a Simple Assignment Statement

Whistle parses problem specifications into an abstract syntax tree, and then executes the statements by traversing the syntax tree in a well-defined manner. To see how this process works in practice, let's begin by working step by step through the details of processing the assignment statement:

```
prompt >> x = 2 in;
```

Figure A.2 shows the parse tree for this statement. The interpreter parses and stores the character sequence "2 in" as the physical quantity two inches. Notice how `2` juxtaposed with `in` implies multiplication; we have hard-coded this interpretation into the scripting language because `2 in` is more customary and easier to read than `2 * in`. This quantity is discarded once the statement has finished executing. The abstract syntax tree is as follows:

```
Starting PrintAbstractSyntaxTree() ...
======================================== ...

<COMPOUND>
   <ASSIGN>
       <VARIABLE id="x" level="0" />
       <QUANTITY_CONSTANT value="[ 2.000, in]" />
   </ASSIGN>
</COMPOUND>

======================================== ...
Finishing PrintAbstractSyntaxTree() ...
```

Compound statements allow for the modeling of sequences of individual statements. The assignment is defined by two parts, a variable having an identifation "x" and a quantity constant having the value 2.0 in.

```
--------------------------
QUANTITY NAME AND VALUE
--------------------------
Quantity Name  : x
Quantity Value : 0.0508 (m)
-------------------------------------------------------------------------
UNITS
-------------------------------------------------------------------------
Units Name      : "in"     Length Exponent : 1     Temp Exponent   : 0
Units Type      : US       Mass Exponent   : 0     Radian Exponent : 0
Scale Factor    : 0.0254   Time Exponent   : 0
-------------------------------------------------------------------------
```

Table A.1: Symbol table storage for quantity $x = 2$ in.

Internally, the quantity constant is automatically converted to its metric counterpart. Table A.1 shows the name and value of variable "x" as well as details of the units type, scale factor and exponent values.

A.3 Example 2: Oscillatory Flow between Two Tanks

Whistle supports the representation of differential equations in their discrete form, and solution via numerical integration techniques.

As a case in point, the problem of computing the oscillatory flow of fluid between two tanks as illustrated in Figure A.3 can be simulated using Whistle. Let v(t) and Q(t) be the velocity (m/sec) and flowrate (mˆ3/sec) in the pipe, measured positive when the flow is from tank 1 to tank 2. For a pipe cross section, A_p, and tank cross-section areas A_1 and A_2, conservation of mass implies:

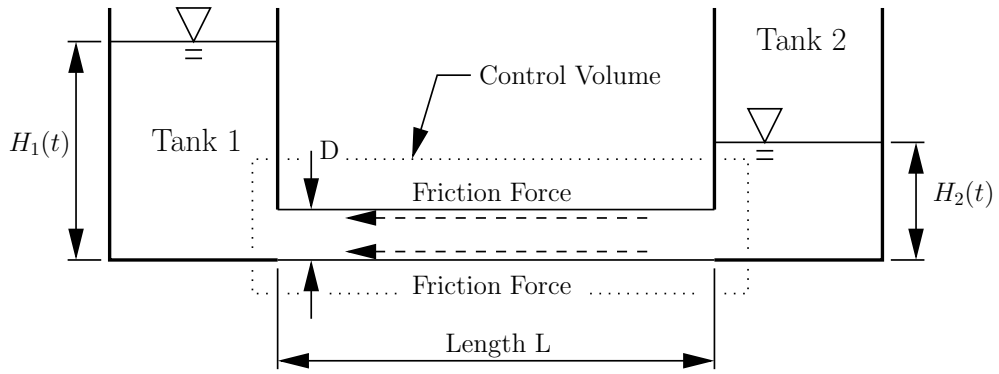


Figure A.3: Summary of forces acting on a pipe element connecting two tanks.

$$Q(t) = A_p v(t) = -A_1 \frac{dH_1(t)}{dt} = A_2 \frac{dH_2(t)}{dt}. \tag{A.1}$$

When water depths $H_1(t) \neq H_2(t)$, a pressure differential will cause fluid to flow through the pipe. Transient behavior of the fluid flow is obtained from the equations of momentum balance in the horizontal direction of the control volume, i.e.,

$$\left[\frac{dv(t)}{dt}\right] + \left[\frac{f_1}{2D}\right] v(t)|v(t)| = \left[\frac{g}{L}\right] [H_1(t) - H_2(t)]\,. \tag{A.2}$$

Notice that each term in equation A.2 has units of acceleration, and that damping forces work to reduce and overall amplitude of accelerations. Damping forces are proportional to pipe roughness and inversely proportional to pipe diameter. The time-history response is computed by creating discrete forms of equations A.1 and A.2, and systematically integrating the first-order equations of motion with Euler integration. First, the update for momentum balance is given by:

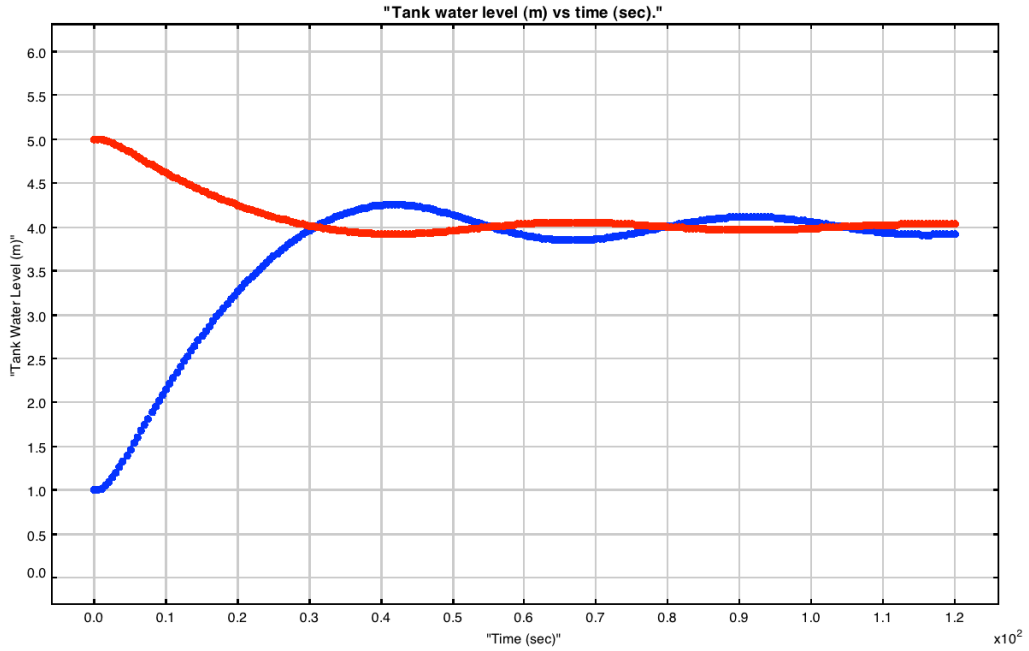


Figure A.4: Tank water levels (m) versus time (sec).

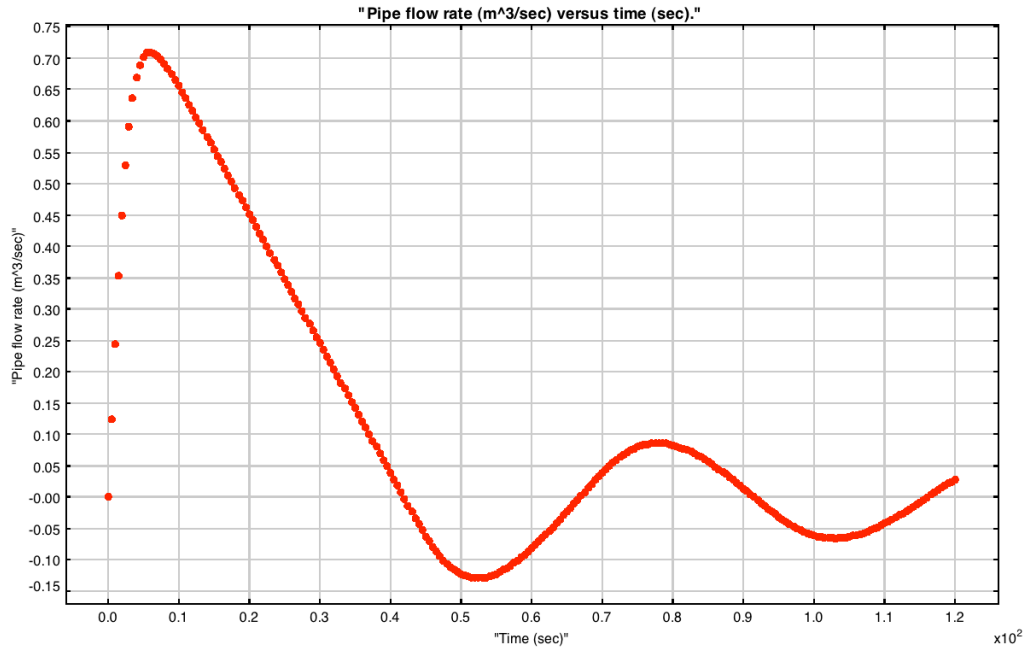


Figure A.5: Volumetric flow rate (m^3/sec) versus time (sec).

$$v(t+dt) = v(t) + \left[\frac{dv(t)}{dt}\right] dt. \tag{A.3}$$

Updates in the water depth for each tank are given by:

$$H_1(t+dt) = H_1(t) - \left[\frac{A_p}{A_1}\right] v(t)dt. \tag{A.4}$$

and

$$H_2(t+dt) = H_2(t) + \left[\frac{A_p}{A_2}\right] v(t)dt. \tag{A.5}$$

If the tank and pipe components are defined as follows:

```
// Define tank and pipe components ....

tank01 = RectangularWaterTank();
tank01.setName("Tank 01");
tank01.setHeight( 10 m );
tank01.setBaseWidth( 3 m );
tank01.setBaseDepth( 5 m );
tank01.setWaterLevel( 5 m );

tank02 = RectangularWaterTank();
tank02.setName("Tank 02");
tank02.setHeight( 5 m );
tank02.setBaseWidth( 2.0 m );
tank02.setBaseDepth( 2.5 m );
tank02.setWaterLevel( 1 m );

pipe01 = Pipe();
pipe01.setLength(  5.0 m );
pipe01.setRadius( 10.0 cm );
pipe01.setRoughness( 0.005 );
```

then the script:

```
velFluid  = pRough/(4*pRadius)*velOld*Abs(velOld)*dt;
velUpdate = g/pLength*( h01Old - h02Old )*dt;
velNew    = velOld + velUpdate - velFluid;
```

shows the essential details of computing the fluid velocity update with Euler integration. During the executable phases of simulation (right-hand side of Figure **??**), the runtime interpreter checks for dimensional consistency of terms in statements before proceeding with their evaluation. Figures A.4 and A.5 are plots of the tank water levels (m) versus time (sec), and volumetric flow rate (m^3/sec) versus time (sec), respectively.

A.4 Example 3: Continuous/Discrete Behavior of a Tank with Water Supply System

An executable statechart package was integrated with Whistle to demonstrate discrete behavior, In this example, modeling flow in a tank with water supply and shut-off valve was explored. This example, adapted from Turns [122], illustrates the steady and transient states of mass conservation and control volume of a tank with a shut-off valve and water supply system.

The system behavior corresponds to four states as follows: (I) The tank is empty, (II) The tank is being filled to a depth of 1 m, (III) The shut-off valve is opened and the water level is decreasing, (IV) The water level in the tank reaches a steady state and does not change. Based on conservation of mass for an unsteady filling process, we obtain the change in water level from equation (A.6),

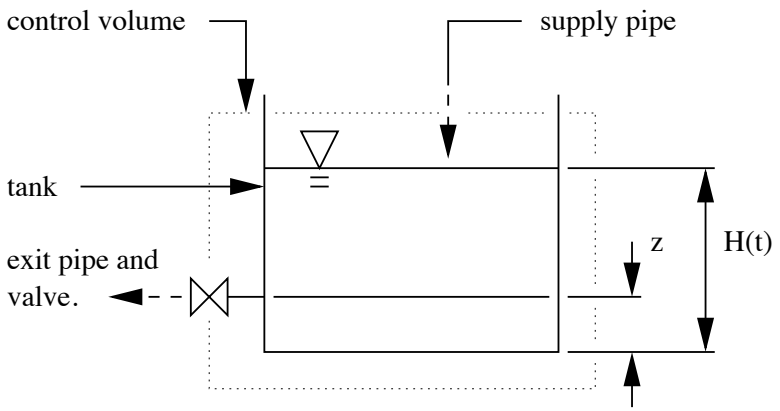


Figure A.6: Front elevation of tank, supply pipe, and exit pipe and valve.

$$\left[\frac{dH(t)}{dt}\right] \rho A_t = \rho v_1 A_1, \tag{A.6}$$

where $H(t)$ is water height in the tank in m, ρ is water density and is equal to $997 kg/m^3$, A_t is cross-section area of the tank in m^2, A_1 is cross-section area of supply pipe in m^2, v_1 is average velocity of inlet water in m/sec. When the water height is 1 m, the shut-off valve opens and the height of water in the tank will be updated based on equations:

$$\left[\frac{dH(t)}{dt}\right] \rho A_t = \dot{m}_1 - \dot{m}_2, \tag{A.7}$$

where $\dot{m}_1$ and $\dot{m}_2$ are the instantaneous mass flow of inlet and outlet pipes in kg/s:

$$\dot{m}_2 = \rho v_2 A_2, \tag{A.8}$$

where A_2 is the cross-section area of the outlet pipe in m^2:

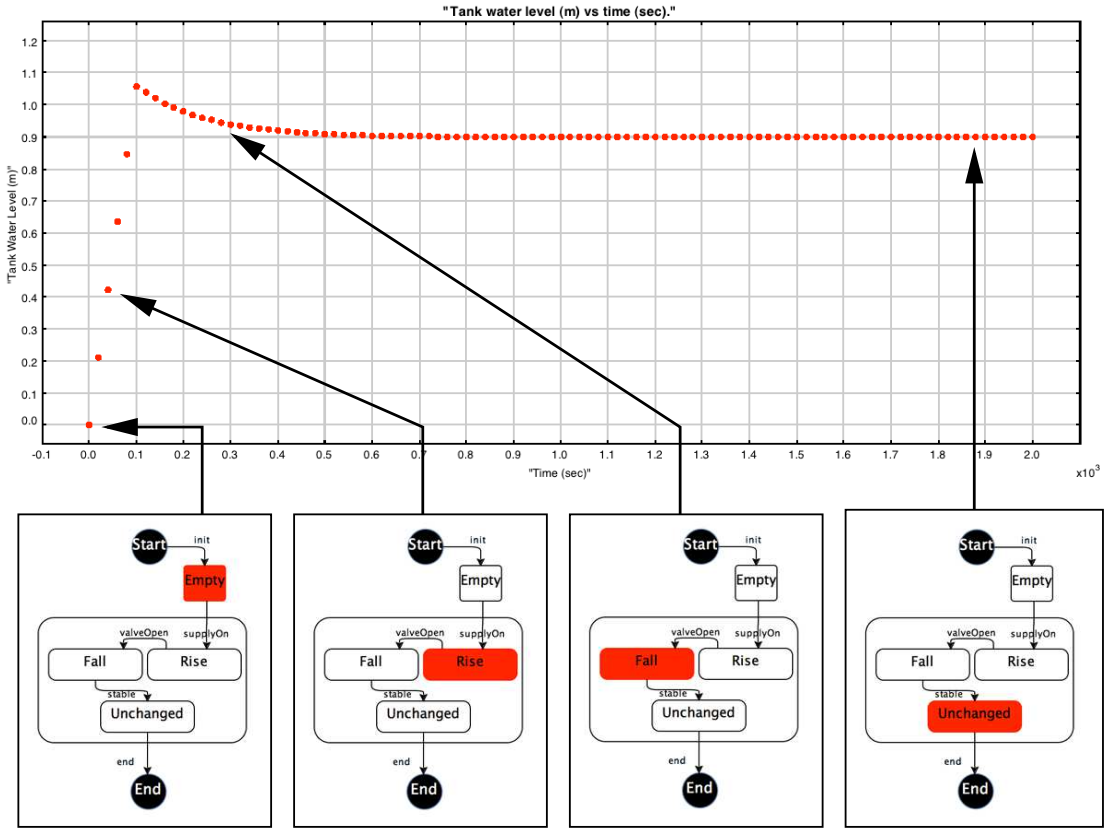


Figure A.7: Time-history response for a tank having a water supply and shut-off valve. Upper plot: tank water level (m) versus time (sec). Lower plot: discrete statechart behaviors at various points in the time-history response.

$$\dot{m}_1 = \rho v_1 A_1, \tag{A.9}$$

$$v_2(t) = 0.85\sqrt{g\,(H(t) - z)}, \tag{A.10}$$

where $v(t)$ is outlet velocity in m/s and z is the location of the shut-off valve in m. In order to mimic the physical equations, we used the scripting language to model components of the tank, supply, and exit pipes with their associated parameters.

The fragment of script below illustrates the essential details of defining the circular water tank and pipe components:

```
// Define tank and pipe components ....

tank01 = CircularWaterTank();
tank01.setName("Tank 01");
tank01.setDiameter( 1*0.15 m);

// Define supply pipe ....

pipe01 = Pipe();
pipe01.setRadius( 10.0 mm );
```

The heart of the time-history simulation is a looping construct that contains two cases (or discrete states) for physical behavior:

```
// Case 1: Water level is below 1 m:

DepthUpdate =  pipe1Velocity * pipe1Area*dt / tankArea;
DepthNew    =  DepthOld + DepthUpdate;
response01 [i][0] = i * dt;
response01 [i][1] = DepthNew;
DepthOld = DepthNew;

// Case 2: Water level is above 1 m:

massFRSupplyPipe = rho*pipe1Velocity * pipe1Area;

velocityExit   = 0.85*Sqrt(g*(DepthOld - 0.1 m));
massFRExitPipe = rho* velocityExit*pipe02.getArea();

massFlowRateCV = massFRSupplyPipe - massFRExitPipe;

dHeight  = massFRCV/(rho*tankArea)*dt;
DepthNew = DepthOld + dHeight;
response01 [i][0] = i * dt;
response01 [i][1] = DepthNew;
DepthOld = DepthNew;
```

Figure A.7 shows the time-history response of the water level in the tank as it transitions from an empty tank to steady state where the water level remains unchanged

t height of 0.9 m. In order to visualize the discrete behavior of this system, we employ our previously developed executable statechart package [37]. This package is capable of modeling and implementation for event-driven behavior with finite state machines. It supports modeling for: (1) Simple, hierarchical and concurrent states, start and final states, (2) History and deep-history pseudostates in hierarchical states, (3) Fork and join pseudostates for concurrent states, (4) Segmented transitions using junction points, and (5) Events, guards and actions for transitions. Visualization of the statechart behaviors is supported through use of mxGraphics in our code. The abbreviated script:

```
import whistle.statechart.TankStatechart;

statechart = TankStatechart();
statechart.startStatechart();
statechart.TransitionEvent(init);

if( DepthOld >= 1 m ){
  statechart.TransitionEvent(valveOpen);
  ....
}
....
```

shows how a statechart element for the water tank is created in an input file developed by the scripting language, and how the language is capable of triggering an event to the statechart when the water level exceeds 1 m. The bottom level of Figure A.7 shows how different regions of continuous behavior correspond to the discrete states in the tank statechart.

Bibliography

[1] ASHRAE Standard 552010 Thermal Environmental Conditions for Human Occupancy, 2010.

[2] U.S. DOE. (2010). U.S. DOE Energy Efficiency and Renewable Energy. Retrieved 2013, from Buildings Energy Data Book, 2010.

[3] Building energy software tools, 2016.

[4] Agarwal, T., Balaji, B., Gupta, R., Lyles, J., Wei, M. and Weng T. Occupancy-Driven Energy Management for Smart Building Automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building (BuildSys 2010)*, pages 1–6, Zurich, Switzerland, November 3-5 2010.

[5] Allen J.F. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[6] Allen J.F. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23(2):123–154, 1984.

[7] Apache Jena:. An Open Source Java framework for building Semantic Web and Linked Data Applications. For details, see https://jena.apache.org/, 2016.

[8] Austin M.A. Matrix and Finite Element Stack Machines for Structural Engineering Computations with Units. *Advances in Engineering Software*, 37(8):544–559, August 2006.

[9] Austin, M.A. and Delgoshaei, P. and Nguyen, A. Distributed System Behavior Modeling with Ontologies, Rules, and Message Passing Mechanisms. *Procedia Computer Science*, 44:373 – 382, 2015. 2015 Conference on Systems Engineering Research.

[10] Austin, M.A., Chen, X.G. and Lin, W.J. ALADDIN: A Computational Toolkit For Interactive Engineering Matrix And Finite Element Analysis. Technical Research Report TR 95-74, Institute for Systems Research, College Park, MD 20742, August 1995.

[11] Austin, M.A., Lin, W.J. and Chen X.G. Structural Matrix Computations with Units. *Journal of Computing in Civil Engineering, ASCE*, 14(3):174–182, July 2000.

[12] Balaji, B., Bhattacharya, A., Fierro, G., Jingkun, G., Joshua, G., Dezhi, H., Aslak, J., Koh, J., Ploennigs, J., Agarwal, Y., Berges, M., Culler, D., Gupta, R., Kjærgaard, M.B., Srivastava, M. and Whitehouse, K. Brick: Towards

a Unified Metadata Schema For Buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys 2016, pages 41–50, New York, NY, USA, 2016. ACM.

[13] Batic, M. , Tomasevic, N. , and Vranes S. . Ontology based Fault Detection and Diagnosis system Querying and Reasoning examples. In *ICKDDM 2015 : 17th International Conference on Knowledge Discovery and Data Mining*, volume 2. International Science Index, Industrial and Manufacturing Engineering, 2015.

[14] Baumgrtel K., and Scherer R. Automatic ontology-based Green Building Design Parameter Variation and Evaluation in Thermal Energy Building Performance Analyses. In *ECPPM 2016, 11th European Conference on Product and Process Modelling*, At Limasol, Cyprus, 2016.

[15] Bayer, T., Dvorak, D., Friedenthal, S., Jenkins, S., Lin C., and Mandutianu S. Foundational Concepts for Building System Models. In *SEWG MBSE Training Module 3, see http://nen.nasa.gov/web/se/mbse/documents*, California Institute of Technology, CA, USA, 2012.

[16] Beetz, J. van Leeuwen, J., and de Vries B. IfcOWL: A Case of Transforming EXPRESS Schemas into Ontologies. 23(1):89–101, 2009.

[17] Berners-Lee, T., Hendler, J. and Lassa, O. The Semantic Web. *Scientific American*, pages 35–43, May 2001.

[18] BioPAX: Biological Pathways Exchange, 1991. For details see: http://www.biopax.org/.

[19] Brooks C., Lee E.A., Liu X. , Neuendorffer S. , Zhao Y., and Zheng H. *Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II).* Technical Report ECB/EECS-2008-28, Department Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, April 2008.

[20] 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).

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

[22] Bushby, S.T. BACnetTM: A Standard Communication Infrastructure for Intelligent Buildings. *Automation in Construction*, 6(5):529 – 540, 1997.

[23] Berkeley Yacc: See http://invisible-island.net/byacc/, (Accessed: August 1, 2013).

[24] Castilla, M., lvarez, J.D., Normey-Rico, J.E., and Rodrguez, F. Thermal Comfort Control using a Nonlinear {MPC} Strategy: A Real Case of Study in a Bioclimatic Building. *Journal of Process Control*, 24(6):703 – 713, 2014.

[25] Chandan V., Do, A.T., Jin, B., Jabarri, F. Brouwer, J. Akrotirianakis I., Chakraborty, A. and Alleyne, A. Modeling and Optimization of a Combined Cooling, Heating and Power Plant System. pages 3069–3074, 2012.

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

[27] Chen, X., Wang, Q., and Srebric J. Occupant Feedback-based Model Predictive Control for Thermal Comfort and Energy Optimization: A Chamber Experimental Evaluation. *Applied Energy*, 164:341 – 351, 2016.

[28] Cigler, J., Prvara, S., Va, Z., Komrkov, D. and ebek, M. Optimization of Predicted Mean Vote Thermal Comfort Index within Model Predictive Control Framework. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3056–3061, December 2012.

[29] Clark D.R. HVACSIM+ Building Systems and Equipment Simulation Program Reference Manual. Technical report, Center for Building Technology, National Bureau of Standards, Gaithersburg, MD 20899, 1985.

[30] Clarke D. *Advances in Model-Based Predictive Control*. Oxford University Press, 1994.

[31] Clarke, J.A. and MacRandal, D.F. The Energy Kernel System: Form and Content. Proceedings of 3rd International IBPSA Conference, Adelaide, Australia, 1993.

[32] Clements-Croome, D.J. *Intelligent Buildings*, chapter Sustainable Healthy Intelligent Buildings for People, pages 1–24. Spon Press (an imprint of Taylor & Francis), ICE, 2013.

[33] Corry, E., Pauwels, P., Hu, S., Keane, M., and O'Donnell J. A Performance Assessment Ontology for the Envioronment and Energy Management of Buildings. *Automation in Construction*, 57:249–259, 2015.

[34] Corsar, D., Markovic, M.o, Edwards, P. and Nelson J.D. *The Transport Disruption Ontology*, pages 329–336. Springer, 2015.

[35] Crawley, D.B. and Hand J.W. and Kummert, M. and Griffith, B.T. Contrasting the capabilities of Building Energy Performance Simulation Programs. *Building and Environment*, 43(4):661 – 673, 2008.

[36] Crawley, D.B., Lawrie, L.K., Winkelmann, F.C., Buhl, W.F., Huang, Y.J., Pedersen, C.O., Strand, R.K., Liesen, R.J., Fisher, D.E., Witte, M.J. and Glazer, J. EnergyPlus: Creating a New-Generation Building Energy Simulation Program. *Energy and Buildings*, 33(4):319 – 331, 2001. Special Issue: {BUILDING} SIMULATION'99.

[37] Delgoshaei, P. Software Patterns for Traceability of Requirements to State Machine Behavior. M.S. Thesis in Systems Engineering, University of Maryland, College Park, MD 20742, November 2012.

[38] Delgoshaei, P. and Austin, M.A. Software Design Patterns for Ontology-Enabled Traceability. In *Conference on Systems Engineering Research (CSER 2011)*, Redondo Beach, Los Angeles, April 15-16 2011.

[39] Delgoshaei, P. and Austin, M.A. Software Patterns for Traceability of Requirements to Finite-State Machine Behavior. In *10th Annual Conference on*

Systems Engineering Research (CSER 2012), St. Louis, Missouri, March 19-22 2012.

[40] 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.

[41] Delgoshaei, P., and Austin, M.A. Framework for Knowledge-Based Fault Detection and Diagnostics in Multi-Domain Systems: Application to Heating Ventilation and Air Conditioning Systems. *International Journal On Advances in Systems and Measurements*, 2017. (In Review).

[42] Delgoshaei, P., Austin, M.A. and Pertzborn A. A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems. *International Journal On Advances in Systems and Measurements*, 7(3-4):223–238, December 2014.

[43] Delgoshaei, P., Austin, M.A., and Veronica, D.A. A Semantic Platform Infrastructure for Requirements Traceability and System Assessment. *The Ninth International Conference on Systems (ICONS 2014)*, February 2014.

[44] Delgoshaei, P., Austin, M.A. and Veronica, D.A. Semantic Models and Rule-based Reasoning for Fault Detection and Diagnostics: Applications in Heating, Ventilating and Air Conditioning Systems. *The Twelth International Conference on Systems (ICONS 2017)*, pages 48–53, April 23-27 2017.

[45] Delgoshaei, P., Austin, M.A., Pertzborn, A., Heirdarinejad, M. and Chandan V. Towards a Cross-Disciplinary Control Strategy for Building Simulations: Integration of Semantic Inference-Based and Model Predictive Control. In *15th International Conference of IBPSA (Building Simulation 2017)*, San Francisco, CA, August 7-9 2017.

[46] Delgoshaei, P., Heidarinejad, M., Ke, X., Wentz, JR., Delgoshaei, P., Srebric, J. Impacts of building operational schedules and occupants on the lighting energy consumption patterns of an office space. *Building Simulation*, 10(4):447–458, Aug 2017.

[47] Dibley, M., Li, H., Rezgui, Y. and Miles, J. An Ontology Framework for Intelligent Sensor-Based Building Monitoring. *Automation in Construction*, 28:1 – 14, 2012.

[48] Dung, T.Q. and Kameyama, W. *Ontology-based Information Extraction and Information Retrieval in Health Care Domain*, volume 4654 LNCS, pages 323–333. 2007.

[49] Energy Information Administration. Commercial Buildings Energy Consumption Survey (CBECS) 2003 Office buildings, 2006.

[50] Faruque, M. and Ahourai, F. A Model-based Design of Cyber-Physical Energy Systems. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 97–104, January 2014.

[51] Feigenbaum L. Semantic Web Technologies in the Enterprise, 2006.

[52] Fiala, D., Havenith, G., Brde, P., Kampmann, B. and Jendritzky G. UTCI-Fiala Multi-Node Model of Human Heat Transfer and Temperature Regulation. *International Journal of Biometeorology*, 2011.

[53] FMI: Funcational Mock-Ip Interface. For details, see: http://fmi-standard.org/, 2017.

[54] Freirea, R.Z., et al. Predictive Controllers for Thermal Comfort Optimization and Energy Savings. *Energy and Buildings*, 40(7):1353–1365, 2007.

[55] Fritzson, P. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3*. John Wiley and Sons, 2014.

[56] Ghaffarianhoseini A., Berardi, U., AlWaer, H., Chang, S., Halawa, E. and Clements-Croome D.C. What is an Intelligent Building? Analysis of Recent Interpretations from an International Perspective. *Architectural Science Review*, 59(5):338–357, 2016.

[57] Grau, B.C., Horrocks I., Motik, B., Parsia, B., Patel-Schneider, P. and Sattler, U. Owl 2: The next step for owl. *Web Semantics*, 6(4):309–322, 2008.

[58] Gruber T.R. Toward Principles for the Design of Ontologies used for Knowledge Sharing? *International Journal of Human-Computer Studies*, 43(5):907 – 928, 1995.

[59] Guo, W. and Zhou, M. Technologies toward Thermal Comfort-based and Energy-Efficient HVAC Systems: A Review. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3883–3888, Oct 2009.

[60] Haarslev, V. and Möller, R. RACER System Description. In *Proceedings of the First International Joint Conference on Automated Reasoning*, IJCAR '01, pages 701–706, London, UK, UK, 2001. Springer-Verlag.

[61] Hafner, I., Rssler, M., Heinzl, B., Krner, A., Breitenecker, F., Landsiedl, M., and Kastner, W. Using BCVTB for Co-Simulation between Dymola and MATLAB for Multi-Domain Investigations of Production Plants. In *Proceedings of the 9th International Modelica Conference*, Munich, Germany, September 2012.

[62] Haines, R.W. and Hittle, D.C. *Control Systems for Heating, Ventilating and Air Conditioning (Sixth Edition.* Boston: Kluwer Academic Publishers, 2003.

[63] Han, J., Jeong, Y.K., and Lee I. Efficient Building Energy Management System Based on Ontology, Inference Rules, and Simulation. In *2011 International Conference on Intelligent Building and Management*, Singapore, 2011.

[64] Han, J., Jeong, Y.K. and Lee I. A Rule-Based Ontology Reasoning System for Context-Aware Building Energy Management. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2134–2142, October 2015.

[65] Han, Y., Hyun, J., Jeong, T., Yoo, J.H. and Hong J.W.K. A Smart Home Control System Based on Context and Human Speech. In *2016 18th Interna-*

tional Conference on Advanced Communication Technology (ICACT), pages 165–169, Jan 2016.

[66] Hayes P. A Catalog of Temporal Theories. Tech Report UIUC-BI-AI-96-01, University of Illinois, 1996.

[67] Heidarinejad, M., Dalgo, D., Mattise, N., Srebric, J. Personalized cooling as an energy efficiency technology for city energy footprint reduction. *Journal of Cleaner Production*, 171(Supplement C):491 – 505, 2018.

[68] 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, March 2010.

[69] Honeywell. *Engineering Manual of Automatic Control for Commercial Buildings: Heating, Ventilating, Air Conditioning.* Minneapolis, MN: Honeywell Plaze, 1989.

[70] Hong T., D'Oca S., Taylor-Lange S.C., Turner W.J.N, Chen Y. and Corgnati S.P. An Ontology to Represent Energy-Related Occupant Behavior in Buildings. Part II: Implementation of the {DNAS} Framework using an {XML} Schema. *Building and Environment*, 94, Part 1:196 – 205, 2015.

[71] Hordeski, M.F. *HVAC Control in the New Millennium.* Lilburn, GA: The Fairmont Press, Inc., 2001.

[72] JFlex – The Fast Scanner Generator for Java: See http://jflex.de/, (Accessed: August 1, 2013).

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

[74] Java Topology Suite (JTS). See http://www.vividsolutions.com/jts/ (Accessed August 4, 2017).

[75] Kalyanpur A., Parsia B., Sirin, E., Grau, B.C. and Hendler, J. Swoop: A Web Ontology Editing Browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144 – 153, 2006. Semantic Grid –The Convergence of Technologies.

[76] Katipamula, S., and Brambley, M.R. Review Article: Methods for Fault Detection, Diagnostics, and Prognostics for Building SystemsA Review, Part I. *HVAC&R Research*, 11(1):3–25, 2005.

[77] Kim, W., and Braun, J.E. Impacts of Refrigerant Charge on Air Conditioner and Heat Pump Performance. In *International Refrigeration and Air Conditioning Conference at Purdue*, July 2010.

[78] Kim, YS., Heidarinejad, M., Dahlhausen, M., Srebric, J. Building energy model calibration with schedules derived from electricity use data. *Applied Energy*, 190:997 – 1007, 2017.

[79] Klein, S.A., Beckman, W.A., et al., 1994. TRNSYS: A Transient Simulation Program, Engineering Experiment Station Report 38-12, University of Wisconsin, Madison.

[80] Kolokotsa, D., Pouliezos, A., Stavrakakis, G., and Lazos, C. Predictive Control Techniques for Energy and Indoor Environmental Quality Management in Buildings. *Building and Environment*, 44(9):1850 – 1863, 2009.

[81] Krachina, O., and Raskin, V. Ontology-Based Inference Methods, CERIAS TR 2006-76, For details, see: https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2006-76-report.pdf, 2006.

[82] Kwak, Y., Huh, J.H., and Jang, C. Development of a Model Predictive Control Framework through Real-Time Building Energy Management System Data. *Applied Energy*, 155(Supplement C):1 – 13, 2015.

[83] Lee E.A., 2003. Model-Driven Development – From Object-Oriented Design to Actor-Oriented Design, Presentation at Workshop for Software Engineering for Embedded Systems, From Requirements to Implementation, Chicago,.

[84] Levenhagen, J.I., and Spethmann, D.H. *HVAC Controls and Systems*. New York: McGraw-Hill, Inc., 1993.

[85] Lord, P., Bechhofer, S., Wilkinson, M.D., Schiltz, G., Gessler, D., Hull, D., Goble, C., and Stein, L. *Applying Semantic Web Services to Bioinformatics:*

Experiences Gained, Lessons Learnt, pages 350–364. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[86] Lu, J., Sookoor, T., Srinivasan, V., Gao, G., Holben, B., Stankovic, J., Field, E. and Whitehouse K. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys 2010)*, pages 211–224, Zurich, Switzerland, November 3-5 2010.

[87] MagicDraw Architecture Made Simple: SysML Metamodel, Version 18.1, No Magic Inc, Allen, Texas, 2015.

[88] Mahdavi, A. and Taheri, M. An Ontology for Building Monitoring. *Journal of Building Performance Simulation*, pages 1–10, October 2016.

[89] Mak R. *Writing Compilers and Interpreters: A Software Engineering Approach (Third Edition)*. Wiley Publishing Inc, 2009.

[90] Memoori: Smart Buildings StartUps and their Impact on Smart Buildings, 2017. For details, see: https://www.memoori.com/portfolio/startups-impact-smart-buildings-2017/.

[91] Merdan M. Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain. *Ph.D. Dissertation, Vienna University of Technology*, 2009.

[92] Modelica Buildings Library. 2017. Open Source Library for Building Energy and Control Systems Analysis and Simulation. For details, see http://simulationresearch.lbl.gov/modelica/.

[93] Motik, B., Shearer, R. and Horrocks, I. *Optimized Reasoning in Description Logics Using Hypertableaux", bookTitle="Automated Deduction – CADE-21: 21st International Conference on Automated Deduction Bremen, Germany, July 17-20, 2007 Proceedings*, pages 67–83. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[94] Nassif, N., Kajl, S. and Sabourin, R. Optimization of HVAC Control System Strategy Using Two-Objective Genetic Algorithm. *HVAC&R Research*, 11(3):459–486, 2005.

[95] Nicolai, A., and Paepcke, A. Co-Simulation between Detailed Building Energy Performance Simulation and Modelica HVAC Component Models. In *Proceedings of the 12th International Modelica Conference*, pages 63–72, Prague, Czech Republic, May 2017.

[96] Nouidui, T.S., Wetter, M., and Zuo, W. Functional Mock-Up Unit Import in EnergyPlus for Co-Simulation. In *Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association*, pages 3275–3282, Chambery, France, August 2013.

[97] O'Connor, M. and Amar, D. SQWRL: A Query Language for OWL. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED'09, pages 208–215, Aachen, Germany, Germany, 2009. CEUR-WS.org.

[98] Osterhout, J.K. *Tcl and the Tk Toolkit*. Addison-Wesley Professional Computing Series, Reading, MA 01867, 1994.

[99] Otter M., Arzen, K.E., and Dressler I. StateGraph A Modelica Library for Hierarchical State Machines. In *Proceedings of the 4th International Modelica Conference*, Hamburg, Germany, March 7–8 2005.

[100] Petnga, L., and Austin, M.A. Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems. *Procedia Computer Science*, 16:403 – 412, 2013. 2013 Conference on Systems Engineering Research.

[101] Petnga, L., Austin, M. Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems. *Procedia Computer Science*, 16(Supplement C):403 – 412, 2013. 2013 Conference on Systems Engineering Research.

[102] Privara, S., Siroky, J., Ferkl, L., and Cigler J. Model Predictive Control of a Building Heating System: The First Experience. *Energy and Buildings*, 43(23):564 – 572, 2011.

[103] Protege: A Free, Open-Source Ontology Editor and Framework for Building Intelligent Systems. For details see: http://protege.stanford.edu/.

[104] Purdon, S., Kusy, B., Jurdak, R. and Challen G. Model-Free HVAC Control using Occupant Feedback. In *38th Annual IEEE Conference on Local Computer Networks - Workshops*, pages 84–92, 2013.

[105] Prez-Lombard, L., Ortiz, J. and Pout, C. A Review on Buildings Energy Consumption Information. *Energy and Buildings*, 40(3):394 – 398, 2008.

[106] Randell D.A., Cui Z., Cohn A.G. A Spatial Logic based on Regions and Connectivity, 1994. Division of Artificial Intelligence, School of Computer Studies, Leeds University.

[107] Sagerschnig, C., Seerig, A., Gyalistras, D., Prvara, S., and Cigler, J. Co-Simulation for Building Controller Development: The Case Study of a Modern Office Building. In *CISBAT 2011 Conference*, September 2011.

[108] Sahlin, P., Eriksson, L., Grozman, P., Johnsson, H., Shapovalov, A. and Vuolle, M. Will Equation-Based Building Simulation Make It? Experiences From The Introduction of Ida Indoor Climate and Energy. Eighth International IBPSA Conference, Eindhoven, Netherlands, 2003.

[109] Scherer, W.T., and White, C.C. Survey of Expert Systems for Equipment Maintenance and Diagnostics, 1989.

[110] Schumann, S., Hayes, J., Pompey, P. and Verscheure O. Adaptable Fault Identification for Smart Buildings. In *2011 AAAI Workshop (WS-11-07)*, 2011.

[111] Siegel, J.A., and Wray, C.P. An Evaluation of Superheat-based Refrigerant Charge Diagnostics for Residential Cooling Systems. *ASHRAE Transactions*, 108(2):965 – 975, 2002.

[112] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. and Katz, Y. Pellet: A Practical OWL-DL Reasoner. Web Semantics: Science, Services and Agents on the World Wide Web, Volume 5, Issue 2, June 2007, Pages 5153, 2007.

[113] Sowell, Edward F and Haves, Philip. Efficient Solution Strategies for Building Energy System Simulation. *Energy and Buildings*, 33(4):309 – 317, 2001. Special Issue: {BUILDING} SIMULATION'99.

[114] Staroch P. A Weather Ontology for Predictive Control in Smart Homes, 2013. M.S. Thesis in Software Engineering and Internet Computing, Vienna University of Technology.

[115] SWEET: Semantic Web for Earth and Environmental Terminology, Jet Propulsion Laboratory, CA, For details see: https://sweet.jpl.nasa.gov/.

[116] Taswell C. DOORS to the Semantic Web and Grid with a PORTAL for Biomedical Computing. *IEEE Trans Inf Technol Biomed*, 12(2):191–204, 2008.

[117] Terkaj W. and Sojic A. Ontology-based Representation of IFC EXPRESS Rules: An Enhancement of the ifcOWL Ontology. *Automation in Construction*, 57:188 – 201, 2015.

[118] 2006. Time Ontology in OWL. For details, see: http://www.w3.org/TR/owl-time/ (Accessed 09/20/2017).

[119] Trcka, M., Hensen, J.L.M. and Wetter, M. Co-Simulation of Innovative Integrated HVAC Systems in Buildings. *Journal of Building Performance Simulation*, 2(3):209–230, 2009.

[120] TRNSYS: The Transient Energy System Simulation Tool. See: http://www.trnsys.com/ (Accessed September 8, 2017)., 2017.

[121] Tsarkov, D. and Horrocks, Ian. *FaCT++ Description Logic Reasoner: System Description*, pages 292–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[122] Turns S.R. *Thermal-Fluid Sciences: An Integrated Approach.* Cambridge University Press, 2006.

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

[124] Valiente-Rocha, P.A. and Lozano-Tello, A. Ontology-Based Expert System for Home Automation Controlling. Universidad de Extremadura, Cceres, SPAIN.

[125] Vazquez, S., Leon, J.I., Franquelo, L.G., Rodriguez, J., Young, H.A., Marquez, A. and Zanchetta, P. Model Predictive Control: A Review of Applications in Power Electronics. *IEEE Industrial Electronics Magazine*, 8(1):16–31, March 2014.

[126] Wagner, D.A., Bennett, M.B., Karban, R., Rouquette, R., Jenkins, S. and Ingham, M.O. An Ontology for State Analysis: Formalizing the Mapping to SysML. In *Proceedings of 2012 IEEE Aerospace Conference*, Big Sky, Montana, March 2012.

[127] Wall, L., Christiansen, T., and Schwartz, R. *Programming Perl.* O'Reilly and Associates, Sebastopol, CA 95472, 2nd edition, 1996.

[128] Wang S. Editorial: Enhancing the Applications of Building Automation Systems for better Building Energy and Environmental Performance. *HVAC&R Research*, 12(2):197–199, 2006.

[129] Wang, S. and Jin, X. Model-based Optimal Control of {VAV} Air-Conditioning System Using Genetic Algorithm. *Building and Environment*, 35(6):471 – 487, 2000.

[130] Wang, S. and Ma, Z. Supervisory and Optimal Control of Building HVAC Systems: A Review. *HVAC&R Research*, 14(1):3–32, 2008.

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

[132] Weather API. See https://openweathermap.org/api (Accessed September 14, 2017).

[133] Weiser M. The Computer for the 21st Century. *Scientific American*, pages 94–104, September 1991.

[134] Wetter M. Co-simulation of Building Energy and Control Systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation*, 4(3):185–203, 2011.

[135] Wetter, M., and Haugstetter C. Modelica Versus TRNSYS A Comparison Betweeb Anequation-Based and a Procedural Modeling Language For Building Energy Simulation. In *Second National IBPSA-USA Conference*, pages 262–269, Cambridge, MA, August.

[136] Wetter, M., and Haves P. A Modular Building Controls Virtual Test-Bed for the Integration of Heterogeneous Systems. In *SimBuild 2008*, Berkeley, CA, August 2008.

[137] Wetter, M. and Zuo, W. and Nouidui, T.S. and Pang, X. Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4):253–270, 2014.

[138] Wetter, M., Zuo, W., Nouidui, T.S. and Pang X. Modelica Buildings Library. *Journal of Building Performance Simulation*, April 2013.

[139] Wicaksono, H., Aleksandrov, K. and Rogalski, S. *An Intelligent System for Improving Energy Efficiency in Building Using Ontology and Building Automation Systems*, chapter Automation, pages 1–14. Spon Press (an imprint of Taylor & Francis), London and New York, March 2012.

[140] Wiggins, M. and Brodrick J. HVAC Fault Detection. *ASHRAE Journal*, 54(2):78 – 80, 2012.

[141] Windham, A.W. A Multi-Agent Decision Process for Controlling Heating, Ventilation, and Air-Conditioning Systems, 2014. Ph.D. Dissertation in Architectural Engineering, The Pennyslvania State University, University Park, PA 16802.

[142] Zaheer-Uddin, M. and Zheng, G.R. Optimal Control of Time-Scheduled Heating, Ventilating and Air Conditioning Processes in Buildings. *Energy Conversion and Management*, 41(1):49 – 60, 2000.

[143] Zuo, W. and Wetter, M. and Tian, W. and Li, D. and Jin, M. and Chen, Q. Coupling Indoor Airflow, HVAC, Control and Building Envelope Heat Transfer in the Modelica Buildings library. *Journal of Building Performance Simulation*, 0(0):1–16, 2015.