A PERENNIAL SIMULATION FRAMEWORK FOR INTEGRATED CRISIS MANAGEMENT STUDIES

by

SETH N. HETU
(*B.Sc., Rensselaer Polytechnic Institute*)

A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

April 2013

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this thesis.

This thesis has also not been submitted for any degree in any university previously.

Seth N. Hetu
22 April 2013

Acknowledgements

With thanks to my supervisor, Associate Professor Gary Tan, for his advice and guidance, and to my review committee (Associate Professor Teo Yong Meng and Associate Professor Chan Mun Choon) for their valuable feedback.

With thanks to my parents, family, Myat Aye Nyein, Heather Scoffone, the staff at CREATE, and all the wonderful people I have met in Singapore for (in no particular order) their love, support, advice, friendship, and professionalism, and for a book sent at just the right time.

With mention and thanks to Associate Professor Abhik Roychoudhury for helping me to continue my research at a difficult time.

Contents

NATIONAL UNIVERSITY OF SINGAPORE

Abstract

School of Computing
Department of Computer Science

Doctor of Philosophy

by Seth N. Hetu

An abstract of the thesis of Seth N. Hetu in partial fulfillment of the requirements for a Ph.D. in Computer Science, presented April 2013.

Title: A Perennial Simulation Framework for Integrated Crisis Management Studies

This thesis presents a *perennial simulation framework* that targets the trans-disciplinary field of *crisis management simulation*. The state of the art in crisis management recognizes a broad spectrum of tasks, categorized as *hindsight*, *foresight*, or *decision support*, with the ultimate goal of achieving information superiority over a given crisis. Computer simulation is invaluable in this regard, but the development of comprehensive, modern simulations for crisis management is stymied by the stringent requirements of the latter. Our research provides a robust framework which reflects the state of the art in both fields, in addition to exploiting recent novelties such as virtual worlds and symbiotic simulation.

We use the term *perennial simulation* to refer to any integrated, symbiotic simulation created by our framework that targets multiple physical or virtual worlds, and is flexible in its capacity to support hindsight, foresight, and decision support studies. In order to establish the context of perennial simulations, we first provide a lifecycle analysis of a typical perennial system. Next, the framework is detailed at both a conceptual level and as an implementation, followed by a series of experiments which test the capabilities of the framework. The first of these employs a perennial simulation to test users' response to egress advisories during a building evacuation. In addition, a novel configuration of our framework called MMOHILS is used to overcome weaknesses in traditional agent-based simulation through an appeal to virtual worlds. The second study focuses on mining traffic data from video feeds in an effort to determine the benefits of adding a perennial component to a traditional simulation environment. A side goal is to successfully integrate *legacy models* into our framework without restricting

their access to sensor data. The final study created a prototype perennial system that targets an existing sensor-enabled building for the purpose of enhancing "building sweep scenarios" for mixed-reality participants. This serves as an instructional overview of the framework's practical usage, with an emphasis on using an established sensor test-bed. Throughout these studies, validation and scalability concerns are addressed.

Results indicate that the perennial simulation framework is suitable for crisis management simulation studies. Live exercises demonstrated symbiotic simulation's efficacy for building egress scenarios, and scalability tests confirm that this technique can easily accommodate 100 agents in a world of arbitrary size. Symbiotic simulation was shown to be practical within the tight time constraints of crisis management, and a technique that trades accuracy for performance was demonstrated. Simulations created with the perennial framework were demonstrated to have a clear benefit to decision makers even under increased sensor-level uncertainty. Finally, validation techniques for agents in crisis-relevant scenarios were presented, and a rigorous practical validation of our egress MMOHILS was performed. Considered collectively, our experiments demonstrate the capacity for trans-disciplinary crisis management simulation evident in our framework.

List of Tables

List of Figures

List of Publications

MMOHILS: A Simpler Approach to Valid Agents in Human Simulation Studies
Seth N. Hetu and Gary Tan
In WSC '08: Proc. of the 40th Conference on Winter Simulation, pp. 909-913.
Winter Simulation Conference, 2008
ISBN 978-1-4244-2708-6

Real-Time Simulation in Java: A Feasibility Study
Seth N. Hetu and Gary Tan
In System Simulation and Scientific Computing, ICSC 2008, pages 396-399.
Asia Simulation Conference, 2008
ISBN 978-1-4244-1786-5

Proper Handling of Real Players in Serious Gaming Studies
Seth N. Hetu and Gary Tan
In Learn to Game, Game to Learn; the 40th Conference ISAGA
International Simulation And Gaming Association, 2009
ISBN 978-981-08-3769-3

Potential Benefits of Symbiotic Simulation to Pedestrian Evacuation
Seth N. Hetu and Gary Tan
In Asia Simulation Conference 2009
Japan Society for Simulation Technology, 2009

The Big Picture of Symbiotic Decision Support: Designing a "What-If" Simulation Framework for Crisis Management
Seth N. Hetu and Gary Tan
In FISAT: Second International Conference on Advanced Computing and Communications Technologies for High Performance Applications
FISAT, 2010, *Keynote*

Perennial Simulation of a Legacy Traffic Model: Implementation, Considerations, and Ramifications
Seth N. Hetu and Gary Tan
In WSC 11: Proceedings of the 43rd Conference on Winter Simulation
Winter Simulation Conference, 2011
ISBN 978-1-4577-2107-6

Application of Symbiotic Decision Support to Managed Evacuation Studies Using a Perennial Framework
Seth N. Hetu and Gary Tan
In Asia Simulation Conference, 2011
Korea Society for Simulation (KSS), 2011
ISBN 978-4-431-54215-5

Chapter 1

Introduction

1.1 The Magnitude of Preparedness

Crisis management is a field characterized by the stark contrast between pairs of similar crises at different points in time. Such "before and after" comparisons evince the radical effect that proper handling of a critical situation can have on lives saved, property salvaged, and health risks ameliorated.

Consider the deadliest natural disaster in the history of the United States: in 1900, a category four hurricane swept into the coastal city of Galveston, flooding the area and leading to six thousand deaths. The residents of Galveston had been concerned about hurricanes striking the city, but were nonetheless ill-prepared for the disaster. Compounding the situation with grim irony, the Galveston Weather Bureau (GWB) section director had publicly stated only nine years earlier that "it would be impossible for any cyclone to create a storm wave which could materially injure the city" and recommended not to build a seawall [1]. Reacting to this disaster, the GWB immediately reversed its position and pushed forward with plans to strengthen the city against future hurricanes. A five meter high seawall was constructed, and the entire city was elevated several meters more using dredged sand. A mere fifteen years later, Galveston was struck by a storm of the exact same strength. This time, there were only fifty-four deaths [2].

The story of Galveston is a triumph, but what about crises with more far-reaching consequences and fewer directly obvious solutions? The last century of influenza epidemics offers some relevant historical knowledge about these types of endeavors. One of the worst modern outbreaks was the Spanish Flu, a particularly virulent disease which claimed roughly 50 million lives worldwide in three outbreaks between 1918 and 1919 [3]. Fast-forward to 2007, and densely packed urban areas combined with easy access to

intercontinental flights have created a situation ideal for spreading disease. Yet despite being physiologically similar to the deadliest pandemic in history, various descendants such as the Avian flu and the Hong Kong flu have wreaked far less havoc on the world's population. Certainly some of this is beyond the realms of organized response; for example, the H1N1 virus had a lower infection rate among people over 40 due to resistances developed from past exposure to flus. In addition to simple luck, though, several deliberate disease control techniques have also had an impact. Vaccines are now developed quickly and deployed globally. At the same time, various non-vaccination policies such as *contact tracing* and *quarantine* have proven to be extremely effective in stymieing pandemics. The former provides decision support to health officials at the time of crisis, and the latter can actually restrict the spread of viruses with long incubation periods and parallel development of symptoms and susceptibility [4]. To emphasize, vaccines can be combined with these techniques to boost the efficacy of the combined response effort.

Although progress is usually reactionary, sometimes the risk of a disastrous outcome is enough to inspire preventative action. Such is the case with traffic control systems in dense urban environments, where congestion and reckless driving can amass and lead to deadly consequences. The city of New York has collected traffic statistics for slightly over a century. During that time, traffic fatalities have decreased *in total* from 471 to 209 despite the population doubling [5]. Other cities were forced to modernize more rapidly. Public safety concerns leading up to the 1984 summer Olympics prompted Los Angeles to invest heavily in a then-untested automated traffic control system called ATSAC. This system monitored and adjusted traffic lights at 118 intersections, providing real-time statistics and allowing administrators to manually override signal timings if such direct control was necessary. In total, a record-breaking 5.7 million Olympic tickets were sold that year, adding to the 7 to 8 million already living in the city (although there was certainly some overlap). Against this incredible population crunch, the ATSAC system was successful at minimizing congestion —so successful, in fact, that it was immediately expanded to four times its original size. This new system paid for itself in a year, and has been expanded now to cover the entire city [6] [7] [8].

All three cases share a similar theme: the magnitude of preparedness to mitigate a crisis. When we think of crisis management, we often think of grandiose examples such as the first one, and indeed such broad strokes are often required to combat the immediate event. Galveston was able to strengthen itself against hurricanes by understanding the nature of the crisis (i.e., that hurricane damages are caused by *storm surges* rather than high winds) and by applying a straightforward mechanical solution. The influenza example, on the other hand, stressed the importance of maintaining an "information

superiority" of sorts throughout the development of the crisis. *Contact tracing*, *quarantine*, and vaccination can have radically different costs and benefits depending on the nature of the epidemic. Being able to accurately estimate their effects and tradeoffs is invaluable to anyone in a decision-making capacity. In the case of traffic planning and preparedness, a clear understanding of the problem *before* it developed into a single catastrophic event was enough to prompt New York and Los Angeles to employ preventative solutions. Learning from past crises (*hindsight*), dealing with a crisis as it develops (*decision support*) and planning for future crises (*foresight*) are three key goals of crisis management, and will be a recurring element of this thesis.

In addition to demonstrating the inherent *variety* of crisis management, the three anecdotes just presented also confirm its *complexity*. Shoring up Galveston's defenses was a straightforward, localized effort, while effective *contact tracing* can require massive centralized information systems. Similarly, managing traffic in New York at the turn of the 20^{th} century demanded far less sophistication than automating signal timings in Los Angeles eighty years later. Ending back where we began, in Galveston, one might note that modern hurricane tracking systems and community training exercises have done at least as much as sea walls in terms of saving lives. Communication and collaboration are required to defend against any modern crisis.

1.2 Trends in Crisis Management Simulation

A common technology used to perform crisis management research is *computer simulation*. Indeed, most crisis-related fields have embraced simulation to some degree. Hospitals simulate patient flow through emergency rooms in an attempt to learn what happens upon reaching peak capacity. Fire spread models are applied to past crises to determine how different building designs might have aided evacuation or impeded fire spread. Even community training exercises benefit from having a central simulation one can query about the current state of the virtual crisis. A comprehensive assessment of crisis management asserts that crises are best managed by acting on all possible intervention points before, during, and after a crisis [9]. Assuming that one can be created, a complete simulation environment is very useful in this regard, as it provides a rigorous, robust framework for coordinating response while minimizing uncertainty.

Although effective, such comprehensive approaches can be challenging to realize through simulation, which has its own requirements and restrictions. Figure 1.1 depicts a seemingly-credible visualization of Detroit's infamous housing and crime situation in the early 1970's. Despite its appearance, this visualization was actually extracted from the computer game *Micropolis*, and it is merely a facsimile of the true economic reality

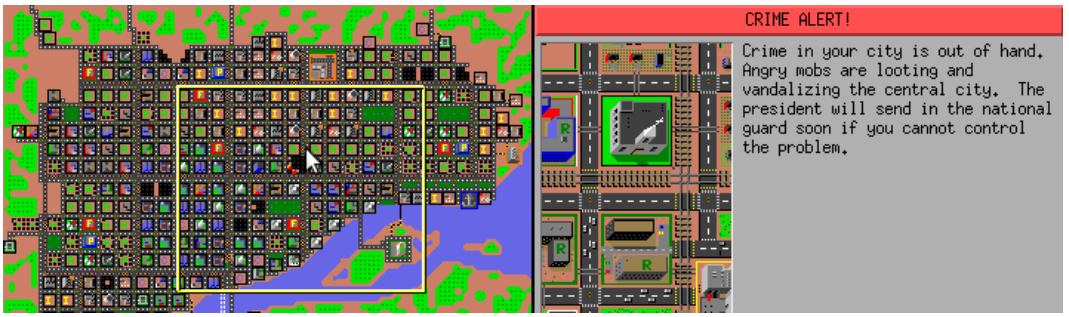

FIGURE 1.1: A scene from 1972 Detroit, Michigan, as visualized in *Micropolis* (from the same source code as *Sim City*). This game is often misconstrued as a simulation by the general public; in actuality, its academic credibility is negligible.

of Detroit at the time. *Verification* and *validation* are two key tasks which distinguish computer simulation from other software development endeavors. Each individual *model* which composes the simulation must be validated, as must the entire interconnected system. Not surprisingly, these tasks increase in difficulty as the system grows in size. Producing a valid system at the time of crisis is challenging, as is maintaining a long-running simulation without sacrificing validity. Some systems are, by design, easier to validate than others. The ATSAC system, for example, lists as a key feature its computation of real-time traffic flow statistics. These are used to evaluate the performance of the system, and can be compared against the original signal strategy as a means of hypothesis verification. This automatic confirmation of expectations is reminiscent of a technique from *symbiotic simulation*, which will be introduced in Chapter 2. It demonstrates that care must be taken while constructing a system to ensure that it has the means to remain relevant over time.

Unfortunately, modern developments in simulation come with their own challenges. The ATSAC's use of real-time sensing and feedback may help prepare it to function as a symbiotic simulation, but such systems are often costly to implement and maintain. Other modern techniques such as agent-based simulation enable new research of more complex heterogeneous interactions, but feature additional challenges regarding validation. In particular, human behavior under certain conditions may be difficult to measure quantitatively, frustrating efforts at empirical validation [10]. This is discussed more fully in Section 3.4. Finally, the paradox of new techniques is that they tend to obviate previous work which has already proven its worth. Any attempt to improve the field of simulation must avoid cutting off the past several decades of progress as a necessary requirement.

Solving these issues is crucial to enabling practical, credible systems which make full use of the benefits of simulation. A commitment to verified, valid models is what distinguishes the simulation sciences from traditional software development where "good enough" is considered acceptable. Many of the tools used to assist crisis management

are based on crude or outdated technology, partly because the risk of potentially invalid results from newer, untested systems is simply too high. Discerning how to apply the science of simulation to the field of crisis management in a way that maximizes both software reuse and validity is a challenging task that we will set out to accomplish and describe in this thesis.

1.3 The Path Towards a Comprehensive Solution

The primary problem our research is trying to address is that simulation for crisis management currently lacks a comprehensive, conceptual framework that meets its needs as a trans-disciplinary field. (The full extent of this problem is presented in Chapter 2). We approached this problem from a modeling and methodology point of view. The primary goal will be to develop a framework which encapsulates the necessary aspects of simulation reuse for *foresight*, *hindsight*, and *decision support* studies. This framework will be designed to operate within the restrictive demands of crisis management systems, but it will also be applicable to simulation in general. Such a system will necessarily take a long view in its approach; as we shall see, some researchers have made progress towards resolving various pieces of the problem, but the state of the art is nowhere near a comprehensive solution. Rather than focusing on one key problem area and solution, we will attempt to generalize our framework in a way that maximizes its potential for conceptual reuse, as well as providing a non-trivial amount of library-level reuse.

Once a clear foundation has been established, we will demonstrate a best-case *reference implementation* of the framework given the current technology available to simulation scientists. Moving from a purely theoretical framework to an implementation will necessitate that trade-offs are made. The ubiquitous decision in computer science between performance and memory utilization will require careful deliberation. In addition, several design decisions specific to simulation will require our attention. We will justify these when appropriate. Finally, as we build the implementation, we will test its efficacy —and, by association, that of the framework— in a series of real-world simulation studies. Each of these will be designed to stress a different aspect of the simulation framework.

In addition to our primary goal of developing a framework, we are also interested in exploring new research opportunities enabled through the incorporation of useful cross-domain technologies such as virtual reality. We are particularly interested in the possibility of using virtual environments populated by physical (human) users to capture input in situations which would otherwise require approximation. Thus, we are not developing new behavioral models for humans, but rather providing a mechanism by

which such models can be created in previously inaccessible circumstances of interest. Another tangential goal is the ability to incorporate imperfect video information into our simulation in real time as a means of maximizing existing infrastructure utilization. Chapter 2 lists all major and minor contributions, and any additional novelties will be covered as they become relevant in the succeeding chapters. In order to properly establish the scope of these contributions, our framework and the experiments it enables will be evaluated in comparison to similar existing technologies, when such systems exist.

As a tertiary goal, we will also consider the performance implications of the framework and its various configurations. Several of the latter involve the use of *virtual worlds*, leading us to investigate the limits on perennial simulations. In particular, we investigate the upper bound on world size, simultaneously connected users, and the accuracy of symbiotic simulation versus its performance.

Finally, we are concerned with the ability of any new system, including our own, to function as well as possible with the abundance of existing models and simulations. Any new system will necessarily obviate some amount of previous work; it is our goal to provide some means of backwards compatibility which allows legacy systems to interoperate to some useful degree with new systems designed with our framework.

1.3.1 Objectives

The goals discussed in the previous section will now be consolidated into the objectives of this research. These objectives are, in order of importance:

- To formulate a **perennial simulation framework** which bridges the transdisciplinary gap between simulation and crisis management. This system will contain elements which ensure its applicability across all levels of crisis response.
- To develop an implementation of this framework and use this to test its limits pertaining to crisis management. The exploration of side goals such as virtual world interaction and symbiotic simulation's efficacy in particular are considered part of this objective.
- To develop general techniques for crisis management simulation which help to expand its applicability despite the real-world challenges faced. In particular, the incorporation of real-time data and the difficulty in modeling human behavior in crisis-relevant situations will be discussed.

1.3.2 Introduction of Perennial Simulation

Previous sections have referred to a framework with **perennial** characteristics. The use of this term is specific to this thesis and its contributing research, and was justified as a means of distinguishing the framework from similar techniques with different focuses (see Section 3.9). The lexicological motivation behind the term *perennial* is its emphasis on "persistent, enduring" and "regularly repeated" processes [11] —qualities which the perennial simulation framework attempts to incorporate. With this in mind, we define the perennial simulation framework as follows:

Perennial Simulation Framework

> The perennial simulation framework enables the creation of robust, long-running simulation systems which target physical/virtual locations and their interactions. These simulations are flexible in their capacity to provide foresight, hindsight, and decision support studies, particularly under the tight time constraints inherent in crisis management. The integrated nature of this framework allows more accurate modeling of human agents in novel situations through the use of a technique called MMOHILS (discussed later).

Given this definition, we refer to simulations created by our framework as **perennial** in nature, or as having **perennial** elements. An important clarification to the remainder of this thesis is that the term "perennial simulation" does not connote a new field to rival that of simulation, and that when we discuss perennial simulation in comparison to traditional simulation, we are merely employing a useful shorthand to talk about "simulations not created by our framework that are lacking integrated, symbiotic elements" versus "simulations created by our framework that feature integrated, symbiotic elements". The value of the perennial simulation framework is the greater ease it affords in the creation and maintenance of perennial simulations, while the simulations themselves primarily feature the ability to meet the trans-disciplinary needs of crisis management simulation.

1.4 Thesis Outline

The remainder of this thesis will proceed as follows:

- **Chapter 2** will cover relevant background information, setting the problem in its proper context and defining the shape of the solution as well as listing specific contributions.

- **Chapter 3** presents a full summary of all related work in the fields of crisis management, computer simulation, and various minor relevant areas.
- **Chapter 4** details the proposed framework, its various interacting components, and its intended usage. A sample implementation is also provided.
- **Chapter 5** covers the various studies undertaken to show the efficacy of the proposed framework. Each of these tests a particular component of the overall framework or implementation. An explanation of results obtained accompanies all reported data.
- **Chapter 6** concludes the thesis.

Chapter 2

The Path Towards a Solution

2.1 The Trans-Disciplinary Nature of Crisis Management Simulation

In Chapter 1, simulation was offered as a technology for enabling powerful crisis management studies. Simulation is suitable for managing the increased complexity inherent to these studies, in addition to providing a level of formalization which is missing from mere *ad-hoc* solutions. Unfortunately, several issues complicate the reality of this dependency. To begin with, simulation cannot simply be "applied" to a given crisis management task without first satisfying its myriad requirements: in particular, verification, validation, the incorporation of real-time data, and the analysis of sensitivity. Additionally, as simulation is repeatedly applied to an ever-increasing number of crisis management studies, it will undoubtedly generate new techniques which must be incorporated back into the field of simulation. Finally, as the domain evolves, care must be taken to ensure that any borrowed techniques are modified to maximize reuse without sacrificing accuracy. For example, agent-based simulation —a technique borrowed from the field of artificial intelligence— lacks the flexibility to deal with novel study environments without also risking validity. This minor point must be addressed before the value of agent-based simulation can become fully exploited by the cross-domain field of crisis management simulation.

In fact, crisis management simulation is far beyond a multidisciplinary domain —it is a true *trans-disciplinary* field, in that it crosses into disciplines beyond the academic domain and may require "extensive interaction between the developers and the end

users" [12]. Figure 2.1[1] outlines the two domains, listing key components in blocks I and II. Each domain can be seen as the side to a cube, with block III enumerating the crossover field of "Simulation for Crisis Management". A "borrowed" component in Block I originated in a field other than than simulation, but was later incorporated due to its perceived utility to simulation scientists. Each component will be examined more thoroughly in the following sections.

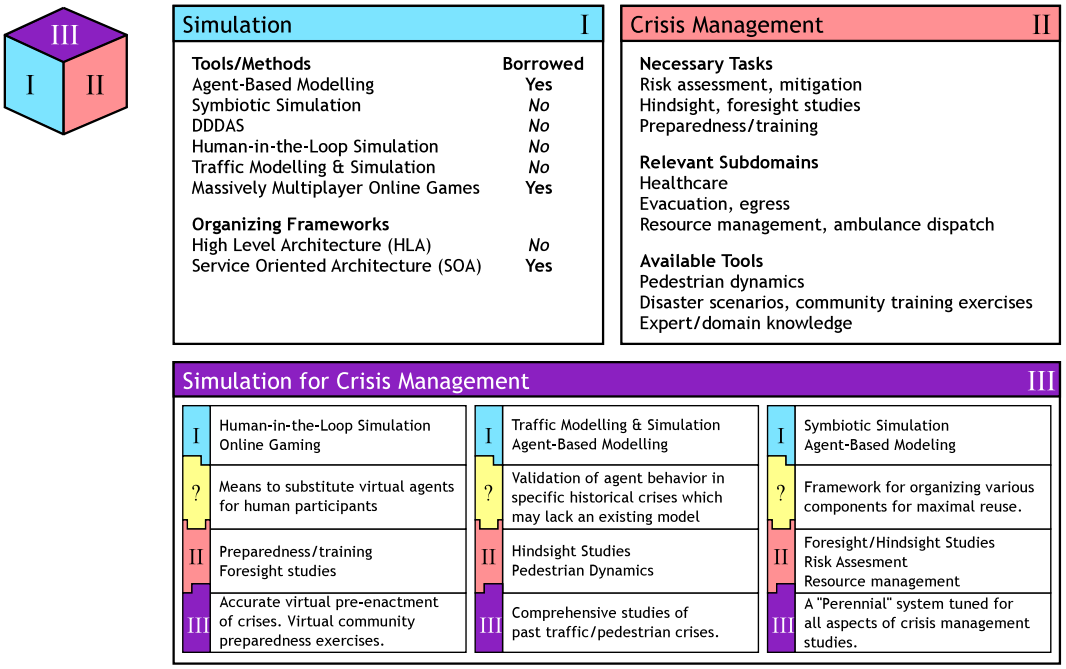

FIGURE 2.1: Breakdown of the trans-disciplinary overlap between simulation and crisis management. Some simulation techniques originated in a different domain; these are marked as "borrowed" in Block I. Unknown components for trans-disciplinary compatibility are marked with a "?" in Block III.

2.1.1 The Science of Simulation

The reader is expected to be familiar with simulation in general, and relevant work in the field will be covered in Chapter 3. For completeness, we will provide a minimal overview of simulation; [15] is recommended for in-depth coverage targeting novices to the field.

A *simulation* is a "model of a real or imagined system [designed for] conducting experiments" [16]. Simulation is used when experimenting with the physical system directly is too expensive or otherwise impractical. Simulation requires *models* of the systems

[1] An attempt was made to color-code all critical information in this thesis in such a way that readers with color vision deficiency and related vision impairments will be able to distinguish it. (See: [13] and [14]) Please contact the author if you are nonetheless unable to view this document properly.

under investigation, and the entire application must undergo vigorous phases of *verification* and *validation* to ensure that it is an acceptable approximation of the system being modeled.

Lacking sophisticated technology, simulation can be done manually or via spreadsheets. Spreadsheet simulation in particular occupies a research niche which is still being explored. New developments in this area includes the parallelization of Excel-based models on a grid [17], integrating better state-space searching for supply chain models [18], and a push for Monte Carlo spreadsheet simulation as an easily accessible tool for finance and marketing [19]. For the most part, however, the field of simulation has come to mean exclusively *computer simulation*, in which the various models and connective components are realized using a *simulation programming language* or with help from a *simulation library*. Computer simulation enables processing of significantly more complex interactions, such as "human-in-the-loop" simulation (Figure 2.1, Block I), which leverages a real-time, highly interactive simulation to train a user in a complicated or otherwise dangerous task. In addition, computer simulation can readily "borrow" interesting techniques from other fields in computer science, encouraging cross-domain research and ensuring the field will never grow stale. A good example of this is the work done by developers of Massively Multiplayer Online (MMO) games. Research on distributed simulation —performed by the military and academia— ran in parallel to research for online games —performed by private corporations. Each of these groups had their own design goals, leading to the development of vastly different solutions. Recently, several researchers have started importing the work done regarding online games into the field of simulation, leading to systems that are cheaper to develop and more compatible with general-purpose programming languages and commodity hardware.

A technique called *symbiotic simulation* is both relatively new and comparatively niche; as such, even domain experts may require a brief overview. Introduced early into the 21^{st} century, a symbiotic simulation is defined as a continuously-executing simulation which attempts to optimize a corresponding physical system in a way that is mutually beneficial [20] [21]. As depicted in Figure 2.2, this requires constant monitoring of physical sensors. A controller will periodically dispatch multiple "What-If?" simulations, the results of which are analyzed and used to predict the future behavior of the system. At this point, the system may be adjusted through the use of *effectors*, with the intent of optimizing its behavior. All predictions can be validated over time, allowing the system to double-check the efficacy of its proposed solutions. The power and automation afforded by symbiotic simulation cements its place as a key component in our proposed solution.

As a result of the constantly increasing size of the field of simulation, several *organizing frameworks* were developed to manage its complexity. Two of these are listed in Figure

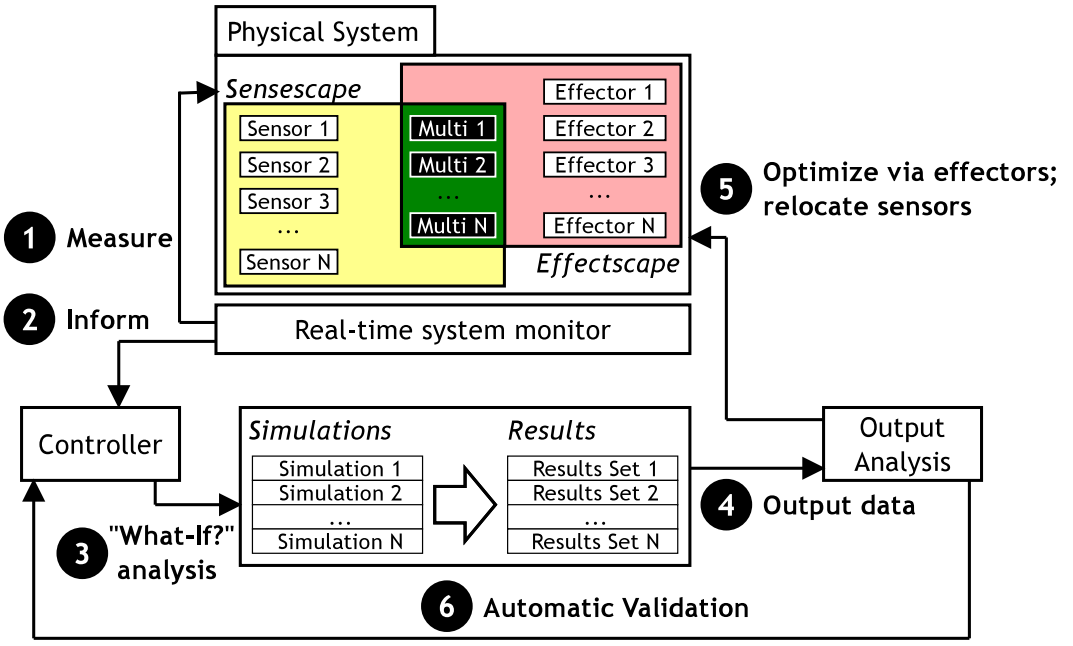

FIGURE 2.2: Illustration of the symbiotic feedback loop central to symbiotic simulation. Note the possibility of "Multi" components, which are capable of acting as both sensors and effectors.

2.1: the High-Level Architecture (HLA) and the Service-Oriented Architecture (SOA). The first of these is a general-purpose distributed simulation framework with origins in the military. The HLA is language-agnostic, allowing programs written in any language to connect over a network through a shared *run-time infrastructure*. The SOA, on the other hand, was designed primarily for inter-operability, and originated in the field of information technology. An SOA attempts to abstract business services in a way that allows trading or distributing them online. It goes without saying that some of these services may be simulation components, hence the use of SOAs for inter-operability in simulation. A key observation for both the HLA and SOA is that each framework was designed to meet the needs of its users as best as possible given the relevant historical context. The HLA, for example, was specifically designed to replace an older technique called Distributed Interactive Simulation (DIS). Likewise, the SOA is generally considered to have evolved into the field of *cloud computing*. Both of the older techniques (DIS and SOA) are still widely used, as they address different needs than their progeny.

2.1.2 The Field of Crisis Management

A full breakdown of the field of crisis management will be presented in Sections 3.1.1 and 3.1.2. Here, we will provide a brief summary to aid in understanding Figure 2.1.

Crisis management is a field that encompasses all techniques to mitigate, prevent, and respond to crises. A *crisis* is a disruptive, unpredictable event that can lead to loss of life or resources if badly managed. Crises may include disasters, epidemics, "man-made" mistakes like oil spills, and active instigations of violence such as riots. Managing these disastrous events requires a full-spectrum response, including preventative measures long before the actual event, immediate (stop-gap) mitigation techniques at the time of crisis, and a sustained post-crisis response.

The general public is usually unaware of the full breadth of tactics that crisis management teams must deploy, assuming instead that fire escape routes and flu vaccinations constitute the bulk of crisis management. More conscientious citizens may take part in community training exercises, learning how to report and deal with tropical storms and flooding. And those affected by a crisis will no doubt see clearly visible response teams soon after the initial event. Contrary to its superficial aspects, crisis management is, fundamentally, a constant war of information. Fire escape routes must be tailored to maximize egress time while minimizing bottlenecks. Flu vaccines require precision deployment strategies, as noted in Chapter 1. Community training coordinators require a full understanding of the nature and spread of potential future crises, lest they teach the wrong response and inadvertently *increase* the risk their trainees will encounter. Finally, disaster response teams need to know which regional hospital to dispatch ambulances to —a particularly difficult task, as hospitals tend to operate near peak capacity even under non-crisis conditions.

Fortunately, crisis managers have a variety of tools available to help them cope with the complexity of a given crisis. As most of the key tasks of crisis management are information-centric, it should come as no surprise that these tools tend to focus on information as an end goal. Pedestrian dynamics, for example, offers a well-researched set of movement patterns for pedestrians under different movement conditions and density levels. Community training exercises, as mentioned earlier, provide necessary information to responsible members of the community in advance of future crises. In addition, expert consultation is often utilized when designing mitigating infrastructure projects (e.g., "How high should we build the seawall?"). Many of the tools listed in Block I in Figure 2.1 would also benefit crisis management, which leads to the natural question of what simulation for crisis management would look like.

2.1.3 Bridging the Trans-Disciplinary Gap

The potential for collaboration between simulation and crisis management is vast. Figure 2.1 depicts three major examples of this crossover in Block III, decomposed into pieces

which "fit" together to form the overall solution.

The first crossover area is concerned with virtual pre-enactment of crises. Community training exercises from Block II are useful in preparing the general public for crises, but the quality of the training depends heavily on the quality of the *arbiter* mediating the exercise. It is common for participants to under-estimate the time required to perform key tasks: an ambulance driver, for example, may estimate his arrival time based on non-crisis traffic conditions, failing to take into account the increase in congestion due to panic [22]. The arbiter is responsible for affirming each time estimate and decision made, but this is often beyond the capability of one human to accomplish. Thus, one might consider importing techniques from human-in-the-loop simulation and online gaming (Block I), thereby allowing the simulation engine to act as an arbiter. This also allows training exercises to increase in size, since communication between multiple arbiters is straightforward if each arbiter is actually a simulation. In order to combine these two technologies (I and II) to arrive at our solution (III), a new component is required —marked with a "?" in Figure 2.1. This component provides the means to substitute virtual agents for human agents, and it has two facets. At one extreme, humans must be indistinguishable from software agents *to the simulation engine*, since agent generation will be necessary to run estimation models. At the other extreme, software agents must be indistinguishable from humans *to the participants*, since interaction patterns must remain the same despite who is controlling each agent. Approaching each extreme will require an increasing amount of effort, so one would expect a workable solution to lie somewhere in the middle.

The second crossover area aims to apply the power of existing models and simulations to past crises, in an attempt to identify the exacerbating factors in each scenario. In this case, one might consider creating an agent-based system that deals with traffic simulation systems. This work might be combined with pedestrian dynamics studies with the goal of robustly analyzing the past. In this case, the missing component is a means for validating agent behavior in a specific historical context. Agent-based simulations suffer from difficult validation cycles, and the possibility of *emergent behavior*, covered in Chapter 3, may require novel combinations of agent types to be re-validated. Unfortunately, the traditional method of model-building —namely, observing humans in an existing scenario and extracting the model through repeated measurement— is impractical for two reasons. First, the past event may be impossible to recreate, as the physical location it takes place at may no longer exist. Second, introducing human agents into such a scenario may be dangerous. Some form of proxy is required.

The third crossover area aims to create a comprehensive system which is specifically tuned for crisis management simulation. The promising, relatively new field of *symbiotic*

simulation is the key contribution from Block I. Likewise, all levels of crisis management, from *foresight* all the way through *hindsight* will be incorporated from Block II. Combining these elements to create a generalized solution will require a great deal of effort, mostly in the form of an overarching framework. Such a framework will necessarily maximize re-use; it would be unfortunate to create an integrated system for foresight studies that is incapable of handling hindsight. Realizations of this framework will require more up-front effort, but will incorporate factors for constant renewal, thus prolonging their lifespans and leading to the term *perennial* being used to describe them. Henceforth, we shall speak about the *perennial framework* as a means of enabling simulation for crisis management.

2.2 A Simulation Framework for Crisis Management

In Section 2.1, we outlined the intersection between crisis management and simulation. In this section, we will explore the fundamental nature of the problem before us, and the shape of a solution.

2.2.1 The Shape of a Solution

Given the broad nature of both simulation and crisis management, it is especially important that we clearly describe the shape of our proposed solution. Section 2.1.3 listed three key examples of inter-operability: a means of seamless human/virtual agent interaction, a validation technique for novel, crisis-relevant domains, and a "perennial" framework to link the fields. We had begun introducing several additional goals in Chapter 1, such as exploring the potential of virtual worlds. Combining these together, we arrive at the shape of our solution to the problem of integrating crisis management and symbiotic simulation. We will know we have arrived at an adequate solution when:

- We have tested the use of symbiotic simulation in a crisis-relevant scenario, preferably through the use of an online virtual environment with feedback to human users.
- We have developed a framework which is capable of formulating and encapsulating the above, and have assessed it with respect to existing alternatives.
- We have outlined a variety of useful configurations of the framework, and have demonstrated the steps necessary to utilize the most interesting of these cases.
- We have stressed the framework to determine the real-world overhead of perennial simulation.

2.2.2 Specific contributions

The major contributions of this thesis are, in order of importance:

- A **perennial simulation framework**, designed with crisis management in mind, which enables component re-use through foresight, hindsight, and decision-support studies.
- Means for validating agent behavior in specific, difficult situations required by crisis management, thus expanding the breadth of scenarios that can be modeled via agent-based simulation.
- Techniques for mixing virtual and human agents in virtual worlds and simulation studies, allowing simulations to pad a world with agents of either kind when necessary.

In addition, the following minor contributions will be provided, in no specific order:

- A reference implementation of the perennial simulation framework, with some means of inter-operability for legacy systems (i.e., those systems designed without a perennial element in mind).
- An analysis of several possible interactions between real and virtual worlds, as enabled by the major contributions.
- A strategy for quantifying value in a way that is useful to simulation administrators and enables the analysis of tradeoffs between various action plans, while also taking into consideration the need for a fast solution given limited computing time.
- Some insight into the efficacy of symbiotic simulation for crisis management, especially with regards to the question of whether or not humans respond positively to symbiotic feedback.

2.2.3 The Generic Quality of The Perennial Simulation Framework

One would be remiss in assuming that the solution, once fully realized, is restricted solely to the inter-operability between crisis management and simulation. Revisiting Figure 2.1, one might note that only three sides of the cube are visible. Components introduced for III will have other uses, enabling solutions for entirely different domains. Leaving aside the cube metaphor for a moment, consider the case of ambulance dispatch discussed in Section 2.1.2. Such a system is intended to minimize capacity crunches

during a crisis, but a similar setup might be used by, e.g., an automotive repair provider to dispatch incoming requests to one of several similarly equipped repair shops. In addition, the virtual worlds we use for gathering pedestrian data (Section 5.1) could easily be reconfigured to experiment with virtual-presence education, similar to the joint lecture experiment carried out in 2011 between the University of Western Australia and the University of Kentucky [23].

The key observation here is that, although our system is designed with crisis management as its primary application domain, its components are generic enough that several other domains can benefit from it with minor modifications. Thus, we will refrain from cluttering the framework's description in Section 4.2 with crisis management jargon, and we will keep the reference implementation in Section 4.3 loosely coupled with respect to crisis-specific components. When necessary, we will narrow the scope of discussion to details specific to crisis management, but all major and minor contributions should be considered generic improvements first, and crisis-specific connectivity second.

Chapter 3

Related Work

3.1 Work in Crisis Management

Crisis management is an important focal point of this research; thus, a topical review of existing research in the field is pertinent. However, the techniques encompassed by this domain depend heavily on the fundamental definition of what constitutes a crisis. Sections 3.1.1 and 3.1.2 will detail the terms "crisis" and "crisis management", and Section 3.1.3 will present related work in the field.

3.1.1 Definition and Taxonomy of a Crisis

Most work on crisis management neglects to specify what is actually meant by the term "crisis". Based on their subject matter, it is possible to arrive at the authors' assumed definitions, of which there are three. First and most prevalent is the idea of a crisis as a natural or man-made **disaster** [24] [25] [26] [9] [27]. Disasters may occur naturally, as with hurricanes and earthquakes [25], or they may be a result of human activity like traffic collisions [27]. Continuing the analogy, a disaster may occur on a very large scale, or it may only affect a small area. As noted in [27], even small disasters may escalate to affect a wide area if not properly dealt with. This reflects the *disruptive* nature of a crisis. Typically, information sharing is vital to stymie the compounding escalation of these events [28]. The second general category of crises includes agents acting against the goodwill of the general public [26] [29]. Such an **instigated** crisis may be a terrorist attack or an insider threat; the unknown nature of the agents' intentions requires information hiding and meta-level reasoning [29]. Finally, there is the notion of a **business** or public-relations disaster [30] [31] [32]. Here, the risk is not to human

lives or infrastructure, but to the well-being of a company. Timely and precise decision-making is key to resolve a business crisis [33], which might include strikes, consumer privacy violations, and product recalls.

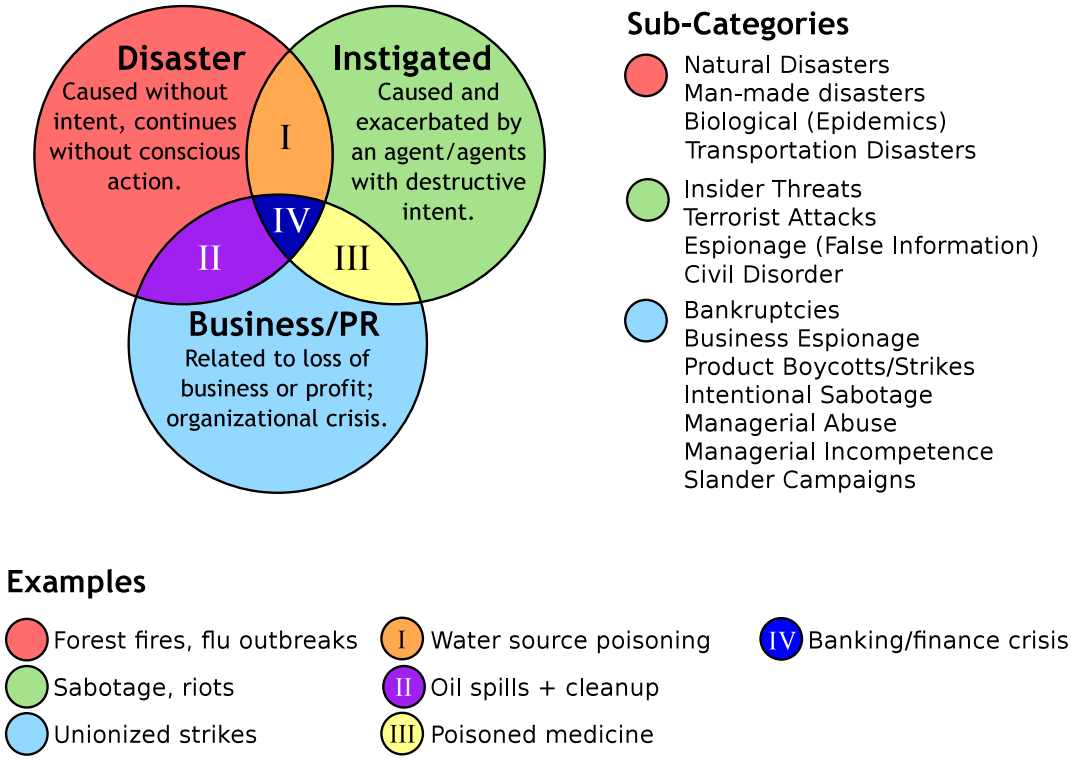

FIGURE 3.1: The various types of crises, arranged in a taxonomy with examples.

Figure 3.1 represents this empirical taxonomy, depicting the three primary categories, numerous sub-categories in each, and a few representative examples. Several observations are in order. First, disasters and instigated crises often include a high risk for loss of life, while business crises tend to only affect a single company and are generally non-lethal in and of themselves. Second, the main difference between disasters and instigated crises is the fact that the instigator *consciously continues to act* to prolong the crisis he may have also caused. This makes the event much more unpredictable, and often introduces the need for adversary modeling. Third, we observe that many crises straddle two or even all three categories. A case of poisoned medicine might combine aspects of an instigated crisis with that of a business crisis [31]. Bio-terrorism straddles disasters and instigated crises. Finally, *all* crises have an element of public relations (PR) to them. Nonetheless, Figure 3.1 provides a useful conceptual isolation, in that it separates the "difficult" crises (insider threats) from the straightforward ones (natural or man-made disasters), and completely isolates the business elements of a crisis, which never incur loss-of-life. Some crises, such as peacekeeping and humanitarian relief [26] [34], may not benefit from this classification, and will require *ad hoc* categorization based on the

unique aspects of each incident. For the remainder of this paper, whenever we use the term "crisis", we will use it to mean either a disaster or an instigated crisis. Business crises are far outside the scope of our research and will not receive further discussion.

In addition to this *prima facie* categorization of the nature of a crisis, a handful of authors have taken the time to provide a more rigorous description; the simplest of these describes a crisis as "[an event] with potentially disastrous consequences"[9], or "a situation that has reached an extremely difficult or dangerous point" [35]. Expanding on this, [36] explains that a crisis "occurs as a surprise... threatens one or more valued goals... and leaves little time for response". Finally, [24] states that crises are "situations with a high degree of threat to important values, and a high degree of time pressure". Note that, as mentioned in [27], minor disasters can build up to create the "extremely... dangerous point" from [35]. Resource management is crucial to mitigating crises; [27], for example, notes that "every crisis requires allocation of certain resources in order to rectify the situation", and others agree [9]. Finally, most crises are complicated by the fact that performance indicators are not weighted equally within the same domain, country, or expert committee [28]. In other words, choosing the "best solution" is not often obvious or even possible.

Combining the unique aspects of each definition with the literature-based taxonomy of crises yields an appropriate definition which we will use for the remainder of this paper:

Crisis

A crisis is a disruptive event that cannot be predicted. If mismanaged or otherwise left unchecked, a crisis will have a cascading effect, leading to a loss of life or resources. Crises may either occur naturally or be instigated and exacerbated by an iniquitous entity. Crises require swift action to mitigate their destructive potential.

3.1.2 Explanation of Crisis Management

The works referenced in the previous section —in addition to contributing towards a definition of the term "crisis"— primarily dealt with ways to mitigate the damage caused by crises. The formal study of these factors and how to best deal with their myriad complexities and inter-connected relationships constitutes the field of *crisis management*, which can be understood based on its *events* and *actions*.

The *events* a crisis manager must deal with are manifold, spanning the time before, during, and after a crisis. They are also referred to as "problems", "tasks", or "goals". Tufekci and Wallace explain how different events in the timeline of an emerging crisis

require radically different strategies to mitigate. Figure 3.2 is from their research on Emergency Management [9], modified for Gupta and Ranganathan's traffic collision crisis scenario [27].

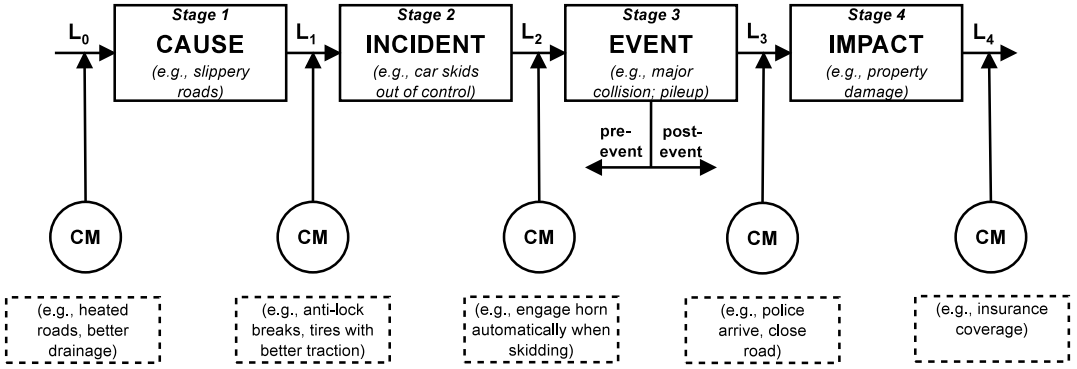

FIGURE 3.2: Tasks of Emergency Management arranged in time with respect to the crisis event. Potential crisis management intervention periods are marked as CM; example interventions from a traffic crisis are given in dashed boxes.

A **crisis event** is divided into four stages including the event itself: **cause**, **incident**, **event**, and **impact**. For the sake of discussion, assume the **event** is Gupta and Ranganathan's example of a multi-car traffic collision crisis [27]. In this case, the **incident** may be a single car skidding out of control[1]. At this point the event is imminent, but some last-minute intervention (such as automatically sounding the car's horn when it detects skidding) may still be both possible and helpful. Directly preceding the incident is the **cause**, which in our case is a slippery roadway. Finally, after the event is the **impact**: property damage. Mitigation can occur before or after each stage. If an intervention strategy occurs, for example, between **cause** and **incident**, then we can say that its purpose is to either stop the transition from cause to incident, to limit the amount of the cause that remains effective, or to limit the degree to which the incident is magnified by the cause. The nature of any given crisis will radically affect the number and cost of strategies available at each stage. By extension, some strategies may take an exceedingly long time to implement, or may become effective only after a certain penetration level has been reached. Installing a special braking system is an example with both of these properties. Several other example strategies are provided by Figure 3.2; most need no further explanation. As noted by the authors, the best strategies will generally involve a combined effort over all five mitigation points.

Broadly speaking, all of Tufekci and Wallace's categories use *foresight* to prepare intervention strategies. The notion of *hindsight* only exists implicitly in Figure 3.2 as an assumed source of domain knowledge from which one draws intervention strategies.

[1] In a densely populated area, even a minor accident can build up to disastrous consequences. Los Angeles, origin of the ATSAC system, lists an "extreme incident" or *Sig Alert* as "any unplanned event that causes the closing of one lane of traffic for 30 minutes or more".

Other researchers have relied on hindsight as a first-class relative to foresight, including [34], which lists it as a key task. Additionally, the Federal Emergency Management Agency (FEMA) includes hindsight in the "Information Update" stage of its lifecycle diagram of crisis management [37]. Figure 3.3 depicts the FEMA diagram, updated with crisis management tasks from the literature reviewed earlier [9] [38] [26]. Several trends are evident. First, each of the five phases relies on some form of *information superiority* —this is unsurprising given the nature of crisis management. Second, the early phases feature tight time constraints, while the later phases can spare time but demand a high level of accuracy. This mirrors the time-driven nature of *decision support* studies, introduced below, versus the knowledge-driven nature of hindsight and foresight studies. Finally, the FEMA diagram makes explicit the cyclical nature of crisis occurrences. The notion of expecting and actively preparing for a future crisis may seem sensible or even trivial, but it is a point that is often overlooked, leading to confusion and misrepresentation. In particular, this clarifies the role of "Preparedness" studies in Tufekci and Wallace's diagram (Figure 3.2), which would otherwise seem to occur both after L_4 and before L_0.

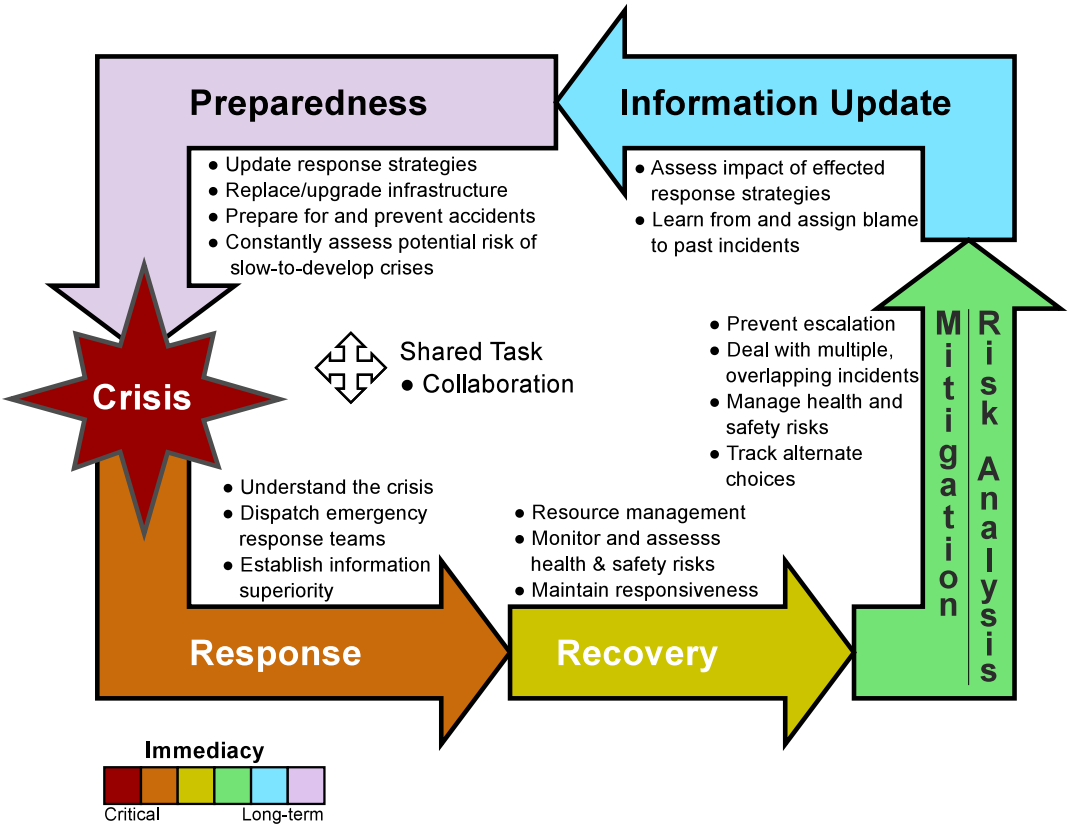

FIGURE 3.3: The FEMA crisis management lifecycle diagram, annotated with tasks from the literature. Note the emphasis on crisis management's cyclical nature.

Hindsight and *foresight* are both extremely useful for large-scale crises. The former

enables administrators to avoid repeating the mistakes of past crises, while the latter encourages experimenting with novel solutions to potential future crises. The final category of crisis management is that of *decision support*. Essentially a component of the "Response" step, it is much broader and more critical than implied in [9]. Decision support encompasses all data gathering, "What-If?" scenario analysis, and tertiary activities that may be required by crisis management administrators during the time directly after a crisis. This category of tasks is often the most difficult element of crisis management, as it is characterized by both an extreme time crunch and a paucity of available information. The tendency of some crises (namely disasters) to destroy portions of a city's technological infrastructure only serves to aggravate this. As such, it is common for decision support tasks to be partitioned such that they can be dealt with as quickly as possible during the actual crisis. For example, several city evacuation routes may be carefully researched in advance of a crisis, with the actual plan chosen depending on sensor readings at the moment of crisis itself. Existing research confirms that —from a decision support point of view— providing a good, timely solution is better than applying the best possible solution after too long of a delay. Section 3.1.3 covers this in more detail.

3.1.3 Topical Review of Crisis Management Research

A great deal of research exists to deal with the various tasks of crisis management.

During the time of crisis itself, as well as before and after, information and communication are the most commonly-cited goals. [25] claims that information is "the common denominator" in all crises, lambasting the damage caused by "conflicting information". That said, the mechanics of most environmental disasters have been well-studied. Fire spread models are thoroughly understood in most interior and exterior contexts. Papers by Belkhouche *et al* [39] and Douglas *et al* [40] are two excellent examples; the former for its simplicity and the latter for its use of DDDAS (a novel dynamic data-gathering technique, described later) to capture the specifics of a wilderness fire. Floods and earthquakes have been heavily researched, too. Besides studies of the physical forces which cause disasters, many researchers are now looking at the capabilities of emergency response teams *after* the initial catastrophic event. Fiedrich, for example, focuses on research allocation after earthquakes [41]. Jain [42] and Shendarkar *et al* [43] both study egress after a bomb attack, rather than modelling the actual explosion. More general studies like that of Low *et al* [44] use sophisticated technologies such as the High-Level Architecture [45] to model the relatively simple —but extremely useful— patterns of crowds of people. Wilcox applies similar logic in an attempt to understand the issue of neighborhood crime. His work is notable for its use of modern agent-based simulation

techniques; in particular, he creates an agent-based lattice architecture to study the complex effects of "reciprocal social influences" on criminal activity [46].

Recently, studies of infectious diseases have become more common. Such studies are usually concerned with the spread of disease within a certain geographical space (such as a university campus [47] or an entire urban region [48]), and the extent of the effect of policy on the spreading of the disease. Some are concerned with life-long diseases such as AIDS [49]. Most studies take preparedness as a theme, speculating, for example, on the effect that various historical outbreaks would have in a modern urban environment. Given the number and importance of these types of studies, we find it disappointing that so few of them rely on DDDAS. This technique, described in detail in the following section, gathers data from very large, online, dynamic data sets, and is often used to study public policy from a simulation-based perspective. We would expect future infectious disease studies to start integrating DDDAS out of necessity, and are surprised this has not already happened.

Resource management is as equally well-studied as the mechanics of physical disaster. In Gupta and Ranganathan's work (which also appealed to the FEMA diagrams), resource management is identified as a key task during a crisis. Nash equilibriums from game theory are employed in an attempt to optimize various tricky crisis-related resource problems [27]. Fiedrich's work, mentioned earlier, approaches earthquakes with the observation that rescue attempts are put under extreme pressure by limited resources and high demand, and that resource allocation after a crisis can have the greatest impact on mitigating damages [41]. Likewise, the World Health Organization (WHO) cites "prioritization of limited resources" as a key goal in dealing with infectious disease outbreaks [4].

Information management takes center stage in a lot of work, often from differing perspectives and to a variety of end goals. For example, Tufekci and Wallace put information at the center of their work, arguing for a holistic approach spanning pre- and post-objectives of crisis management [9]. They argue that the interconnected nature of crisis incidents —even small ones— requires a global analysis of the combined situation. The work of Cross *et al.* and Sakairi *et al.* is similarly concerned with global information management, but from the perspective of collaboration during a crisis. Sakairi *et al.* use AJAX and other web technologies to speed up visualization of GIS data [50], while Cross *et al.* attempt to enhance communication from a military perspective [51]. [52] is similar, using indexing techniques to improve access to real-time information from a Command, Communication, and Control (C3) point-of-view. Finally, [38] uses a federated system to handle the meta-level task of "collaboration management" —that is, organizing collaborations themselves. Collaboration is defined as "occur[ring] whenever

humans and/or computer applications work together to accomplish a common goal", and involves rules on "when, how, and by whom" each collaborative activity is performed [38].

Also studied is the notion of response to a crisis. Smith and Hayne attempt to provide fast decision-making capabilities, noting that "an underlying assumption of crisis management is that a series of timely but non-optimal decisions will generally yield a better outcome than optimal decisions made too late." [24]. They use "elimination by aspects" to achieve speedup over a utility function when performing multi-objective analysis in real-time. Finally, Jacobs *et al.* utilize a multi-user virtual environment called MUDSPOT to analyze crisis planning by real users in a virtual representation of a crisis [34]. This appeal to a virtual world is not uncommon, as many have noted its potential utility as an emerging technology [9] [53].

In relation to our work, the existing literature on crisis management is considered sufficient, and our primary focus is on bridging the trans-disciplinary gap between simulation and crisis management. The latter is slowly moving towards a reliance on the former. Studies performed without an appeal to simulation are still common, but most recent studies recognize its potential and wholeheartedly embrace simulation as part of an integrated solution [9] [54]. That said, existing work on crisis management is deficient in regards to simulation in general and agent-based simulation in particular. Simulation, as stated in Chapter 2, is capable of shortening the time gap between the occurrence of a crisis event and the moment when information superiority over that crisis is achieved. Agents provide a natural encapsulation model for humans, and allow reasoning about domains which have typically proven challenging for older, top-down simulation methodologies. Our work will attempt to bring these two technologies to crisis management. Finally, we observe the practical success of DDDAS in its stated goal of enabling simulation with exhaustively large data sets. We choose to leverage a similar technology (symbiotic simulation) to achieve some of the benefits of DDDAS without the additional complexity of reconfiguring sensors. Nevertheless, our sensors as defined in Chapter 4 are designed to be DDDAS-capable: although our reference implementation uses a static dependency tree to refer to sensors, an alternative implementation may choose to store sensors in a more flexible data structure and swap active sensors at runtime.

3.2 Work in Health Care Simulation

Related to crisis management is the field of health care simulation. Many hospitals have undertaken logistical studies to increase throughput and decrease waiting time under normal operating conditions. Such studies would, at the time of a crisis, be useful for

measuring the strain on each hospital and allowing ambulances to divert to less congested drop-off points.This assumes, of course, that these simulations are valid for such an application. In fact, the pattern of measuring the system under "normal" conditions and then optimizing it after a certain threshold has been reached is reminiscent of symbiotic simulation's pursuit of equilibrium.

Several interesting studies stand out. One of these manages to reduce the waiting time for low-risk patients through the use of an improved triage protocol known as "Provider-Directed Queuing" [54]. The authors note that, like many other hospital studies, a considerable change in arrival rate over time makes reasoning about the system difficult. Nonetheless, they achieve promising enough results to justify testing a real-world implementation of the modeled system. Improvements in waiting time of 44% to 76% are observed in the pilot study used to validate this simulation. This excellent validation technique allows them to overcome the unfortunately high variance in their output data.

Other hospital simulations follow a similar pattern. A model is formed to study a variable such as queue length [55] or patient no-show rate [56]. The most critical areas for optimization are considered first, with a vigorous output analysis and validation step to support the derived conclusions. Occasionally, a surprising correlation is discovered —such as [56]'s observation that a reduction in the number of appointment *types* at a hospital led to a reduction in the number of no-shows. Overall, hospital simulation studies tend to be logistics studies, focusing on patient flow or hospital resources and capacity.

Healthcare is another field which has been moving steadily towards a reliance on simulation. One noticeable exception is that of physicians themselves, as noted by Fackler [57]. Although physicians identify patterns and perform just-in-time mental modeling [58], there is a curious resistance to extracting the modeling of mundane or repetitive tasks into computer simulations. Some fields of medicine (such as anesthesia) are moving towards simulation technology, with doctors fulfilling a "decision support" style role situated between the simulation and the patients. This notion of experts fulfilling a central role making critical decisions based on simulated data will be encouraged and enhanced in our framework in Chapter 4.

Regarding our work, we see the value in [54]'s use of real-world experimentation as a means of overcoming uncertainty. We make use of this technique in our first study (Section 5.1) through the application of a virtual-world experiment with live users. Our second experimental study, while not directly related to health care simulation, draws inspiration from past work in this field by attempting to analyze the trade-offs of various dispatch strategies in a time and space-critical scenario (Section 5.2). Finally, regarding

past research holistically, we speculate that simulation's lukewarm reception by physicians is indicative of a deficient regard for *credibility*. According to Law, "a simulation model and its results have credibility if the decision-maker and other key project personnel accept them as *correct*." [59]. While validity is of the utmost importance in terms of guaranteeing a model's correctness, credibility is essential for ensuring that experts will actually make use of the system and trust its results. We attempt to address this through the application of virtual world re-enactments of crisis scenarios, and by providing categorical assessments of symbiotic feedback (through, e.g., Pareto analysis), instead of merely providing the "best" solution.

3.3 Work in Symbiotic Simulation

Symbiotic simulation was defined in Chapter 2, and most early work in the field followed the "Grand Challenges" specification to the letter. For example, consider Low *et al*'s [60] application of symbiotic simulation to the problem of queue overflow in semiconductor assembly plants. An emulated system represents the production factory, and the *Witness* modeling software performs "What-If?" analysis. Agent-based technology is used to create the symbiotic link. When a certain queue length is reached, the optimizing agent attempts to determine the upper and lower queue sizes for several Wire-bond and Test machines. Choosing the appropriate queue sizes will limit outsourcing while preventing overflow on-site. Unfortunately, the entire system relied on an *emulation* of the real world; there was no "physical system". Cross-validating with a pilot study performed at a real-world factory would have helped increase confidence in the results of the system. The term "emulation" is also somewhat misleading, as it implies perfect imitation of the physical system. Despite these shortcomings, Low *et al*'s approach stands as an excellent example of a classic symbiotic simulation system. Moreover, it is interesting to note the authors' claim that symbiotic simulation is necessary to "respond to abrupt changes in the physical system" —a claim that echoes our own decision to leverage symbiotic simulation for crisis management. In fact, an "abrupt change" in Low *et al*'s work could be considered synonymous with the "extremely difficult or dangerous point" identified in our previous overview of crisis management.

Later work by Aydt *et al* formalizes the distinction between *reactive* and *preventative* symbiotic simulation [61]. This distinction is visualized in Figure 3.4, which illustrates a thresh-hold condition in each case. In reactive analysis, the threshold is crossed at time t_c, at which point "What-If?" analysis is triggered and preventative measures are taken. By time t_a, these measures have taken effect, and the critical condition is no longer affecting the system. Some time later, the system resumes normal behavior. In

preventative analysis, "What-If?" analysis is triggered at points t_{a3}, t_{a2}, and t_{a1}, all of which predict that the system will soon cross into the critical zone. Depending on the cost of intervention, action can be taken at either of the three action points, and the critical point t_c never occurs.

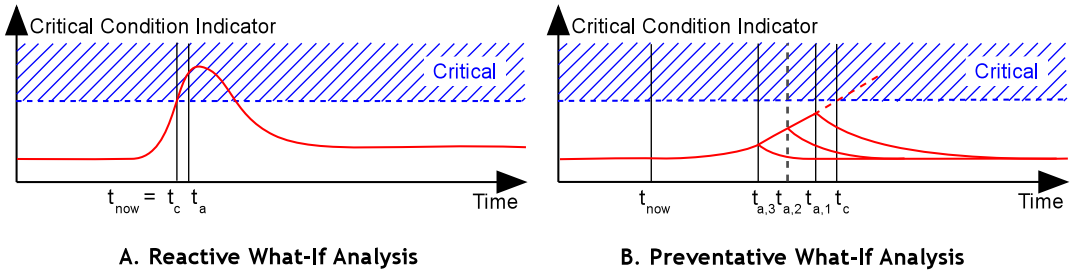

FIGURE 3.4: Reactive versus Preventative Symbiotic Simulation

Aydt *et al.* recognizes that Type I and Type II errors can occur in preventative symbiotic simulation. Type I errors occur if the system does not detect that the threshold has been crossed, and allows the system to continue unimpeded past the crisis point. These can be eliminated entirely by using reactive analysis as a fall back. Type II errors occur if the system predicts that the threshold will be crossed when, in reality, such a thing would not have actually happened. These errors increase in likelihood as the probability of a critical condition decreases, and are usually very costly as they apply unnecessary interventions. The authors introduce a G-value estimation to determine when preventative analysis is appropriate [61].

A related field to symbiotic simulation is that of Dynamic, Data-Driven Application Systems (DDDAS). Darema provides a clear definition of these: "DDDAS entails the ability to incorporate additional data into an executing application and, in reverse, the ability of applications to dynamically steer the measurement process." [21]. Such "additional data" are assumed to be so vast —a full listing of all real estate transactions in a state for the past five years, or real-time atmospheric pressure readings over an entire region— that the steering step is always necessary. Data sets need not be real-time, though: the AIMSS project is an example DDDAS study that mines previously published housing prices and tenant relocations from the world wide web in an attempt to determine the "happiness" and "over-crowdedness" of a neighborhood [62]. Finally, DDDAS is sometimes utilized as an extension to traditional simulation models, such as maintenance scheduling for semiconductor supply chains [63].

Our work borrows heavily from symbiotic simulation, deeming it crucial to effective crisis management. As symbiotic simulation is a relatively new field, we find it lacking in several regards. Most importantly, symbiotic simulation is too rigid in its application of a maximization function for measuring performance. More complex and flexible metrics

are necessary when dealing with human agents, which we attempt to address in Section 4.3.3.1 and throughout this thesis. Similarly, the distinction between reactive and preventative simulation does not adequately capture the fact that Type I errors are so costly to crisis management that they must never be triggered. This is because the true cost of triggering crisis response for a non-crisis is not money or time lost, but loss of public confidence in the crisis response mechanism itself —something which will lead to uncertainty and therefore massive loss during the next actual crisis. We attempt to mitigate this by assigning a human the role of **Implementer**, and relegating the perennial simulation to the role of decision support rather than absolute authority. This is similar to simulation's role as discussed in the previous section on health care simulation.

3.4 Work in Agent-Based Simulation

As previously stated, developing a comprehensive perennial simulation framework will necessitate borrowing techniques from agent-based simulation. In addition to serving as the primary modeling abstraction within our framework, agent-based modeling and simulation may also be useful to validate models —especially when combined with virtual worlds.

A simple yet insightful study performed early in the lifetime of agent-based simulation attempted to determine if humans' tendency to over-estimate the probability of "good" events and under-estimate the probability of "bad" events (known as "optimistic bias") was beneficial or detrimental to their betting performance under conditions of uncertainty [64]. A sample world was created which forced agents to make a series of bets, each of which had a probability for success, a reward for success, and a penalty for failure. Knowledge of each of these three data points could be obscured by uncertainty. Several agent were constructed that mimicked the human tendencies of *optimistic bias* and *emotional bias* (ignoring the probability of success when the reward greatly exceeded the penalty for failure, and vice-versa). Classic Von Neumann rational agents, which would *only* act in a way that maximized their expected utility, were included as a control group. After several runs under varying levels of uncertainty, it was found that agents with an optimistic or emotional bias tended to outperform rational agents when some uncertainty was present. The authors speculate that this beneficial departure from normative rationality is reflective of conditions encountered in the real world, as humans are generally capable of discerning the costs of success and failure, but are poor judges of the probability of success. This study took place early into the modern era of agent-based simulation, and serves as an illustrative example of the technique in general.

An excellent overview of agent-based simulation is given by Lee and Son [65]. This crowd simulation study first categorizes agent-based approaches into three categories:

- The *economical approach* assumes that decision makers are rational, and reasons out movement patterns based on this logic. As seen in [64], humans rarely behave with strict rationality.
- The *psychological approach* "considers human cognitive natures explicitly, [but] mainly focuses on... behavior under simplified and controlled laboratory experiments" [65].
- The *synthetic engineering approach* attempts to reconstruct a model from observed human behavior. Whereas the first two approaches are forms of forward reasoning, this approach operates on the basis of reverse engineering.

Lee and Son choose the third approach, settling on Belief-Desire-Intention (BDI) models as the specific enabling technology. BDI models are goal-based, and are thus much easier to understand than complex neurological models. This clarity is very important for a simulation, as validation often requires experts in various different fields to understand model assumptions and interactions.

The BDI model used in Lee and Son's study is reverse-engineered from data gathered by human participants in the Cave Automatic Virtual Environment (CAVE), a fully immersive 3D virtual environment created using screen projection [66]. The study makes reasonable assumptions, and its approach to modeling is both responsible and sensible. The authors mention in passing that the CAVE could be used to validate their model, and in stopping there they —quite unfortunately— pass up a golden opportunity to lend credibility to their model through rigorous validation. This highlights a problem with agent-based simulation studies in general: validation is often left as an afterthought [55] [56].

Many crowd dynamics simulations use agents; these will be listed in Section 3.6. One particularly interesting case stands out: Chaturvedi *et al* took an otherwise-ordinary topic —egress from a burning building— and created a seamless interaction between the agent-based component and the fire model [67]. This allowed the fire to spread and limit the actions of the agents, while at the same time ensuring that any actions taken by agents attempting to escape (e.g., breaking an exterior window) also affected the environment of the fire. Such mutual reflexive feedback is reminiscent of symbiotic simulation, although this particular study is missing a few key technical aspects to be considered fully symbiotic.

A great example of the potential future of the field is Padgham's work, which integrates BDI reasoning capabilities into the Java simulation library RePast [68]. This integrated system is then used to study the behavior of residents and city-planners in response to bush fires. An emphasis is placed on loosely coupling systems as a means of reusing existing high-fidelity systems. Padgham's work reflects a trend of using agent-based cognitive reasoning in the social sciences, which is something that leads naturally to a possible application of symbiotic simulation.

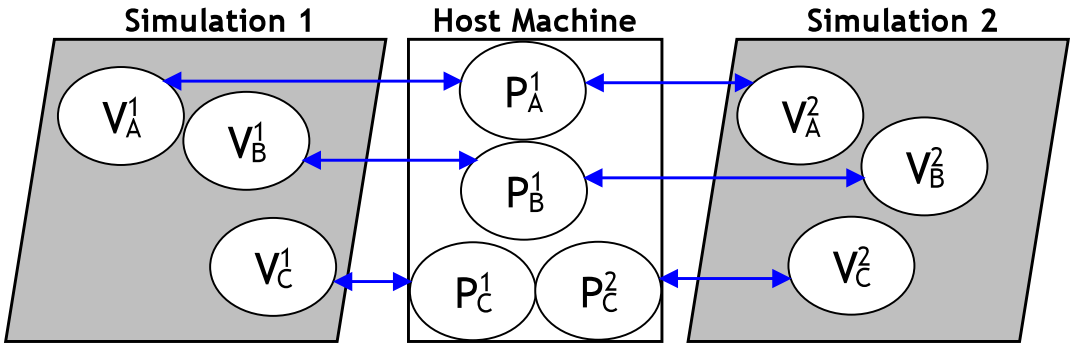

FIGURE 3.5: Virtual processes for two arbitrary simulations running on the same host machine.

Tangential to this is the work of Hybinette and Fujimoto [69] [70], which is becoming increasingly relevant given the inherent complexity and performance overhead of agent-based models combined with the need for reasonably fast results in symbiotic simulation. Using dynamic virtual logical processes, a simulation can reduce the number of physical processes required on a host machine. In all cases, this outperforms replication [70]. Figure 3.5 illustrates the concept. Two simulations are running in concert on the same host machine. Simulation 1 was created first, and then simulation 2 was "cloned" from it. At the point of cloning, all virtual processes in simulation 2 map to the same physical processes that simulation 1 was using. Eventually, virtual process C in simulation 2 (denoted as V_C^2) intends to simulate an action that conflicts with simulation 1's perceived view of the world. The physical process for C is cloned, leading to a total of 4 physical processes and 6 virtual processes. To accomplish this, a message passing paradigm is used to determine if and when a physical process should be cloned [69]. Essentially, any message that is not multi-cast to all virtual processes chained to the same physical process may induce the duplication of a physical process. This agent virtualization technique is transparent and quite suitable for mitigating the performance penalty of massive "What-If?" analysis incurred by our perennial framework.

From the perspective of achieving our goals as laid out in Chapter 2, the breadth of existing work on agent-based simulation is both essential and fully matured. Software agents are well established in the field of artificial intelligence, with frameworks such

as RePast enabling fast prototyping of new agent models and novel multi-agent combinations. Agents are an effective isolation mechanism, enabling a variety of scalability techniques and migration strategies. If speed is paramount, frameworks such as MASON offer similar functionality to RePast with excellent, scalable performance [71].

Regarding the use of agent-based simulation for new areas (such as crisis management simulation), the literature primarily identifies challenges related to validation. Most of these stem from the emergent behavior inherent in systems built using a bottom-up, heterogeneous approach [72]. Going further, [10] approaches the difficulty of empirical validation from the viewpoint of agent-based economics models, but the problems identified are applicable to the field of agent-based simulation in general. Empirical validation is usually approached as either an indirect calibration approach, or as a check against available historical case-studies. The problem with the former approach is that model development is often encouraged by the existence of large data sets. In contrast, "there are many potentially important variables and parameters for which data do not currently exist. Some may not be amenable for quantitative measurement" [10]. This is compounded by the potential lack of availability or insufficient quality of empirical data. The problem with the latter approach is that it does not apply a very strong test to the model. Individual traces are not sufficient to determine typical model behavior, in addition to the challenge of accurately capturing the historical traces in the first place. Non-economics agent-based simulation research confirms these issues. [73] notes the lack of empirical data impeding traditional validation approaches for multiagent systems, and proposes relying on face validity via a human expert immersed in a virtual instantiation of the model. Similarly, [74] notes that while agent-based models may aid in understanding complex social systems, a high level of inter-agent interaction, increased number of assumptions inherent in the model, and common lack of an experimental control dataset frustrates validation efforts. Finally, [75] describes the validation of a juvenile delinquency model using the indirect calibration approach identified by [10]. The authors rely on data from a large, multi-year study, comparing the model's performance given the first year of data to the actual data reported in the second year. It should be clear that in the case of crisis management simulation, the rare and random nature of crises hinders the ability to build the large datasets required for validation, while at the same time challenging the construction of historical case-studies. To address these issues, we draw inspiration from [54], and use high-fidelity real-world studies carried out in a virtual environment, as described in Section 4.2.6. A consequence of this approach is the need to address the mixing of humans and software agents in virtual worlds. In Sections 5.1.3.1 and 5.3, we show how to allow this type of mixing by carefully limiting agent-to-agent interactions with a focus on *role validity* at runtime, while simultaneously maintaining immersion.

3.5 Work in Human-In-The-Loop Simulation

The earliest human-in-the-loop simulations were flight simulators, which were used to reproduce the experiences encountered in an aircraft cockpit while shielding pilots from any inherent risks and reducing the overall cost of training [76]. Following that, space flight began to use simulation technology for the same purpose. Ackerman presents comprehensive insight into what these kinds of systems might take into account [77]. In hindsight (the paper was published in 1959) some of Ackermans's suggestions might seem outlandish —his casual assumption that spacecraft will be nuclear powered, for example, or the various questionably ethical means by which one might simulate pressure— but his categorical survey of necessary components is valid and worth reviewing. A spacecraft module simulator would have to replicate the sights and sounds of space flight, while accounting for weightlessness, food, and fatigue. Computer technology is relied upon only for sound and visual simulation; a great deal of the paper is concerned with mechanical means for mimicking the realities of space flight.

Over the intervening decades, the space industry has expanded its simulators to cover more than simple flight sensations. Peaden describes the massive human-in-the-loop simulations used by the National Aeronautics and Space Administration (NASA) to train teams to coordinate properly on missions [78]. Introduced partly as a response to the Challenger disaster, these real-time exercises train groups of personnel ranging in size from individual component maintainers all the way up to full launch teams. NASA's training simulations bear some similarity to a technique called MMOHILS, which is discussed in Chapter 4. Tellingly enough, Peaden reflects on "whether it might be possible and beneficial to add some low fidelity software simulated humans to certain training curriculum" —an idea which MMOHILS expands on wholeheartedly.

There exist applications of human-in-the-loop simulation in the military as well. Downes *et al* describe a system which is used for gauging the performance of information systems within the military, using real humans as well as software agents to gather data on information sharing between soldiers throughout the course of a mission [79]. Datasets gathered from humans are found to be more accurate than those gathered from software agents, exposing several flaws in the way software agents are modeled. Downes *et al*'s work is interesting, but possible future work and extensions are greatly underemphasized. For example, one might consider upgrading the existing agent models using real-time data gathered from the human participants.

Many more examples of human-in-the-loop simulation exist within the literature. Hopkinson and Sepulveda's work is the most relevant, as it is concerned with the issue of role validity within *all* such simulations. Essentially, a human acting withing a loop

satisfies *role validity* if he is acting in accordance with the specific *role* he is meant to be playing for the purpose of the exercise. If a human is not role-valid, then the simulation can either prompt him to correct his behavior, or remove him from the virtual world and abort the training procedure. Either approach conserves resources, and reduces the total time that the simulation is running in an invalid state. Hopkinson and Sepulveda use case-based reasoning to identify violations of role, creating a flexible yet dynamic system which can catch errors in real-time [80].

Our research builds directly on existing work in the field of human-in-the-loop simulation, extending it with techniques from massively-multiplayer online games to form a niche configuration of perennial simulation that we call MMOHILS (see: Section 4.2.6). Additional methods are presented which allow for easier *role validation* and *psycho-social validation* within a MMOHILS; this becomes important if participants are not trusted (i.e., if the "loop" in this case is an existing onine game world). To this end, [80]'s work on role validation is used almost verbatim. Generally speaking, we found the existing research in human-in-the-loop simulation to be of excellent quality, but at the same time unfortunately dated. MMOHILS is our attempt to bring to modernity the elements of this field that we need for our perennial framework.

3.6 Work in Crowd Dynamics

Many crisis conditions require an understanding of crowd movement patterns, and indeed one of our prototype studies is largely crowd-based. As such, work from the field of crowd dynamics will now be covered.

One of the most well-known studies in crowd dynamics was Fruin's *Pedestrian Planning and Design* [81]. After observing a staggering amount of pedestrian traffic, overall distributions of human speed versus crowd density were extracted, and a generalized flow analysis was applied. Though not motivated by simulation, Fruin's classification of rooms and walkways into "Levels of Service" was easy to translate into a digital format, and many simulation studies have worked off of or validated to his data (including our own).

Fruin's "Levels of Service" (L.O.S.) are widely used in building planning projects; they are both accurate and extremely simple to grasp. The EvacNET modeling framework, introduced in the next chapter, provides a good summary of each L.O.S. [82]. Figure 3.6 details the walkway L.O.S., which are used for long, straight rooms like walkways or hallways. Other L.O.S. exist for stairways, landings, and situations of queuing as well.

Level of Service A	Level of Service B	Level of Service C	Level of Service D	Level of Service E	Level of Service F
Area Occupancy: 35 sq.ft/person *Average Speed:* 260 ft/min *Avg. Flow Volume:* 7 PFM *Description:* Minimum maneuvering to pass; crossing and reverse movements are unrestricted; flow is appx. 25% of maximum capacity.	*Area Occupancy:* 25–35 sq.ft/person *Average Speed:* 250–260 ft/min *Avg. Flow Volume:* 7–10 PFM *Description:* Occasional interference in passing, crossing & reverse movements. Flow is appx. 35% of maximum capacity.	*Area Occupancy:* 15–25 sq.ft/person *Average Speed:* 230–250 ft/min *Avg. Flow Volume:* 10–15 PFM *Description:* Passing is restricted but possible with maneuvering. Flow is reasonably fluid and is about 40 to 65% of maximum capacity.	*Area Occupancy:* 10–15 sq.ft/person *Average Speed:* 200–230 ft/min *Avg. Flow Volume:* 15–20 PFM *Description:* Conflict-free passing is rarely possible; cross and reverse flow are restricted. Flow is appx. 65 to 80% of maximum capacity.	*Area Occupancy:* 10–15 sq.ft/person *Average Speed:* 110–200 ft/min *Avg. Flow Volume:* 20–25 PFM *Description:* Frequent shuffling; passing, cross and reverse flow always conflicted. Flows at maximum capacity; interrupted.	*Area Occupancy:* 5 sq.ft/person *Average Speed:* 0–110 ft/min *Avg. Flow Volume:* 25 PFM *Description:* Frequent shuffling; passing, cross and reverse flow always conflicted. Flows on the verge of breakdown.

FIGURE 3.6: Walkway levels of service (L.O.S.), summarized by EvacNET. PFM stands for "pedestrians per foot-width of walkway, per minute".

Fruin's contribution to pedestrian planning should not be understated. Recently, however, there have been clarifications and extensions to his original measurements. Togawa's observations [83] of movement speed versus density are generally considered to be more accurate [84], and newer studies such as Aoki [85] and the Green Guide [86] match Togawa's data, thus adding to the consensus.

Moving beyond mere mechanics, Still's work on the *Legion* simulation software [84] is an excellent showcase of the effort involved in validating an agent-based simulation platform. *Legion* is one of the few simulation toolkits available for crowd dynamics studies which uses software agents, although this may be less reflective of a trend against agent-based methods *in general* than a preference for agent-aware *libraries* such as JADE [87] or RePast [88]. Indeed, the number of agent-based crowd dynamics studies is not small [43] [44] [67].

Other studies are much more focused. Helbing's introduction of *social forces* posits that a large crowd composed of sub-groups of people is concise, flexible, and accurate for describing pedestrian movement [89]. Groups (or individuals) are assigned three vectors representing their attractiveness *towards* their goal location, *away from* other people in general, and *towards* others in their group. Although incompatible with traditional physical forces (the social forces in a model do not balance), the equations that govern their application are the same as those found in physics. One recent study augments social forces with the means to approximate age-based parameters [90]. Agents are considered too intensive to process individually, so small groups of humans featuring "homogeneity of goals" are handled by the model. The assumption is made that only the resultant (group) behavior must be valid; the internal workings of the model need not represent the conscious decision-making process of an actual pedestrian.

Although the social forces approach to modeling crowds is attractive, we discount its use for crisis management because it relies on two controversial assumptions:

- *The behavior of a person exhibits certain regularities* [90]. We consider this assumption unrealistic in the face of a crisis.
- *A person will make optimum movement choices* [90]. Research has already shown [91] that this assumption is tenuous, and may endanger the validity of a model.

Crowd dynamics studies are useful to crisis management; however, a great deal of attention must be paid to the assumptions common to each study. The nature of a crisis —sporadic, disruptive, and catastrophic— couldn't be further removed from the mundane nature of a crowd of pedestrians. Helbing's original paper on social forces admits that his model probably cannot represent complex human behavior [89].

Regarding our research, existing work from crowd dynamics is used extensively in our experiments. Fruin and Togawa's measurements are employed to model movement within several building scenarios, presented in Chapter 5. Note that, despite its prevalence in the academic community, we consider Helbing's work to be unacceptable for a variety of reasons. Besides those mentioned previously, we also consider the social forces model to be inadequate as it only deals with the *operational* level of movement, with no regard for the *strategic* or *tactical* level (e.g., route choice). Regardless, once a suitable model was selected, our remaining contributions to the field of crowd dynamics were minimal. In particular, we attempted to show that symbiotic feedback could be used to optimize a crowd under egress conditions, while reducing counterproductive behavior such as herding.

3.7 Work In Traffic Modelling and Simulation

In addition to pedestrian elements, many crisis simulations, including one of our prototype studies, have an element of traffic simulation. Our work benefits from existing traffic models, of which the field is well-saturated. Boxill and Yu provide a thorough overview of the most-used traffic simulation packages as of the turn of the century [92]. Broadly speaking, these models operate at one of three resolutions. *Microscopic* models are the slowest and most accurate, operating at the level of each vehicle. *Macroscopic* models calculate congestion based on each link (road and intersection). They are less accurate but much faster. *Mesoscopic* models operate in between the two extremes, both in terms of accuracy and performance. Some studies operate at multiple resolutions,

such as [93], which leverages both micro- and mesoscopic models simultaneously. Recently, microscopic models have received a great deal of research attention; Chowdhury *et al.* explore them in more detail in [94]. The most feature-complete simulation packages were AIMSUM, Paramics, and DYNAMIT. In addition, MITSIMLab is a popular open-source alternative with a comprehensive feature set [95]. Other interesting models like DRACULA and NEMIS focused on smaller tasks such as driver behavioral patterns over time or evaluating the impact of route choice. Some newer research attempts to combine microscopic traffic models with agent-based simulation techniques [96]. The black-box nature of agents is an attractive mechanism for abstracting driver behavior, and provides a modular means of adding new types of agents such as pedestrians or traffic lights. Tao and Huang provides a good example, using the FIPA-compliant JADE framework [87] to design a traffic simulation which represents cars as agents and can dynamically shuffle agents onto different networked computers if the resources of one system become exhausted [97].

Our traffic study relies on the work of Krauss, which defines a microscopic, collision-free, space-continuous traffic model with a focus on driver imperfections and jam modelling [98] [99]. Drivers respond to congestion through dynamic user assignment, which is implemented as described in [100]. The SUMO simulation package provides an integrated framework for all of these approaches [101]. In addition to being backed by sound reasoning and mathematical rigor, Krauss's work has several key properties which make it suitable for our prototype study. These will be presented in Section 5.2.2.2.

Similar to crowd dynamics, our research used traffic modeling techniques primarily for experimental means. Existing research was found to be acceptable, and was used without modification.

3.8 Work in Massively Multiplayer Online Games and Virtual Worlds

The simulation community has recently taken notice of the research potential of online games. Massively-Multiplayer Online (MMO) games can be seen as having a complementary feature set to human-in-the-loop simulation: they boast an extraordinarily large number of users, but serve no research or training purposes and exist solely for profit or leisure. In Chapter 4, we will introduce a technique to take advantage of this feature set by integrating it into a simulation. This section will provide an overview of the technology and merits of online games and virtual worlds in general.

Many online games run on cheap, ubiquitous hardware such as the i686 architecture coupled with any of the major vendors' graphics cards. Rapid innovation has led to games like *Spiral Knights*, which runs on almost any computer using a combination of Java and OpenGL, features an immersive 3D world, and operates at 30 frames-per-second (fps), dropping down to 25 fps only for frenzied action scenarios. Figure 3.7 shows the *Spiral Knights* client in operation with an fps counter overlaid.

FIGURE 3.7: The online game *Spiral Knights*, running reasonably fast on a modest machine from 2010.

As a point of reference, we consider 30 fps more than adequate for most immersive online experiences. Feature length films have been standardized to 24 fps since the 1920s, with only the most recent movies filming at 48 fps [102]. Hand-drawn animations are inked at an even lower resolution (often 12 fps, or even 6 fps for slower scenes) and then simply duplicate frames to reach 24 fps [103]. Information collected by the Insomniac games studio, which produces action-intensive interactive games, showed no correlation between frame rate and either sales or reviewers' ratings of games rendering at 30 versus 60 fps. We are not aware of any studies specifically measuring the effect of framerate on immersiveness, but the subjective evidence presented indicates that 30 fps is adequate for action-intensive scenes, with lower framerates suitable for slightly less frenetic virtual worlds.

Online games generally measure success by the size of their user bases. As the number of connected users increases, however, scalability quickly becomes an issue.

A great deal of work has been done by both commercial entities and researchers towards this end. Valadares *et al.* provide a good overview of the challenges faced and common techniques in the field [104]. Online virtual worlds typically require a small number of *transaction* updates to the server's database, and a large number of *self-state* updates to users in the immediate area. As the latter overwhelmingly affects performance, scalability has typically been achieved through either zoned partitioning of the virtual world or *area-of-interest* (AOI) techniques which restrict the broadcast of updates. These two scalability mechanisms are based on the observation that network bandwidth is usually the limiting factor in large-scale virtual world performance [105]. Lesser-used techniques include *distributed memory caching*, which achieves some amount of partition tolerance through key distributed, and *transactional memory*, which functions as an optimistic approach to consistency maintenance. Overall, though, much of the field focuses on the improvements gained by AOI techniques, with results ranging anywhere from two to seven times the number of agents [106] [107].

The problem of consistency in the face of heterogeneous client connectivity is explored in more detail in [108], which states that *absolute consistency* is neither possible nor necessary, and further categorizes consistency as either *causal*, *observational*, or *space-time*. A consistency approach based on dynamic scene-graphs is then recommended. In terms of raw numbers, [109] requires anywhere from five to ten servers to support 1000 connected users, and performs a thorough analysis of the effect this has on the per-user consistency of the virtual world. Recently, there has been a trend towards peer-to-peer (P2P) networked systems rather than traditional client-server models. For example, the *Badumna* network suite achieves a 75% reduction in network traffic my migrating non-critical operations to a P2P system [110]. Another novel research initiative involves the application of the REST principles to multi-user environments. The principle of "statelessness", combined with selective caching of large datasets, enabled a sample game world to scale up to 1,600 simultaneous agents [111]. Studies such as these are pushing the boundaries of what we can expect from virtual environments in the future.

Partly as a response to the promising work done for online games, Adobbati *et al.* developed a multi-agent infrastructure called *Gamebots*, which runs on top of the popular Unreal Tournament engine [112]. The early version of this system, developed in 2001, was limited to 32 simultaneous connections and suffered from weak support for agent-based techniques such as domain ontologies. It was later re-implemented as part of the *Pogamut 3* platform, which added full support for agent-based simulation [113]. The Pogamut 3 perception cycle operated at 10 fps, with the number of connected agents limited only by the Unreal Tournament server. Following the success of Pogamut 3, the *CIGA* middleware was designed to further the development of intelligent agents in virtual environments, particularly with regards to performance [114] [115]. The CIGA

developers noted several reasons why agent-based systems were not able to use existing MMO technology as-is. These range from the conceptual (game engines deal with low-level representational requirements while multi-agent systems are concerned with high-level semantic abstractions) to the practical (typical multi-agent systems feature non-durative agent actions, while game engines require reasoning to be performed over the duration of an agent's action), and help to explain why the earlier Gamebots system suffered from such poor performance. The CIGA developers explored the use of subscription filtering, tight coupling of an agent's physical and cognitive sensing procedures, and caching of the semantic world model, and tested up to 150 connected agents in non-arbitrary contexts [115].

3.9 Comparable Existing Techniques

3.9.1 Existing Simulation Technology

Having presented a review of existing relevant work in the field, we will now attempt to consolidate and compare the existing techniques which are most similar to the perennial simulation framework. Unlike the earlier literature review, this section will focus specifically on where each similar technique is lacking, with detailed comparisons to the framework itself as presented in Chapter 4. The reader may wish to review Section 4.2 before continuing with this section, as framework-specific terminology is used heavily.

The technologies that most resemble perennial simulation are DDDAS, the CAVE the HLA, and symbiotic simulation. Table 3.1 lists these in the first column, with possible roles they may fulfill as framework components in the second column. The third column lists shortcomings of each technique when compared to the perennial simulation framework. We will now address each candidate in greater detail.

DDDAS explicitly focuses on the relationship between the data being sensed at runtime and the quality of the running application (usually a simulation) as a result of this data [21]. Research from this field has developed several means of steering the measuring process when faced with an insurmountable quantity of data [40] [116] [63], and would certainly prove useful to perennial simulations as they scale in size. As a paradigm, DDDAS is inherently powerful enough to handle any task; however, its current form lacks several key refinements required for crisis management simulation. For one, DDDAS is overly focused on real-time applications, implicitly ruling out *hindsight* and *foresight* studies. The ability to incorporate real-time data can only be applied in real-time, unlike our framework's sophisticated history windows and swappable sensors,

Technique	***Strengths, Applications***	***Lacking Elements, Limitations***
DDDAS	• Dynamically sample *sensescape* • Manage very large *sensescapes*	• Historical "What-If?" simulation • Strong hierarchy of sensors and their capabilities/availabilities • Strongly defined role for Implementers • Tools for Implementers to quantify crises and their possible solutions
CAVE	• Very high immersion for MMOHILS • Potential for navigating *information* • Clear benefits to training exercises	• Essentially a visualization technique; not a simulation-centric one • No physical/virtual mapping; virtual worlds are isolated • No means to address validity, esp. auto-validation
HLA	• Scalable via interest management • Flexible via object models • Possible foundation for implementation	• High overhead for adhering to federation's assumptions • Benefits may not be realized, esp. if used within one organization • Over-emphasizes training scenarios
Symbiotic Simulation	• Used centrally by perennial simulation	• Unable to cope with crisis-required performance metrics • Only applicable to one phase of crisis management (*Response*) • Lacks insight from the crisis management point of view

TABLE 3.1: Comparison of perennial simulation to similar existing techniques

which can be reconfigured as necessary to speculate about hypothetical solutions to historical crises. At the other end of the spectrum, DDDAS does not provide a suitable enough role for **Implementers**, thereby limiting its usefulness for symbiotic decision support. Certainly some DDDAS studies —such as the fire spread model in [40]— function acceptably within the domain of crisis management. In fact, one may argue that our work is a subset of DDDAS in the strict definition of the term. However, we find this classification to be an oversimplification which masks the focus our framework places on crisis management simulation. For example, [40] relies on an oversimplified performance metric, as addressed in Section 3.1.1. Providing a realistic tracking of fire spread is useless to Implementers unless they can experiment with the model and quantify their options according to various, often conflicting metrics. Put another way, arguing that

perennial simulation is a subset of DDDAS is akin to arguing that it is a subset of simulation in general —true, perhaps, but irrelevant to our ultimate goal of addressing the needs of crisis management simulation.

The CAVE is a virtual reality environment with several notable benefits. Its hardware requirements are relatively modest: essentially five screen projectors and several sets of 3D glasses. Multiple users can explore the same virtual environment at the same time, navigating around virtual obstacles as well as each other [117]. The immersion level is very high, as users' physical steps directly correlate to their movements through the virtual room. As such, a CAVE would be an excellent means of achieving higher immersion in the perennial niche technique of MMOHILS. Furthermore, the ability to create flat "panels" that users can interact with permits a whole new range of potential applications in community training [118] [119]. There is a cheaper, single-screen alternative to CAVE called "ImmersaDesk", which achieves some amount of 3D immersion at a fraction of the cost, and is fully compatible with models developed for CAVE [117]. New consumer technology such as the Oculus Rift headset may yet bring affordable VR immersion to the masses [120]. Despite these advantages, the CAVE and its variants are solutions to the narrow problem of virtual world immersion, and lack the general scope of our research. Putting crisis management aside for a moment, even the niche configuration of MMOHILS has a broader scope than the CAVE. The former serves as a means of gathering data for simulations, and addresses the concerns of verification and validation that are essential in the domain of simulation. The latter exposes a sophisticated API for building and deploying virtual worlds, but provides no mechanisms for scrutinizing their correctness. Returning to the broader issue of crisis management simulation, perennial simulations offer a strong notion of *interacting* worlds that the CAVE lacks. The idea of *virtual studies* and *virtual establishment* provide sophisticated methods for reasoning about interconnected physical-to-virtual spaces while carefully managing sensors and effectors so as to avoid accidental contradictions. In addition, perennial simulation re-uses symbiotic simulation's idea of a constantly-running model to allow for automatic validation of a subset of model assumptions; this feature is also absent in the CAVE. While it serves as an excellent solution to the problem of high-fidelity virtual world creation, the CAVE would serve better as a component of a total solution based on the perennial simulation framework, and is clearly incapable of replacing it.

The HLA (and, by extension, its predecessor DIS) consists of interacting simulations called "federates" which adhere to compatible timing and serialization protocols. It boasts an impressive feature set, such as language-agnostic cooperation among federates, the ability to utilize multiple time management strategies simultaneously, and excellent scalability of world size via sophisticated information management techniques

[45]. That said, the HLA lacks the ability to be agile, requiring a great deal of development effort to use effectively. As a result, researchers have sought several lither solutions, each with a reduced feature set but a more rapid development process [22] [121]. An implementation of the perennial simulation framework may use the HLA as an enabling technology, thereby gaining all of its benefits as well as its flaws. Our own work eschews this approach, based on the observation from Figure 4.1 that both the creation and primary use of a perennial simulation for crisis management are contained within the same organizational entity. Thus, we expect that development resources can be conserved by relaxing the restrictions of the HLA and supporting only a reduced feature set. At that point, it is misleading to continue refering to that subset with the standardized term "HLA". In addition to this narrow limitation, the HLA also suffers from the broad issue of being over-focussed on training exercises. Although *hindsight* and *foresight* studies are not precluded by the HLA, its primary use for training has led to design decisions which favor this use. The already cumbersome HLA becomes even more time-consuming to set up for scenarios such as *mirrored* worlds —a flaw which it shares with the CAVE. Beyond these practical considerations, the HLA "lacks facilities for agent-centric sensing and acting" [114], reducing its utility for agent-based, virtual world simulation. In summary, though the HLA may serve as an adequate foundation for a comprehensive implementation of the perennial simulation framework, it is hampered by too many design requirements that incur a cost yet provide no appreciable benefit for perennial simulations.

Symbiotic simulations are designed to interact with physical systems in "mutually beneficial" ways [20]. The dynamic feedback loop which optimizes a real-world system was covered in Section 2.1.1, and extending this nascent technology is a key element of this thesis. Given that its inclusion in the perennial simulation framework is a foregone conclusion, all that remains is to evaluate its shortcomings. First, symbiotic simulation is myopic in its assessment of the physical system. Univariate performance metrics are abundant, and "What-If?" analyses are expected to allow straight-forward comparisons. Analysis of alternatives usually does not require human intervention. Crisis management simulation, on the other hand, can affect human lives and livelihoods. Crisis management personnel (**Implementers**, in the parlance of perennial simulation) may be forced to choose between several imperfect solutions, in part due to tight time constraints. The importance of "What-If?" analysis in a symbiotic decision support context is not to optimize the system *per se*, but rather to provide Implementers with the means of evaluating alternative scenarios to the maximal extent possible in the time available. Second, symbiotic simulation provides only an operational-level solution (that of how to constantly optimize a dynamic, physical system) with no means of approaching tactical

or strategic problems. In terms of Figure 3.3, symbiotic simulation may be an appropriate tool to use for the *Response* phase, but attempting to increase knowledge of the system to mitigate future crises is far outside of its scope. For crises that cannot be fully eliminated, the *Information Update* and *Preparedness* phases of Figure 3.3 exist to ensure that future variants of the same crisis are mitigated based on experience gained addressing the current crisis. Finally, symbiotic simulation is fundamentally a narrowly-focussed technique from the field of simulation. The perennial simulation framework provides several tools (such as MMOHILS) designed to enable, among other things, community training exercises and virtual re-enactments of crises. To this extent, it is a true trans-disciplinary approach to the problem of simulation for crisis management. Relying on techniques from only one half of the trans-disciplinary divide risks alienating experts on the other side; a comprehensive solution must incorporate the state of the art from both domains.

In conclusion, although there are some techniques which bear a similarity to our perennial simulation framework, none of them target the same niche or provide a truly comparable feature set. This is to be expected; if a sufficient technology already existed, there would have been no need to explore a new conceptualization. That said, our framework builds off of several of the techniques discussed above, and anyone implementing the framework would do well to study all of them and borrow existing solutions to various narrow problems.

3.9.2 Existing Software Engineering Frameworks

A great many frameworks exist to coordinate reuse within large-scale software systems. Our framework's reference implementation deals primarily with re-use at the conceptual level, as well as providing a small (but effective) amount of library-level reuse through shared components and inheritance. Nonetheless, it is pertinent to briefly review existing software engineering frameworks so that future implementations of the perennial framework may target software reuse more effectively.

Software reuse is a well-studied problem in general, although its success often varies by domain. In addition to practical matters such as cost, successful re-use relies on applications sharing some commonality that can be exploited [122]. A variety of software frameworks exist to facilitate this process, such as COM and CORBA, which allow the creation of components written in different programming languages to communicate and interoperate with ease [123] [124] [125]. Similarly, the FIPA specifications manage lifecycles and encapsulate message-passing behavior among heterogeneous interacting agents within an agent-based software system [126] [127]. One issue with software reuse

is the method by which one searches the repository of developed software libraries to find existing compatible components. To this end, solutions such as the FIPA-compliant JADE framework have introduced *yellow pages* services to aid in discovery [87].

Research on software reuse has also been undertaken in the field of computer simulation. The HLA, as discussed previously, caters to simulation interaction and time synchronization, achieving reuse primarily through its Base Object Models (BOMs). These models "contain the essential elements for specifying aspects of the conceptual model and documenting an interface of a simulation component that can be used and reused in the design, development, and extension of interoperable simulations" [128]. Similarly, the Discrete Event System Specification (DEVS) aids interoperability by representing discrete event simulations using powerful formalisms [129]. One criticism of these systems is that they only allow re-use of a component "as-is", rather than providing for further combinations. A good overview of the state of the art in model composability, reuse, and validation is provided in [130].

Chapter 4

Proposed Framework

The previous chapters established the simulation community's need for an enduring system which is robust for a myriad variety of studies; that is, one with *perennial* elements. This chapter will fully elaborate on the proposed *Perennial Simulation System* as the primary theoretical work of this thesis. First, it would be prudent to carefully construe exactly what such a perennial system would entail. The most reasonable way of doing this is to examine the lifecycle of such a system from the viewpoint of its intended usage. Following this, we will elaborate the proposed framework's components as they pertain to the system at a high level of abstraction. Finally, we will present our implementation of this general framework alongside a discussion of the various choices that had to be made during the implementation phase.

4.1 Design

4.1.1 Conceptual Overview of Creation and Usage

The total lifetime of a perennial simulation is split between a **creation** and a **perennial** phase, the latter of which provides the entire motivation for using a perennial system in the first place. Each phase encompasses a distinct timespan, as illustrated in Figure 4.1. The **creation** phase is concerned with the design and construction of the system, and is similar to its counterpart from classical simulation. After proper validation, the system enters into the **perennial** phase, where it is put to its intended use. The process may be restarted at the creation phase if expansion of the existing system is required. This creates a cycle which is common in most simulation undertakings; it is the internal makeup of the perennial phase which distinguishes a perennial system from a classical one, and provides a clear increase in value.

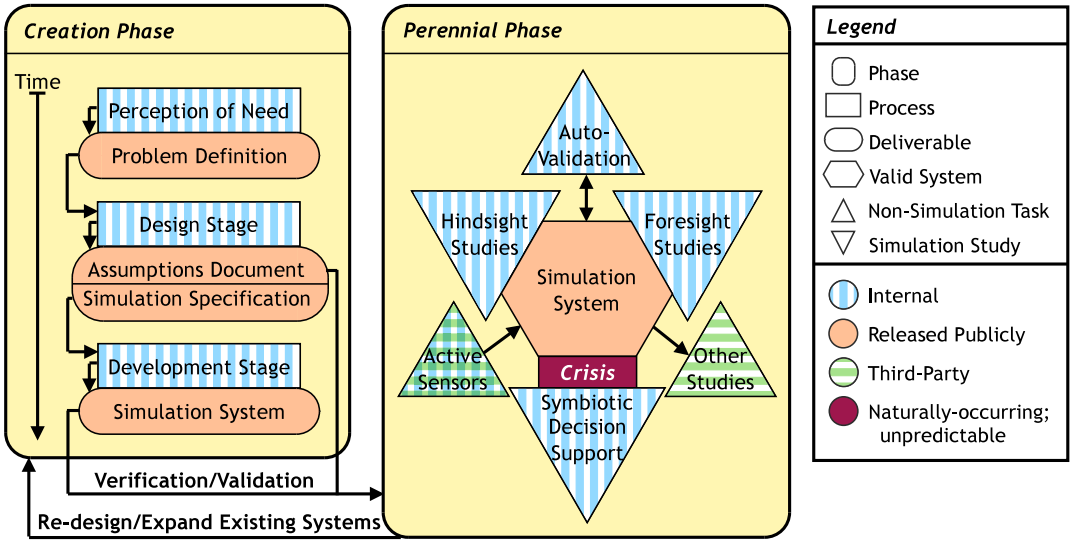

FIGURE 4.1: Conceptual overview of a perennial system.

The **creation** phase proceeds linearly in time from the inception of the study until the system's complete construction. It consists of a series of processes[1] tightly coupled to deliverables. Observation of a real-world problem spurs people in positions of responsibility to seek a solution. Simulation is selected as an appropriate tool which requires, as input, a document that clearly and completely defines the actual problem. This enables domain experts and simulation engineers to commence the design stage; their goal is to prepare two additional documents. The first of these explains the assumptions the simulation team is relying on, and is retained until validation begins. The second document lays out the details of the entire simulation system. It is used by engineers and programmers in the development stage to realize the actual simulation system. Following the system's creation, verification and validation proceed as normal [131] [59] [15], making use of both the assumptions document and the simulation system itself.

In each of these coupled sub-steps, the term "internal" is used to describe the entities performing the planning, design, or construction, while the documents they produce are "public". Internal entities share responsibility for creating and maintaining the simulation, in addition to having the same usability requirements. This is in contrast to "third-party" agents (introduced in the second phase), which only share the latter. Both types of agents need not reveal the exact details of their contributions to the system; "public" items, on the other hand, are always fully visible to any entity regardless of their affiliation. Finally, the notion of an "unpredictable" event is used to capture the essence of crises, which for many reasons occur with little to no warning at unknowable intervals.

[1] Here, the term "process" is used to refer categorically to development meetings, planning sessions, and other constructive events in time.

After construction and validation, the system enters into general use, also called the **perennial** phase. A simple linear time flow is no longer sufficient for understanding the system, as multiple parties vie for its use at arbitrary points in time. Hence, Figure 4.1 depicts the simulation system at the center of all these requests, some of which are now from external third-party entities. The simulation studies are tightly coupled to the simulation system, and are always run by internal groups. Hindsight studies assess historical events based on existing data and other related studies. Foresight studies predict the impact or likelihood of potential future events. They rely on hand-coded test data and extrapolations on or distributions of existing data, in addition to several "non-simulation" tasks. Finally, at the moment of crisis, "What-If?" analysis can be employed to provide decision support, thus leveraging the full power of symbiotic simulation. To avoid confusion, this approach will be referred to hereafter as **symbiotic decision support**.

All core simulation studies require supporting tasks which are not themselves pure simulations. Specifically, input from constantly active sensors must be processed, and the system must undergo constant auto-validation. The active sensors may either be maintained internally or by a third party, while auto-validation is intrinsically an internal task. In addition, other studies may be run using the copious amounts of data gathered by the simulation system. Such studies are generally undertaken by third parties. Inward-pointing arrows on Figure 4.1 denote which of these tasks contribute to the system, while outward-pointing arrows denote those which merely benefit from its existence.

Figure 4.1 captures the most fundamental form of a perennial system's creation and subsequent use. As with any abstraction, this conceptual breakdown should not be seen as rigid or all-encompassing. For example, a multinational perennial system might feature such extreme distribution of control that the term "internal" loses all relevance. In this case, developers should extend the perennial lifecycle diagram with more precise definitions relevant to the task at hand.

4.1.2 Design Goals

Creating a system which categorically fulfils the usage requirements of the previous section should merely be one of the marks of a successful framework. Proceeding from objective measures of practicality, we will now list a set of subjective qualities which are vital to ensuring our framework can be used with efficacy.

The quantitative design goals are as follows:

- The framework should be scalable up to a reasonable number of agents (several dozen to several hundred) within a virtual world of city-like proportions.
- The use of symbiotic simulation should be practical within the tight time constraints of crisis management simulation. Strategies must exist for sacrificing a small amount of accuracy for a large gain in performance.
- Perennial simulation must be shown to have a noticeable benefit with respect to the most similar existing technology. In particular, its improved decision-making accuracy must be demonstrated.
- Symbiotic simulation must be shown to be effective in scenarios with crisis elements and human factors. In particular, we expect that symbiotic feedback will improve egress times in building evacuation scenarios.

The qualitative design goals are as follows:

- The framework should be **modular**. Functionality should be encapsulated in a way that encourages reuse and eases the task of reconfiguring the system. Similarly, when new components are required, they should be simple to define as extensions of existing ones.
- Adhering to the system should require **minimal cognitive overhead** for veterans of simulation and systems design. This will be accomplished by designing lightweight components through an appeal to parsimony. The framework as a whole should not be arduous to implement or use.
- Despite its simplicity, the framework must remain **powerful** and **flexible** enough to meet the requirements set out in Chapter 2. It should be fully capable of managing a wide variety of studies, including *foresight*, *hindsight*, and *decision-support* studies.
- As much as possible, the framework should adequately **capture reality**, including complex phenomena like augmented reality systems and virtual spaces. One of the primary purposes of the perennial simulation framework is to reason about modern constructs such as these, and to leverage them to enable new and interesting studies that push the boundaries of simulation.

The aforelisted qualities were used as guiding principles throughout the creation and design of the perennial simulation framework. Some are less immediately relevant, and will grow in prominence once we begin specifying the implementation.

4.1.3 Framework Scope

Before detailing the framework, an effort will be made to narrow its scope and identify key novelties. As explained in Section 3.9, the perennial simulation framework builds off of the symbiotic feedback loop present in [20]. The virtual environments of the CAVE [117] and the HLA [45] were inspirations for the virtual world interactions of perennial simulation, but were found to be overly-focussed on technical implementation details while simultaneously ignoring the implications therein. Our analysis of these interactions led directly to the development of MMOHILS, one of perennial simulation's most useful niche configurations. These research elements form the primary technical motivations behind the development of perennial simulation as a whole, and establish its scope thus:

- The perennial simulation framework is most suitable for scenarios where a symbiotic simulation must be integrated with a physical location of interest.
- The framework is primarily to be used for running *foresight*, *hindsight*, and *decision support* studies, with an emphasis on the tight time constraints inherent in crisis management scenarios.
- The framework was designed to manage a myriad combination of sensors and worlds (physical and virtual), with a specific focus on their interactions, the ability to replay historical data over different scenarios, and the potential for "What-If?" analyses of varying types.

Perennial simulation was also motivated by the lack of a proper trans-disciplinary tool for crisis management simulation. In this respect, it builds off the work of [38], [56], and [90], which establish simulation's potential in the field, as well as [57] and [58], which establish resistance to the technology, especially from experts. The lack of a credible, powerful tool for applying simulation techniques to crisis management tasks was influential in refining the scope of perennial simulation, having the following effects:

- The framework specifically pertains to crisis management simulation; its intended users are crisis management experts and simulation scientists who are working together in this trans-disciplinary field.
- The framework can be applied to tasks which span all five stages of crisis management. In particular, it is suitable for the cyclical nature of crises as observed by various disaster relief organizations [4] [37].

- A perennial simulation has a long lifecycle punctuated by periods of intense use (corresponding to crisis events or community training exercises). Studies which intend to re-run historical crises and evaluate alternative resolution strategies are within the scope of perennial simulation.

Finally, several minor adjustments were made to the scope of the research, based on similar work in virtual world environments and game-based training, such as [121], [43], and [67].

- The framework's primary niche configuration, MMOHILS, is used to gain data from areas which require monitoring human behavior but are typically difficult to observe directly.
- The framework exists at a high level of abstraction, organizing conceptual elements without limiting the use of any particular implementation.

The virtual aspects of the framework, especially MMOHILS, are of particular use to crisis management simulation, as they broaden the set of plausible study areas and boost preparedness exercises such as community training. Users of the framework will note that there is no comparable alternative in this regard. The organization of the various sensor data can to some extent be supplemented by techniques from DDDAS [21], although the motivation behind this technique is different. The most novel aspect of the framework is its comprehensive approach to crisis management simulation, as it embodies a solution which favors both of these disciplines easily. This is evident from the lifecycle approach to perennial simulation, which is illustrated in the following section. This allows the framework to provide a higher degree of information superiority over crises than existing methods. Finally, the main challenge to overcome is that of *credibility*, from a simulation point of view. A perennial simulation may be valid, but if the Implementers are unwilling to trust its advice then it will inevitably be wasted.

4.2 A Framework for Perennial Modeling and Simulation

Given a clear understanding of the lifecycle of a perennial simulation and its built-in qualities, it is now appropriate to detail the framework one would use to realize the system. Where clarity is required, first-order logic formulas will be used to resolve potential ambiguities. While the building blocks of this framework bear some similarity to traditional simulation components, each has been reworked and carefully combined

to achieve the goals previously listed. From this standpoint, the perennial framework presented is an instrument of methodology rather than, e.g., a software library.

In addition to the contribution of the framework itself, a realistic implementation will also be detailed. Unlike the *framework*, which exists as an instrument of methodology, the actual *implementation* must be realized as a software library. As such, it will be presented as a modular system which is written in a high-level general-purpose programming language, carefully considers the tradeoffs of various implementation decisions, and acts as an independent contribution. Two of the studies presented in Chapter 5 use this implementation in one of its various incarnations, while the remaining study depicts it in a nascent form.

4.2.1 Top-Level Framework Overview

The components and interaction of a perennial simulation are detailed in Figure 4.2. The simplest relationship to grasp is between the **Real System**, which includes relevant physical and virtual locations of interest, and the **Implementers** or **Controller**, who are attempting to study some aspect of these locations. When a crisis occurs, the **Implementers** are in need of symbiotic decision support, whereas the **Controller** attempts to optimize the system during foresight and hindsight-driven studies. The remaining components in Figure 4.2 exist to expedite or enhance usage of the system as a whole [132].

The subset of the real system which can be monitored by sensors forms the **Sensescape**, while the analogous concept for effectors is the **Effectscape**. Individually, **sensors** read data from the real system, while **effectors** are used to change the system based on the Implementers' decisions. The remaining system is either concerned with simulating the crisis (**Models**, **Simulations**, and various **Validation** studies) or with presenting visualizations of it to the Implementers. Note that the real system may contain virtual worlds, populated by virtual users which receive visualization data and interact with Implementers in a similar fashion to their physical counterparts. Finally, the **Converter/Agglomerator** exists to help sensors convert data points between various expected formats. This, along with the **Access Layer** and **DB**, will be discussed in Section 4.2.3.

The internal workings of each component in Figure 4.2 will now be described completely; all sub-components will be enumerated using tables. Where ambiguities exist, first-order logic formulas will be presented to help clarify inter-component relationships more explicitly. Each first-order logic formula uses terms defined in Table 4.1. Complicated terms will be provided as separate, un-numbered definitions as they become relevant.

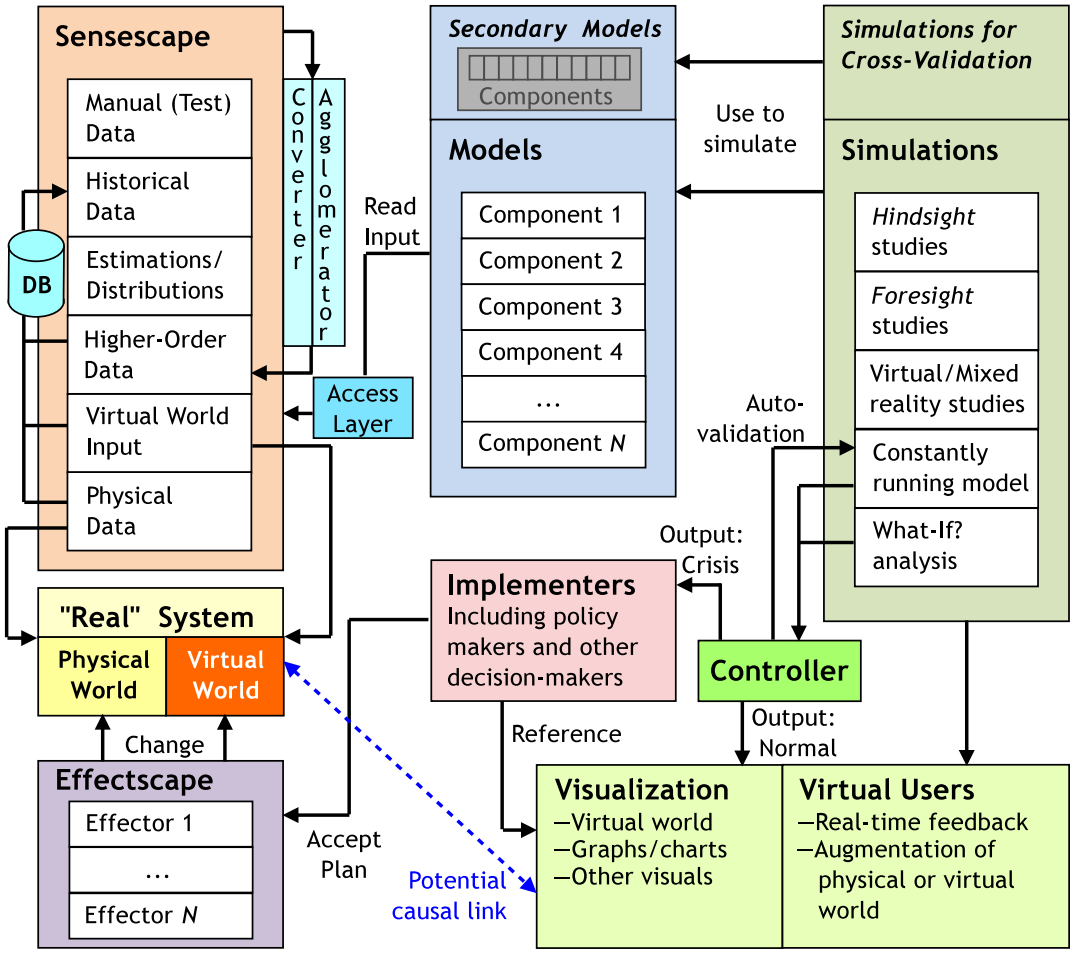

FIGURE 4.2: Perennial Simulation framework

Term	***Meaning***
$I = \{i_1, i_2, .., i_n\}$	The set of all identifiers
$key(i_n)$	The key of identifier i_n
$value(i_n)$	The value of identifier i_n
$W_P = \{w_1, w_2, .., w_n\}$	The set of all physical worlds
$W_V = \{w_1, w_2, .., w_n\}$	The set of all virtual worlds
$W_S = \{s_1, s_2, .., s_n\}$	The set of all sensors; the *sensescape*
$W_E = \{e_1, e_2, .., e_n\}$	The set of all effectors; the *effectscape*

TABLE 4.1: First-order-logic terms used to describe the perennial framework

Optional	***Property (Key)***	***Possible Values***	***Purpose***
N	**name**	*physical.X* *virtual.Y*	Identifies the world. E.g., *physical.1*, *virtual.1*
Y	**description**	(*text*)	Brief rationale for why this world is included in the simulation study.
N	**location**	(*identifier*)	Physical/virtual worlds that share the same identifier are assumed to be capable of interaction.

TABLE 4.2: Properties of a World in the Real System

4.2.2 Real System

The ***real system*** being studied is composed of both the physical environment and a number of virtual worlds, some of which correspond directly to the physical components. Table 4.2 details the properties of each world. The **name** is used to identify the world to the simulation; all names are numbered starting from zero. A brief **description** may help system users to understand the purpose of this world in respect to the simulation as a whole. The **location** is an identifier; the only requirement for identifiers is that they satisfy an **identity** requirement in first-order logic, as shown in Equation 4.1.

$$\begin{aligned} &\forall i_1, i_2 : i_1 \in I \land i_2 \in I \Rightarrow \\ &(key(i_1) = key(i_2)) \land (value(i_1) = value(i_1)) \Rightarrow i_1 = i_2 \end{aligned} \tag{4.1}$$

From Table 4.1, we know that I is the set of all properties, where any given property i_n is composed of a key k_n and value v_n. In other words, any two properties with the same keys will be considered equal if their values are the same. In case of the **location** value from Table 4.2, reasonable identifiers might be a name (e.g., "Building #10") or a set of geographic coordinates (e.g., "*47.362101*LAT,*-122.054144*LNG").

The importance of the virtual world will vary in any given study. Some studies may have several virtual worlds, while others will require none. Conversely, some studies may have no need of the physical world, and will exist exclusively at virtual locations. Physical world locations are always distinct —satisfying Equation 4.2— while virtual locations may overlap. If two worlds share the same location, they can influence each other to some degree. More formally, considering the sets of physical and virtual worlds, Equation 4.3 holds. Conversely, if two worlds do not share the same location, they are said to be *independent*, as Equation 4.4 states. (Note that *independence* is a reflexive

relationship.) Some of the consequences of *influence* are described in Section 4.2.5, with a more complex example presented in Section 4.2.6.

$$\forall w_{p1}, w_{p2} : w_{p1} \in W_P \wedge w_{p2} \in W_P \wedge w_{p1} \neq w_{p2} \Rightarrow \\ loc(w_{p1}) \neq loc(w_{p2}) \tag{4.2}$$

$$\forall w_p \in (W_P \cup W_V), \forall w_v \in W_V \\ : loc(w_p) = loc(w_v) \Rightarrow Influence(w_p, w_v) \tag{4.3}$$

$$\forall w_1 \in (W_P \cup W_V), \forall w_2 \in W_V \\ : loc(w_1) \neq loc(w_2) \Rightarrow Independent(w_1, w_2) \tag{4.4}$$

The effective number of physical and virtual worlds will vary depending on the given study. All studies, however, will require at least one clearly defined world; the formula for this invariant is trivial to construct.

4.2.3 Sensescape and Effectscape

Recall from Table 4.1 that the ***sensescape*** is composed of all available sensors, while the ***effectscape*** consists of the set of actuators. Essentially, the former represents the portions of each world that can be *detected*, while the latter represents areas that one can *effect changes* on. It is common for these two concepts to overlap, as sensors and effectors may be tightly coupled if they meet the requirements of Equation 4.5, another symmetric relationship. Multiple sensors and effectors may interact to form a tightly coupled region, or they may even share the same hardware. For example, a temperature sensor might be hardwired to control the output of a heating unit.

$$
\begin{aligned}
&\forall w \in (W_P \cup W_V) : \\
&\quad \exists s \in W_S, location(s) = location(w) \wedge world(s) = w \\
&\quad \Rightarrow \exists e \in W_E), location(e) = location(w) \wedge world(e) = w \\
&\quad \Rightarrow TightlyCoupled(s, e)
\end{aligned} \tag{4.5}
$$

Table 4.3 details the properties of sensors and effectors. The **name** is used to identify the sensor, while the **target** explains the portion of the world being sensed. The **world** itself is also necessary, to distinguish similar **locations** accessible from different worlds. Sensors will not generate data outside their **range**, and an effector will only accept input that falls within this value. The former of these is shown in Equation 4.6; the latter is a trivial constraint on input parameters and is not formalized.

$$
\begin{aligned}
&\forall s \in W_S, \forall time : \\
&\quad [(time < valid\_datetime(s).start) \vee \\
&\quad (time > valid\_datetime(s).end)] \Rightarrow \\
&\quad !\exists data : time(data) = time \wedge sensor(data) = s
\end{aligned} \tag{4.6}
$$

Skipping ahead, the **restricted** property is used to toggle which effectors can only be utilized by implementers. In the case of decision-support studies, these represent cases where it is preferable for a human to make the final decision on whether or not to activate an actuator. For example, a building's sprinkler system might be automatic and non-restricted, while the ability to request assistance from the local fire department would require approval. The **restricted** flag only applies during decision support studies; during hindsight and foresight studies, any possibility can be considered. The final row in Table 4.3 allows users to expand the functionality of sensors and effectors by defining additional properties.

The remaining two properties do not apply to effectors, and are the key to establishing a diversification of sensor types. The **valid-datetime** specifies a bound on the time periods simulations may request data for. This property may also be set to "Infinity" or "Bound". The former states that the sensor is capable of generating values for any time-span. The latter implies that its availability is tied to another sensor. An additional flag, "Real-time", indicates that this sensor is capable of generating data while the simulation is running. A setting of "Infinity" implies that the "Real-time" flag is also

Optional	***Property (Key)***	***Possible Values***	***Purpose***
N	**name**	(*identifier*)	Uniquely identifies this sensor or effector. (E.g., "*sensor.water-level*")
N	**target**	(*identifier*)	Specify the modeled construct being sensed or changed.
N	**world**	*physical.X* *virtual.Y*	Denotes which world contains the target.
Y	**range**	(*text*)	Range of values measurable by sensors; range of values effectors accept as input.
N*	**valid-datetime**	(*Start*, *End*), *Infinity*, *Bound* + *Real-time?*	Specifies the range this data is valid for. "Infinite" data can be generated. "Real-time" is an optional flag, and means that data will become available later; bound sensors rely on other sensors' *valid-datetime*.
Y*	**mapping-function**	$S = \{s1, s2, s3\}$ $F(S) \rightarrow value$	Set of sensors, and a function which maps these sensors to a combined value.
Y+	**restricted**	*true* *false*	Whether implementer approval is required to activate actuation.
Y	**...**	(*Varies*)	Additional, user-defined properties

** only applies to sensors*
+ only applies to effectors

TABLE 4.3: Properties of Sensors and Effectors

set. Finally, the **mapping-function** reserves a set of sensors, and declares a function which produces values based on these sensors. It is used to combine the input of several sensors, or to convert the raw format from one sensor into another format expected by one of the models (respectively, the "Agglomerator" and "Converter" in Figure 4.2). For implementation purposes, the mapping function may be realized programmatically; it does not necessarily represent a function in the strict mathematical sense. This will be covered in greater detail in Section 4.3.

Given its formal description, we can broadly consider as "sensors" any data-generating entity which the models have access to. Some of these may gather data in real-time, while others provide historical data or estimations based on known distributions. The different sensor "types" listed in Figure 4.2 are all generated from the same template; there is nothing hard-coded about what defines a specific sensor type. As an example, consider *Distributions*, *Manual Data*, and *Higher-Order Data* —three sensor types omnipresent in the field of simulation. The **name**, **target**, **world**, and **range** of a set of sensor will

not help to differentiate them into types. The **valid-datetime** and **mapping-function** attributes, however, allow for a full specialization of sensors by category. Table 4.4 gives an example of how these attributes might be defined to create the three sensor "types" in question. It is important to note that, with the exception of historical sensors, these sensor types are entirely defined by their properties. Thus, a "physical" sensor that reconfigures its "world" property to "virtual" is now considered a "virtual sensor" by virtue of its configuration alone[2]. As a general modeling construct, it is valuable to represent sensors in a way that is flexible to the needs of the modeler.

Conceptually, it is possible to describe historical sensors using the same sensor attributes which serve to differentiate all other sensor types. This would have the unfortunate implication of requiring each model to completely reconfigure its sensor setup merely to switch between current and historical data requests. Instead, the **Access Layer** is defined to take a data request from a model and pick the relevant sensor, historical or otherwise. A database (**DB**) stores data from all other sensors, automatically creating a historical sensor for every other sensor type. Section 4.3 covers the intricacies of implementing this effectively.

The availability of sensors and effectors unambiguously defines the ability of one world to "influence" another. Given the set of sensors and effectors, and assuming that items introduced are not identical to each other unless explicitly stated, a *Mirror* relationship can be defined by Equations 4.7.

$$\begin{aligned} &\forall w_1, w_2 \in (W_P \cup W_V) : \exists s_1, s_2 \in W_S : \\ &(world(s_1) = w_1 \wedge world(s_2) = w_2 \wedge Influence(w_1, w_2)) \\ &\Rightarrow Mirror(w_1, w_2, location(w_1) \end{aligned} \tag{4.7}$$

That is, if two sensors belong to two different worlds which influence each other, the first world is said to *mirror* the second at that location. Similarly, Equation 4.8 holds for effectors.

$$\begin{aligned} &\forall w_1, w_2 \in (W_P \cup W_V) : \exists s_1 \in S, \exists e_2 \in E : \\ &(world(s_1) = w_1 \wedge world(e_2) = w_2 \wedge Influence(w_1, w_2)) \\ &\Rightarrow Tunable(w_2, w_1, location(w_1) \end{aligned} \tag{4.8}$$

[2]This assumes, of course, that the sensor's implementation is updated to target the aforementioned virtual world.

Sensor Type	***Attribute***	***Sample Value***
Estimations/ Distributions	name	*sensor.interarrival.exp*
	valid-datetime	*Infinity*
	mapping-function	$S = \varnothing$, F(S) = exp(1.2)
Manual Test Data	name	*sensor.dec10th-arrivals*
	valid-datetime	{Start; End} = {10/12/2006, 08:00; 10/12/2006, 18:00}
	mapping-function	$S = \varnothing$, F(S) = { (8:00, 1), (8:10, 2), (8:12, 1), (8:16, 1), etc. }
Higher-Order Data	name	*sensor.arrivals-per-hour*
	valid-datetime	*Bound*
	mapping-function	S = { s_1 = *sensor.interarrival.exp* } F(S) = { $g(0, 0, s_1.next())$ } $g = \lambda(x, y, z)$: (if x+z¿60 then y else $g(x+z, y+1, s_1.next())$
Physical Data	name	*sensor.room-occupancy*
	world	*physical*
	valid-datetime	{Start; End} = {01/01/2005, 08:00; 01/01/2006, 23:59} *+Real-Time*
Virtual World Input	name	*sensor.room-occupancy2*
	world	*virtual.2*
	valid-datetime	{Start; End} = {01/01/2005, 08:00; 01/01/2006, 23:59} *+Real-Time*
Historical Data	(*Special case; see below.*)	

TABLE 4.4: Sample Sensor Combinations

In other words, if a sensor for one world and an effector for another are at the same location, then the second world is tunable at that location from the point of view of the first world. *Tunable* sensors can be optimized by "What-If?" analysis, and "Mirrored" locations can be represented to virtual users in a virtual simulation study. These terms have other implications for visualization, discussed later.

It is important not to lose sight of the purpose of these components. Ultimately, sensors are used to describe what data can be gathered, and effectors are used to describe how implementers can propagate changes through the system. A great many sensor types can be modeled by the default property set, and many more are enabled with user-defined properties. New hard-coded types which require system-level support (like the "historical sensors") will generally not be required. They should only be created if an entirely novel situation is being modeled.

4.2.4 Models and Simulations

The ***models*** block contains the various models available to the ***simulations***; the latter providing the data processing necessary to leverage the former to generate useful output. The field of simulation contains several definitions for what constitutes a **model** and a **simulation**; the simplest defines the former as an approximation of an event, and the latter as a running instance of the model that allows for repeated observation [133]. Various sub-domains in simulation contain their own well-founded and stable definition of what constitutes a **model** and a **simulation**, and we do not wish to limit the applicability of our framework by being overly strict with our definition of the two. Therefore, we accept as valid any implementation that matches or extends the following definitions:

Model

A model is a representation of a real-world process that can be advanced through discrete time steps to recreate or approximate that process. Individual decision-making components of a model are encapsulated into independent entities known as agents.

Simulation

A simulation is a runtime representation of a model. Simulations track various sources of input via sensors, and may coordinate dispatching this information to all active agents. Running the same time-sliced sensor information through a simulation (assuming random number generators are reset to the same seed) will produce the same output; thus, simulations are repeatable. Simulations may interact with physical or virtual elements of the real system they are modelling via

effectors, which may in turn require implementers to have the final say in making crucial decisions.

Categorically, all of the simulations in the system fall into one of two groups: **primary simulations**, which represent the essential studies being carried out by the perennial simulation, and **secondary simulations**, which exist for a variety of miscellaneous purposes. Models are grouped the same way, and **secondary models** may add a number of other **components** necessary for their functioning. For example, instead of creating a sensor that produces data for a secondary model, one might create a component that reads in data from the other sensors and converts this to an acceptable format on-the-fly. The component system is designed to be flexible enough to handle "legacy" simulations written without interoperability in mind.

As previously stated, the **primary simulations** will be *agent-based* simulations. Agent-based simulations encapsulate behavior into entities which sense and modify their environment, and communicate with each other [134]. This technique has been shown to model complex human behavior in a fairly intuitive manner, and this intuitiveness can ease certain aspects of validation. Human behavior models will be necessary in most crisis management studies, as crises tend to feature a strong human component, and require more than simple physics-based models to capture the variability of human decision-making.

The **primary simulations** will generally belong to one of several categories. Most simulations will exist for hindsight, foresight, and "What-If?" guided symbiotic decision support. In addition, a constantly-running model is also needed both for auto-validation, and to trigger the "What-If?" analysis when a thresh-hold has been crossed or will be imminently. Finally, several simulations may be virtual in nature, which means that they take place in a virtual representation of a physical location. Using a virtual world reduces costs compared to real-world training exercises, and allows more dangerous physical locations to be simulated safely.

The **secondary simulations** and other secondary components are designed to encapsulate so-called "legacy simulations". A legacy simulation meets or exceeds the following definition; there is no corresponding definition of a "legacy model", since any secondary modeling components used by a legacy simulation are considered black-box details and maintain their own internal specifications.

Legacy Simulation

A legacy simulation is a runtime representation of a real-world process which is capable of reading from various sensors but may not operate effectors. Legacy

simulations may or may not utilize agents and discrete time step updates. Legacy simulations are repeatable.

Despite not being first-class components in the perennial simulation framework, legacy simulations are nonetheless important. Many excellent crisis-related models and simulations exist, though not all of them meet the requirements put forth by our framework. Treating them as legacy simulations enables some degree of interaction with our framework, allowing them to function as helper simulations or to provide data for cross-validation.

4.2.5 Implementers, Visualization, Virtual Users, and the Controller

A great deal of output is available from each of the "What-If?" analysis simulations; this output, along with that from the constantly running model, is sent to a ***Controller*** module, which may use the data in one of three ways:

- The ***Controller*** may send output to the ***Visualization*** block. This is the normal destination for output, as it provides graphs and charts which make the data accessible, as well as animations which convey the progression of a crisis. A special visualization target, ***Virtual Users***, capitalizes on the existence of the virtual worlds to allow for an immersive experience of the crisis.
- Output may instead be sent to the ***Implementers***. This group represents the policy makers who must review the generated scenarios and determine which plans to accept. They may also reference the visualizations.
- Finally, after some delay, output from a "What-If?" simulation may be compared with the actual output of the constantly running model to determine if the system's behavior was accurately predicted. If minor adjustments are necessary, the ***Controller*** may send feedback to the running model.

Supplemental to the controller's functionality, output from any simulation may be forwarded *directly* to the ***Visualization*** block. The idea here is that the ***Controller***'s logic is only necessary for real-time, symbiotic feedback, and any mundane output can be managed directly. Finally, note that the Visualizations may be **causally linked** to the virtual worlds, as indicated in Figure 4.2 with a dashed line. This linking has already been explained in equations 4.3, 4.7, and 4.8. By making similar assumptions to those used in Section 4.2.3 we arrive at one more descriptive property:

$$\begin{aligned}\forall w_v \in W_V, w_p \in W_P : L = loc(w_v) \\ : (Mirrors(w_v, w_p, L) \wedge Tunable(w_v, w_p, L) \\ \Rightarrow VirtualStudy(w_v, w_p)\end{aligned} \tag{4.9}$$

That is, if a virtual location mirrors *and* is tunable by a physical location, then it is possible to do a *Virtual Study* of that location. If the reverse is true (a physical location is mirrored and tunable) then we say the physical world is *Virtually Established*, as shown in Equation 4.10. This implies that changes in the virtual world can be used to modify the physical world as well, which could be useful for allowing implementers to actuate change from within a virtual world. Nonetheless, we are currently not concerned with the potential of virtual establishment within our framework.

$$\begin{aligned}\forall w_v \in W_V, w_p \in W_P : L = loc(w_p) \\ : (Mirrors(w_p, w_v, L) \wedge Tunable(w_p, w_v, L) \\ \Rightarrow VirtuallyEstablished(w_p, w_v)\end{aligned} \tag{4.10}$$

4.2.6 Example Niche Configuration: MMOHILS

The entire space of possible configurations within the perennial framework is vast, and most of these will be created on an *ad hoc* basis for any given study. Some configurations, however, are generally applicable to a specific, commonly-recurring set of studies, and as such are termed *niche configurations*. One niche configuration engineered for an early perennial simulation draws inspiration from human-in-the-loop simulation, and is called Massively Multiplayer Online Human-In-the-Loop Simulation, or MMOHILS[3].

Human-in-the-loop simulation, introduced in Chapter 3, is a technology which can be used to train humans in activities that would otherwise be prohibitively dangerous or expensive. As shown in Equation 4.11, the human's situation within the simulation loop is accomplished via sensors and effectors which act on a virtual training world and which that human has access to.

[3]MMOHILS is pronounced homonymously to "molehills".

$$
\begin{aligned}
&\forall v \in W_V, h : \\
&\quad (Human(h) \wedge InLoop(h, v)) \Leftrightarrow \\
&\quad (\exists s \in W_S : world(s) = v \wedge \exists e \in W_E : world(e) = v \\
&\qquad \wedge HasAccess(h, s) \wedge HasAccess(h, e))
\end{aligned} \tag{4.11}
$$

Extrapolating equation 4.11, we can arrive at the concept of a MMOHILS via a simple insight: if the virtual world is a *Virtual Study* of a corresponding physical system with similar sensor locations, then the *simulation* also benefits from the human in the loop. Continuing our example of pilot training, one might record the real-time reactions of the pilot to virtual sensor data and incorporate this into the behavioral model of an AI-controlled pilot. Moreover, the physical counterpart to the virtual world need not actually exist, so long as it is representative of the real-world conditions one wishes to study. In other words, a virtual environment can be created which is a *Virtual Study* of a hypothetical scenario, in much the same way that a human-in-the-loop simulation may generate weather conditions and terrain specifically to test a pilot's response. Some existing research approximates the functionality of MMOHILS, the CAVE environment being the preeminent example [66]. Unfortunately, most traditional human-in-the-loop simulations are concerned with single-user environments, or require dedicated and expensive hardware —to its credit, the CAVE suffers only from the former.

Starting from the formalized notion of a MMOHILS, further refinements were made by considering the strengths of human-in-the-loop simulation along with those of massively-multiplayer online games. The original paper ([135]) describes the goals of the MMOHILS niche configuration: to track a large number of users as they progress through a virtual environment that is also a simulation, thus providing a valid and valuable "human factor" which might otherwise be missing from comparable agent-based simulation techniques. The incentive scheme used to motivate virtual world participants is described in Section 4.3.3.2. Validation of MMOHILS generally uses traditional techniques with any "human factors" validated using psychological evaluations such as surveys. This is discussed in depth in Section 4.3.3.3.

Niche configurations such as MMOHILS are important because they encapsulate a complex phenomenon using the simple abstractions of perennial simulation. This allows anyone familiar with the framework to model said phenomenon without switching paradigms, thereby expanding the boundaries of simulation in general. It is left as an exercise to future modelers to extract other niche configurations from the perennial simulation-scape.

4.2.7 Benefit of Perennial Simulation Compared to Similar Techniques

Having detailed the framework and one of its niche configurations, the most pertinent concern is what benefit our framework provides over and above alternatives in the field. First, we would stress that the idea of a perennial simulation was conceived primarily as a result of a lack of sufficient existing alternatives, and that no single approach fulfils the exact requirements of the framework as detailed. This was discussed earlier in Section 3.9. That said, we consider a perennial simulation to be most similar to a traditional simulation —that is, one without perennial elements. In the absence of a perennial simulation framework, one would perform classical simulation studies and use them to train **Implementers** how to respond to a hypothetical future crisis. If the actual crisis exhibited different properties than expected, the Implementers would have to rely on their experience and intuition to guide their decision-making. Recall from Section 3.9 that a symbiotic simulation cannot be used as-is when the performance metric is non-trivial. This is often the case for crisis management applications, and is the motivation behind our creation of the perennial simulation framework. Our goal is to allow Implementers to leverage the maximum amount of information possible in the time allotted during a crisis event.

The perennial simulation framework affords several qualitative benefits, such as the ability to organize sensors more effectively for *hindsight* and *foresight* studies, and an effective mapping between physical and virtual worlds that enables powerful simulation techniques such as MMOHILS. These are important considerations, but the greatest benefit afforded is to **symbiotic decision support**, which is also the framework's most difficult benefit to quantify. To arrive at a reasonable measurement of this benefit, it is worth examining how human experts arrive at decisions under similar conditions of uncertainty and limited knowledge. Extensive research has shown that human decision-making proceeds by using *inference* to fill in gaps in existing information, with a higher weight on past knowledge. The order in which information arrives is thus important, as even experts tend to maintain their original assessments when presented with newer, conflicting information [136] [137]. Furthermore, research indicates a decrease in human creativity under conditions of high stress, but no corresponding decrease in general intelligence or decision-making strategy [138]. This supports our view that properly trained Implementers will react positively to simulation results presented at the time of crisis, even if stress conditions are high (but not unbearably so; see [139]). In addition, this emphasizes the importance of maintaining the simulation's credibility or, put another way, of ensuring that we are always generating results the Implementers can trust.

Some of this credibility can be established using a simulation without perennial elements to plan for crisis events in advance. However, the landscape of potential scenarios is too

vast to map entirely beforehand, and for crisis management the problem of uncertainty expands this search space exponentially. Recall that crises may destroy infrastructure which is also part of the *sensescape*. This has the adverse effect of introducing uncertainty into the sensor outputs. Such uncertainty is likely quantifiable, but may result in Implementers unsure of how to choose between multiple plans which are all valid under the current range of reported data. Here, a perennial simulation finds its niche, running decision-support simulations at the time of crisis to assess the implications of the current uncertainty level. In addition, the results obtained can be validated automatically as the crisis progresses, ensuring that any mistakes are quickly corrected. This provides Implementers with a level of information superiority that allows them to make the best possible inferences in novel crisis situations. Thus, we can arrive at a general hypothesis that adding perennial elements to a traditional simulation (via our framework) will lead to improvements in a decision-support environment with some data-related uncertainty. To support this, we build a model of the Implementers' decision-making process for our traffic incident scenario in Section 5.2.7, and use this model to gather data on the exact amount of benefit that perennial simulation provides.

4.3 Implementation

The framework as described is somewhat inchoate, in that it assumes that each component is trivial to realize. This is the result of an intentional effort to avoid dirtying the theoretical elements of the framework with the artifacts and limitations of any given representation. To definitively state the optimum implementation at any given point in time would defeat the purpose of declaring a general-purpose framework, as the technology available to create simulations is constantly improving. Thus, the previous sections focused on the perennial framework as a timeless theoretical and conceptual entity.

In this section, an implementation of the perennial simulation framework is provided with the goal of making the best possible use of current technology. As this goal's nature is subjective, we will first make some general remarks regarding key qualities of the library, possible implementation languages, and other design choices. Following this, a top-level class diagram will be shown. Finally, each component will be detailed fully, with any particularly confounding points expounded on in their own sections. Techniques will be suggested, and their tradeoffs analyzed. The reader should take note that while the framework is independent of the externalities of language, architecture, and platform, the reference implementation is burdened by the limitations of all three. We would therefore advise the reader to use caution in accepting these implementation choices at face value; they are based on the current state of simulation, which will certainly change

in the future. The framework as presented in Section 4.2 is less likely to deteriorate as technology progresses, and should be used as a starting point for future work on novel implementations.

4.3.1 General Implementation Details

It would now be prudent to examine several assumptions which motivate the design of the given implementation, all of which follow from the goals stated in Section 4.1.2. Each assumption, although subjective by its very nature, will be backed up by an appeal to reason which attempts to justify its inclusion on the list of desirable qualities. Following this, a set of implementation choices will be presented. These are more objective in nature, and follow from the accepted assumptions.

4.3.1.1 Implementation Assumptions

The implementation is assumed to have the following properties:

1. It will be presented as a linkable library.
2. It will use object-oriented programming techniques to facilitate interoperability.
3. It will strive to remain domain-agnostic, relying on users to affix further semantic detail to a core set of simulation components.
4. The visualization component will be separate from the remainder of the library.

First, the implementation will be presented as a linkable library. Regardless of what constitutes a "library" in the programming language chosen, the implementation itself will remain cleanly separated from any project which uses it. If possible, the library will be distributable as a single, packaged file. A clean separation of library code from the studies it is used in has several benefits. First, compatibility between different releases of the library will be easier to guarantee, as any additions unique to one particular study will be located outside of the library's namespace. Second, versioning releases of a single file is much easier, as its cryptographic hash value can be calculated and provided independently for verification. SHA-2 and FSB are two secure *cryptographic hash functions* which can be used to generate these checksums. The benefit of using a secure hash function (versus, say the popular MD5 function) is that it enables verifying downloads of the library file in a tamper-proof way.

Second, the implementation will use object-oriented programming techniques as a means of facilitating interoperability. Object oriented programming (OOP) is a programming paradigm which has been shown to increase interoperability through its encapsulation of methods and properties into common *class* descriptions that can be used to generate *objects* of a given class type [140]. Alternatives to OOP include prototype-based programming and aspect-oriented programming, among others. Each of these approaches, including OOP, has its own set of flaws. The reason we settled on OOP is that it provides a simple, easy to grasp abstraction for interoperability that is common across most modern programming languages[4]. Although the current implementation uses a single programming language, employing a widely-supported technique like OOP allows us to remain open to future expansion of the implementation across languages and platforms.

Third, the implementation will avoid, when possible, catering to a single domain of simulation. This will in turn require users of the framework to incorporate the semantics of their own domain of research. This decision fits with our goal of **flexibility** in Section 4.1.2. The alternative is to add provisions for specific domains, ensuring that the majority of our system will never be used in a single study. This would result in a bloated, difficult-to-maintain code base, and would fail to provide any real benefit, as the specification of any particular semantics is bound to fall short of what some percentage of real users will need. The goal of being able to **capture reality** was a guiding force behind the framework definition, and is particularly visible in the way that sensors are defined. We will therefore forward this design choice to the implementation stage, by ensuring that our system is agile enough to capture the syntax of any arbitrary simulation study without burdening users with semantics that they would prefer to add themselves.

Finally, the visualization component will be designed as a separate library. This requirement was added after initial work implementing a portion of the framework, and is necessary due to the extreme variety and incompatibility of visualization tools —even those authored in the same programming language. Despite a move towards cross-compatible high-level programming languages, modern visualization toolkits immediately impose large amounts of overhead. Even generic "windowing toolkits" like the GIMP Toolkit (GTK+) [141], IBM's Standard Widget Toolkit (SWT) [142], and Nokia's Qt framework [143] exhibit this flaw, as they either tie one to a particular operating system (GTK+) or programming language (SWT), or they are bound to obsolete programming practices and are incapable of upgrading (Qt). Newer, experimental frameworks such as Phoenix [144] exhibit a promising abundance of cross-compatibility, but tend to be restricted by

[4]The term "object-oriented" is not strictly defined. Here, we take it to mean that a language provides *polymorphism*, *inheritance*, and *encapsulation*. This definition excludes a small number of common languages such as JavaScript, Lua, and Pascal; however, each of these languages provides some means of approximating the first two properties (usually at the expense of the third).

the most primitive backend (Win32) and, as such, lack more powerful features which are well-established in the more powerful backends (Qt and GTK+).

Every project which uses our library will have its own visualization requirements, so separating the visualization component from the core simulation components will grant particularly assiduous users the ability to write a drop-in replacement without invalidating the core library's checksum. Furthermore, the loosely coupled nature of sensors in a perennial simulation might actually obviate the need for visualization in any centralized way. Consider a typical building thermostat, which contains a tightly coupled sensor, effector, and model. The thermostat measures the room's temperature and attempts to adjust it to match the desired temperature set by the user. Internally, a model is used to avoid jarring temperature changes, likely via a simple backoff algorithm. This type of subsystem does not require built-in visualization support, as it is intended to be treated as a unit rather than a set of components. It can thus operate unmodified and independent from the rest of the perennial framework. This conglomerate sensor system likely has its own vendor-supplied visualization toolkit (written in an arbitrary language) which can be loosely coupled to the perennial visualization component using, e.g., TCP sockets. If the visualization component was tightly coupled to and reliant on the simulation component, such *laissez-faire* integration would not be possible.

4.3.1.2 Implementation Choices

The assumptions provided in the previous section greatly simplify the remaining implementation decisions. These decisions are designated *global* or *local* in scope, depending on whether they affect the entirety of the reference implementation or only one aspect of it. Local decisions will be covered as they become relevant. The global decisions are more far-reaching:

1. Choice of programming language
2. Choice of visualization toolkit
3. User data customization strategy

The first far-reaching decision is which programming language to use to realize the reference implementation. Table 4.5 lists several potential candidates along with their key properties. For the purpose of simulation, programming languages are separated into two categories: **general** purpose and **simulation** specific. The former are designed to be capable of accomplishing any computer-related task, while the latter are customized specifically for computer simulation. Each language's level of abstraction is also listed:

Language	***Category***	***Abstraction Level***	***Performance (versus C)***	***OSS?***	***Latest Update***
C	General	Low	1.0 , 1.0 , 1.0 , 1.0	Yes	2011
C++	General	Medium	1.0 , 1.0 , 1.4 , 1.5	Yes	2011
Java	General	High	2.0 , 2.7 , 2.3 , 3.3	Yes	2012
JavaScript	General	High	4.2 , 4.1 , N/A , N/A	Yes	2011
Python	General	High	43.9 , 38.6 , 50.6 , 54.3	Yes	2012
Modelica	Simulation	High	(*no data*)	Yes	2012
SimScript	Simulation	High	(*no data*)	No	2007
Simula	Simulation	High	(*no data*)	Yes	1984
Simulink	Simulation	Very High	(*no data*)	No	2012
VisSim	Simulation	Very High	(*no data*)	No	2011

TABLE 4.5: Comparison of potential programming languages. General and Simulation languages were considered, based on their performance, popularity, level of abstraction, availability, and pertinence.

low denotes languages which require the user to manage memory and carefully structure their code, while **high** implies automatic memory management and abundant techniques for encapsulation. Some simulation languages abstract at a **very high** level, allowing the modeling of an entire system using block diagrams and flowcharts. The remaining three categories are more straightforward. The performance of each general-purpose language is estimated[5] on four machines. The first two of these (from left to right) are single-core (32-bit, then 64-bit kernels), while the latter-two are quad-core. Next, the **OSS** column lists whether or not a viable version of the software exists as *Open Source Software*. Finally, the date of the most recent update to each language's standard is shown.

After careful consideration, the Java programming language was chosen to realize our reference implementation. In order of importance, the motivating factors behind this decision are as follows. First, a high or very high abstraction level was deemed necessary, as it eases the task of verification by handling lower-level details automatically. Of the general-purpose languages with a high level of abstraction, Java featured the best performance. JavaScript is close, but was considered lacking in the breadth of its standard library, as well as being slightly behind in terms of standardization. A continually updated standard was considered necessary, as it allowed the language to capture new programming techniques that presumably enable more robust design patterns. Compared to simulation languages, Java had the benefits of an Open Source implementation as well as a recent standard update. This left Modelica and Java as the best options representing simulation and general languages respectively. The final

[5]Performance statistics are based on measurements by [145] across 13 benchmarks, scaled relative to the time taken by the fastest implementation. A value of 2.0, for instance, indicates that a given implementation takes twice as long as the fastest implementation.

deciding factor was the accessibility of the Java programming language as evinced by the size of its community on [146]. A wide audience is of the utmost importance to a reference implementation, as it maximizes community feedback to the betterment of the nascent framework. Anecdotally, one might consider the positive effect that a large target audience had on the development of the *Repast* simulation environment, which also enticed new users by being written in Java [88].

Given the decision to use Java, the remaining choices narrowed in scope substantially. As Java had already been chosen as the language of the simulation component, using a different language for the visualization component would complicate interoperability efforts. We were thus restricted to the various Java visualization toolkits. Of these, LWUIT, AWT, LCDUI, and Thinlet were either outdated or developed specifically for restricted devices. Several others, including Java-Gnome, wx4j, and Qt Jambi, were merely wrappers around other toolkits such as Qt or Gtk. As mentioned in Section 4.3.1.1, relying on a single-toolkit wrapper comes with its own host of problems. The remaining two options were Swing, from Sun, and SWT, from IBM. Swing renders all GUI components using Java code, while SWT wraps native components whenever possible. Swing is toolkit-agnostic, while SWT has support for multiple backends. Both support the majority of architectures and operating systems currently available, and neither is a clear winner in terms of performance or memory usage. As both toolkits were essentially equal, the decision between them became arbitrary and SWT was chosen as the visualization toolkit for the GUI component.

The perennial framework also required a strategy for passing data between various sensors, effectors, and additional components. This became particularly pertinent when user-specified types were considered, since each user of the implementation may have varying data requirements. Several possible type management strategies were considered:

- **Untyped** — Use the most generic type provided by the language, such as Java's **Object** class or C++'s **void*** type. This allows full flexibility without providing any type guarantees.
- **Loosely Typed** — Provide a common subclass for each distinct data-type, but require type-casting to extract the expected type. This provides full flexibility with minimal type guarantees.
- **Strongly Typed** — Provide common subclasses for distinct data-types, and perform all data interaction through methods. This is less flexible, but provides strong typing guarantees and makes full use of the object-oriented programming paradigm.

- **Fixed Types** — Provide common subclasses and a set of expected types, but do not allow user-level subclassing. This enables a more streamlined framework design (since all types are known), but does not allow user-defined types.

The first and last approaches each exhibited a strong weakness that precluded them from consideration. Without type information, a sensor risked being mistaken for an effector, and a simulation risked being confused with a model. Without the possibility for user-defined types, the implementation would needlessly restrict its users and represent the framework poorly. From the remaining two options, loose typing was chosen for its flexibility and ease of implementation. Strong typing would have required needless diligence, as user-defined types tend to be aware of each other. For example, a custom sensor type was probably created for use in a custom simulation, so expecting the simulation to cast to that type is not unreasonable.

4.3.2 Class Diagram

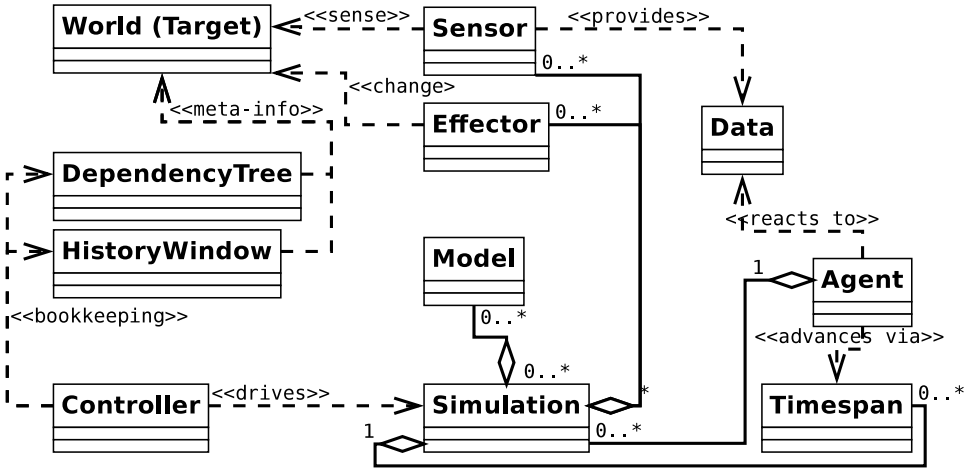

FIGURE 4.3: Concise UML class diagram of a perennial framework implementation.

Figure 4.3 shows the main class interactions of our perennial framework implementation using a reduced Unified Modeling Language (UML) class diagram [147]. This diagram is a concise representation of the components in the software library; more detailed diagrams follow in subsequent sections. Finally, a complete UML diagram which consolidates all of these details and more is catalogued in Appendix B.

As can be seen from Figure 4.3, the implementation of our perennial framework is controller-centric with respect to sensors, effectors, models, and worlds. It is agent-centric with respect to simulations, agents, and their data. We will now describe each element of the implementation in greater detail.

4.3.2.1 World and Target

As shown in Figure 4.4, a **world** maps closely to its theoretical representation. Worlds are constructed with a name, description, and location, similar to their breakdown in Table 4.2. A controller is also passed in during construction, ensuring that worlds have access to controller-provided functionality at all times. Worlds maintain their own sensor coverage mappings, which can be broadened with the function *expandHistoryWindow*. This allows the controller to set up historical simulations by simply polling each world for the list of valid sensor timespans.

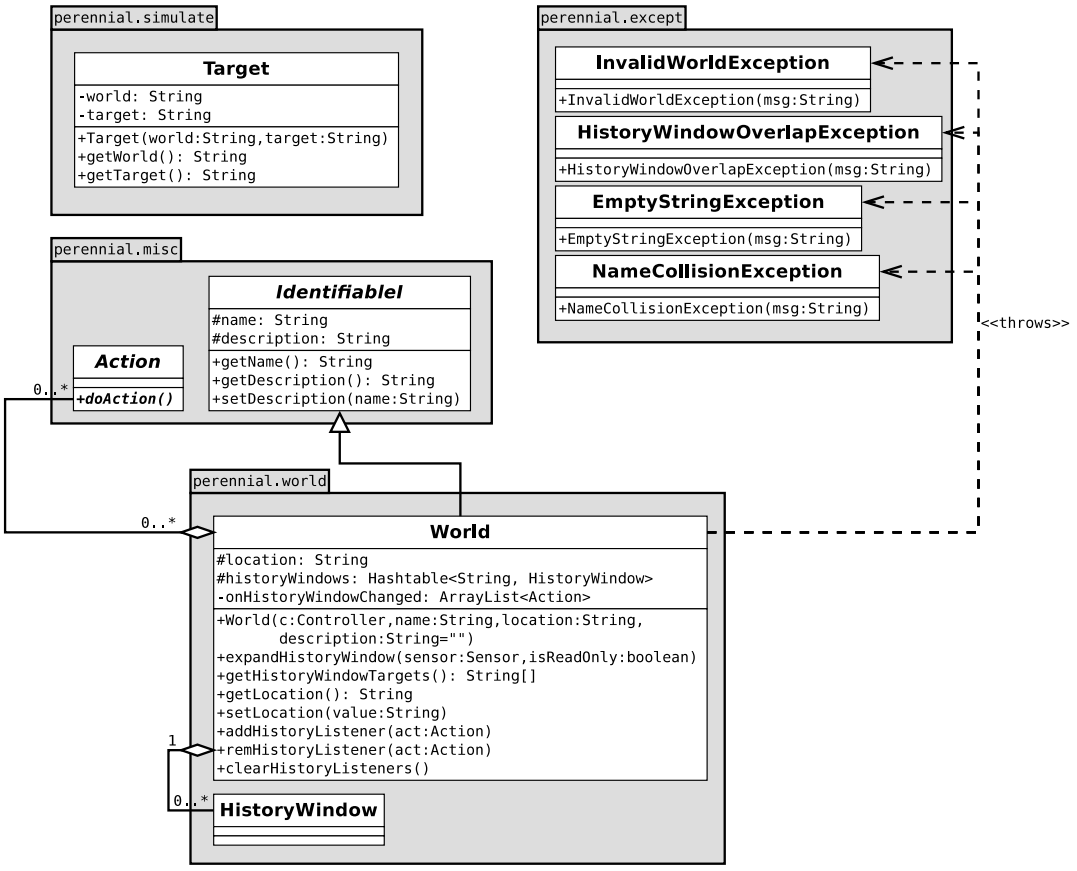

FIGURE 4.4: UML Class Diagram for Worlds and Targets

In addition to the necessary components of a world, Figure 4.4 also depicts the internals of a **target** and various **exceptions** which a world may inadvertently generate. The target is part of the sensor specification, and is used extensively throughout the simulation. In addition to the target identification string, a target also includes a reference to the world in which it is located. This is not required by the framework, but proves useful in that it allows targets to be passed back and forth without losing the context in which they are valid. Various exceptional behavior is captured in the ***perennial.except***

namespace. All exceptions are named after the constraint they enforce; e.g., an **EmptyStringException** occurs when a String value is required to be non-empty. This is the case, for example, with the "world" string passed into a target's constructor.

Finally, the **action** class is used to exchange notifications among a variety of simulation elements. Actions occur when the global setup of the perennial study is modified. For example, an action will occur if a sensor is added to or removed from the sensescape. Actions are not part of the original framework specification, but are a side-effect of the choice to use Java for the GUI as well as the simulation component. By relying on actions to convey relevant changes, we approximate a popular software methodology known as **Model-View-Controller**, and leverage that technique to reduce programming time.

4.3.2.2 Sensor, Effector, and Data

Sensors, effectors, and the data they process are all detailed in Figure 4.5. **Sensors** and **effectors** are constructed with roughly the same set of properties defined in Table 4.3. In addition, several actions are defined to aid integration with the GUI. Up-to-date access to real-time data is provided by a **continuous sensor**, which is another extension specific to this implementation. Traditional sensors and effectors use **timespans** to specify their current and historically valid ranges, while continuous sensors use a singleton object of type **continuous timespan**. Invalid timespan configurations (i.e., those which end before they start) will raise **LogicalExceptions**.

Sensors and effectors both operate on instances of the **data** class, which contains a set of **datum** objects sorted in order of start time. Each individual datum contains "something", which is represented by Java's generic **Object** class. The **data** class itself provides a weak type guarantee as per the weak type requirement introduced in Section 4.3.1.2; beyond that, it is the responsibility of the given sensor, effector, or simulation to downcast the datum into an appropriate class and react to it.

Individual studies performed with this implementation are free to extend Sensor and Effector with needed functionality. To improve usability, two additional classes were added that are not explicitly part of the original framework. The first sensor type is the **SocketSensor** (and, analogously, **SocketEffector**), which reads data from a TCP socket and provides it to the simulation when requested. This allows sensor interaction code to be written in a more systems-friendly language such as C++ and then simply published to a pre-specified socket. Most major languages support socket programming, so this technique greatly widens the breadth of available sensors with which our implementation can interact. The second sensor type is the **DistributionSensor**, which wraps common distributions such as *uniform*, *normal*, *poisson*, or *exponential*. This

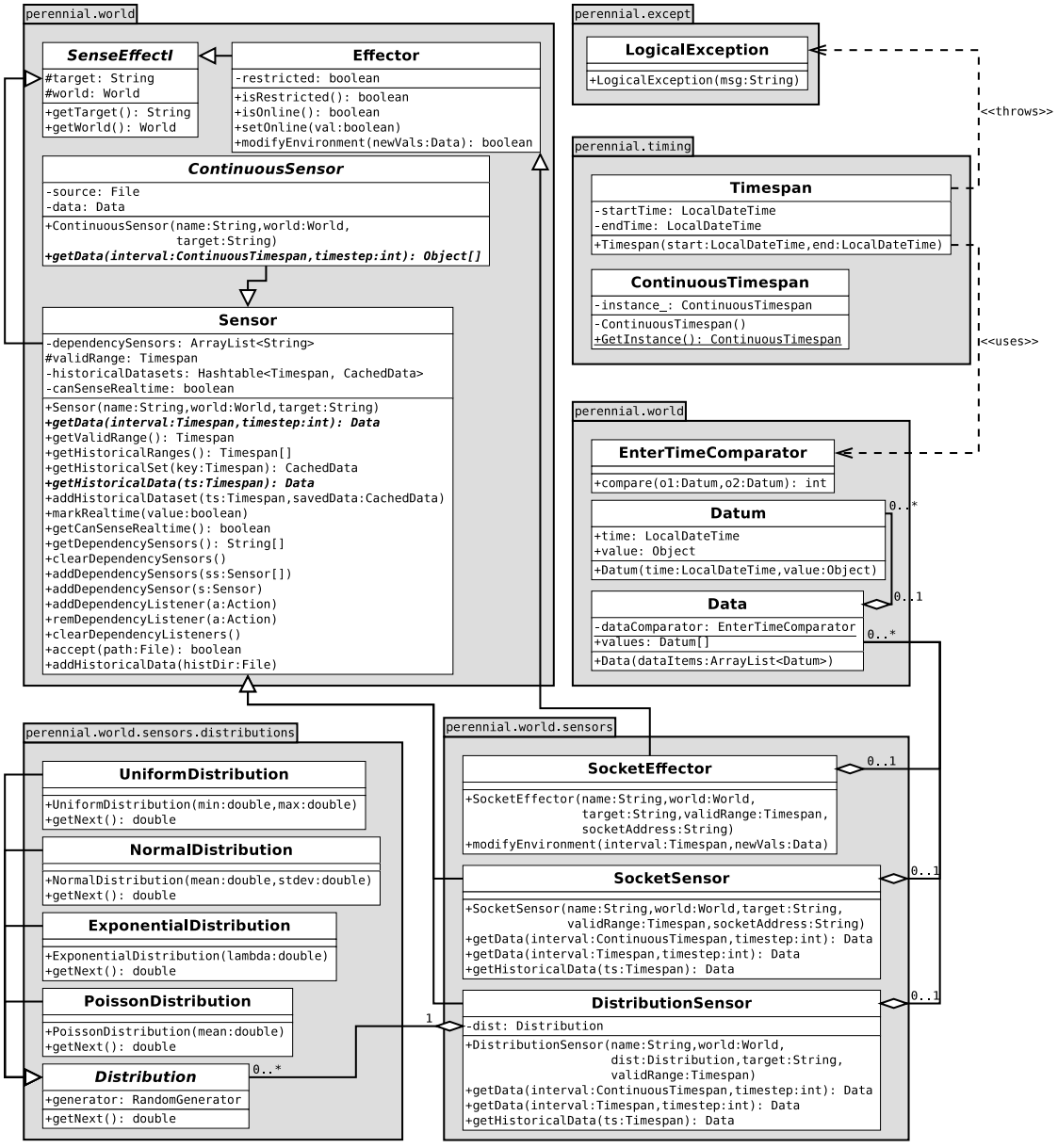

FIGURE 4.5: UML Class Diagram for Sensors, Effectors, and Data

class allows the most common distributions to be represented as sensors, reducing the amount of boilerplate code that must be written to instantiate a new simulation study.

4.3.2.3 Dependency Tree and History Window

Sensors, effectors, and worlds rely on two key components: the **Dependency Tree** and the concept of a **History Window**. The dependency tree is meant to resolve an implementation detail regarding interacting sensors. Consider three sensors, A, B, and C. Sensor A is configured (via its *mapping-function*) to merge data from sensors B and C. Sensor B, meanwhile, presents a higher-level view of the raw data in Sensor

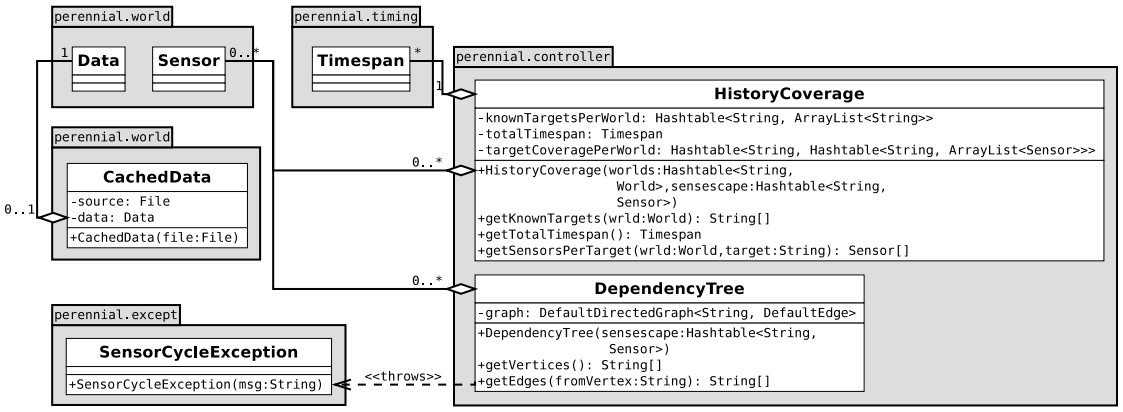

FIGURE 4.6: UML Class Diagram for the Dependency Tree and History Window

C. At each time tick, these sensors must be updated in an explicit order, namely: C first, then B, then A. Any other update order will lead to consistency errors. Thus, the dependency tree is aware of all causal links between sensors, allowing sensor updates to be read and computed from the leaves of the tree upwards. Independent clusters of sensors are collected via a node called *DONE* which is the root of the dependency tree; thus, there is only ever a single dependency tree per simulation experiment. As one might expect, cycles in the dependency tree are an error; if discovered, they trigger a **SensorCycleException**.

The history windows solves another implementation issue, that of access to historical data. This class coordinates all available sensors, providing a temporal mapping of all times at which Sensors were active and recording data in the past. To simplify matters, the local filesystem is used to store historical data, rather than a fully-functional relational database as recommended in Figure 4.2. The **Cached Data** class encapsulates this, loading the data objects on demand, so migrating to a database-backed variant of the current implementation would be a straightforward procedure.

4.3.2.4 Controller

The Controller functions as a highly centralized entity which manages and connects worlds, sensors, simulations, and other lesser components. To this end, it contains a complete mapping of all components present and active in the perennial system. Functions such as *addWorld*, *getWorldNames*, and *getWorld* provide complete, encapsulated access to the global set of components. The controller is responsible for firing off events such as those subscribed to in a world's *addHistoryListener* function. In addition, the controller also manages its own set of events, such as the *addSensorChangeListener*,

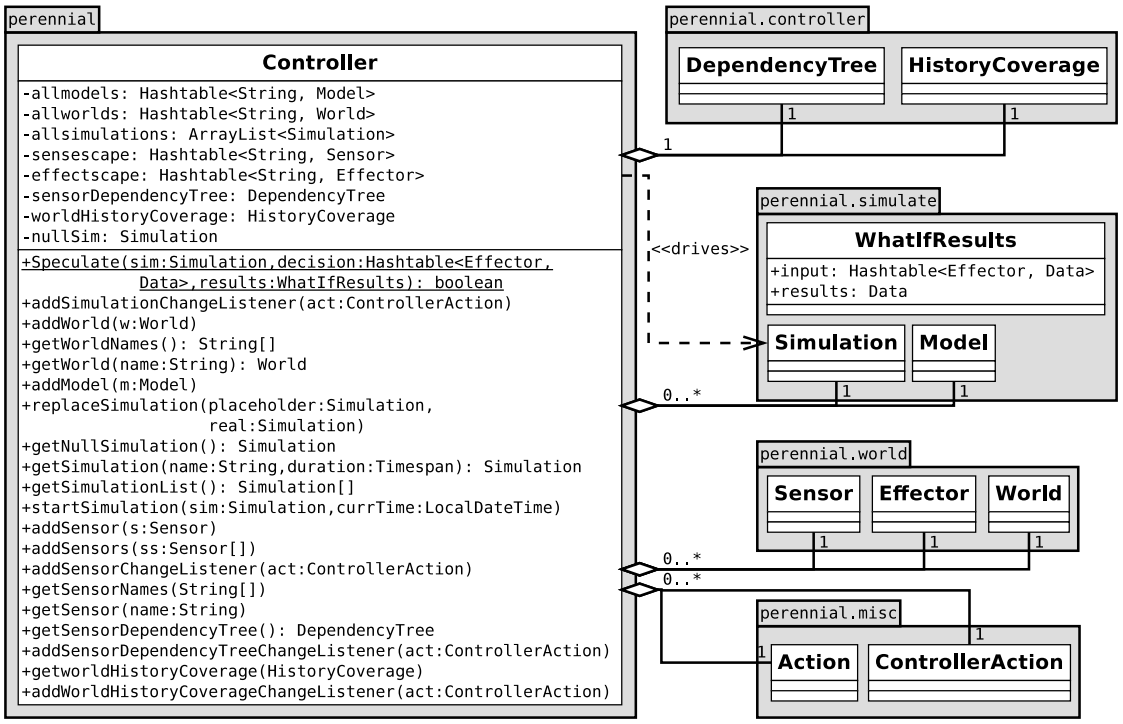

FIGURE 4.7: UML Class Diagram for the Controller

which fires whenever a sensor is added or removed. Many of these listeners are intended to ease GUI integration, but some may have uses within a running simulation as well.

The function *Speculate* is a class-scoped, recursive function which is intended to help with the generation and management "What-If?" simulations. *Speculate* takes a mapping of data items to effectors, which it uses as the basis of the first "What-If?" analysis. The simulation passed in to *Speculate* is run, and the results are saved as an object of type *WhatIfResults*. The function returns *true* if the stopping condition has been reached; otherwise, it may decide to prune the result set or mutate the set of effectors and thus explore a new plan. This continues until a satisfactory answer has been obtained. Section 4.3.3.1 expands on this in more detail, offering both a utility-based metric and stopping condition as well as a Pareto-front approach.

The controller is somewhat complicated internally, but its external interface is unsurprising and concise. The only mildly complicated element introduced by the controller is the **null simulation**. This is a simple construct that allows modelers to specify "no simulation"; it can be used to set up arbitrary sensor templates. The null simulation represents another implementation-defined choice that embellishes the pure theoretical framework.

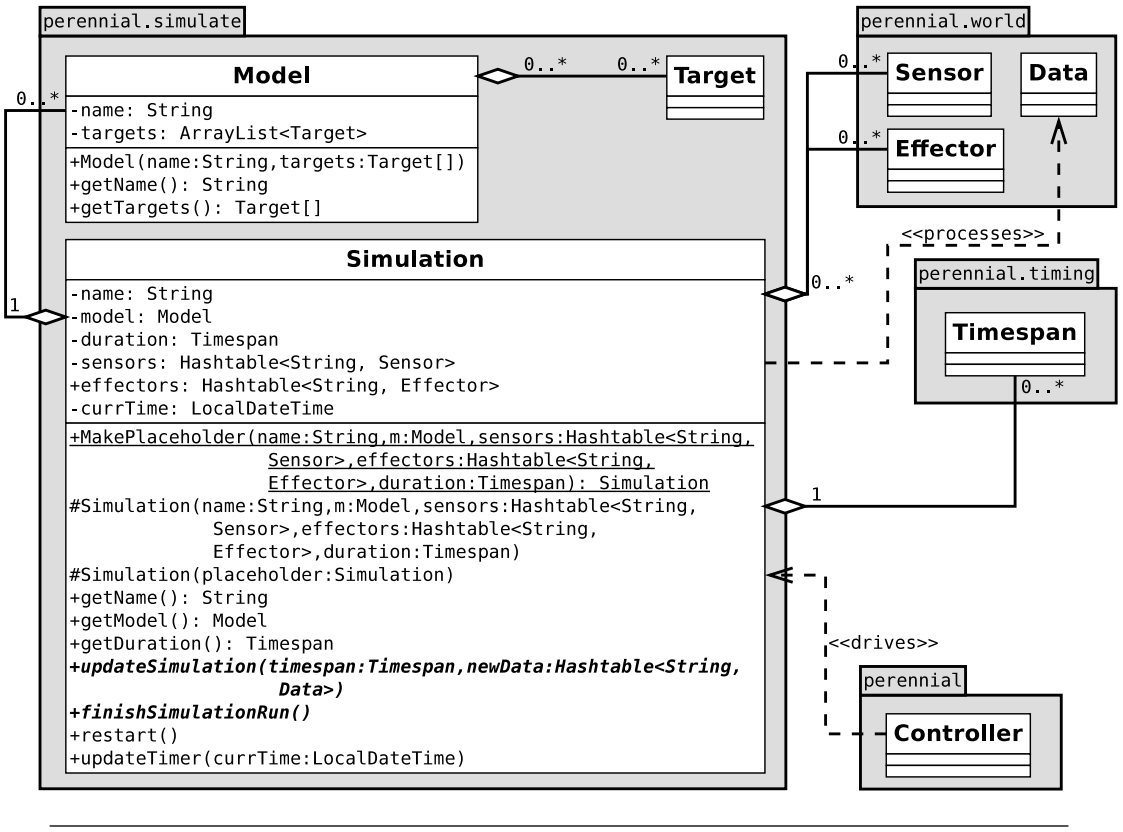

FIGURE 4.8: UML Class Diagram for Models and Simulations

4.3.2.5 Model, Simulation

The contents of models and simulations, shown in Figure 4.8, vary little from their theoretical specification in Section 4.2. A **model** has a name and a list of targets, neither of which can change once it has been constructed. Likewise, a **simulation** wraps a model and represents that model being exercised over the course of a given timespan. Multiple models may be managed by a single simulation through subclassing; the single model listed implies that a simulation must manage *at least* one model. Simulations also contain a mapping of sensors to targets, which allows the same simulation to be reconfigured and reused in various similar studies. Finally, simulations keep track of the last point in (simulation) time that they were updated. This is a dynamic property; it only has meaning for a running simulation.

Models and simulations are loosely-specified components; it is intended that the entirety of their functionality be realized through subclassing. Simulation sub-classes must implement two functions: *updateSimulation* and *finishSimulationRun*. The first of these gives the running simulation a timespan and a list of data mapped to each sensor. In return, the simulation is expected to process all data entities and update its internal state. The second function merely tells the simulation to complete any pending tasks

once the simulation run has ended. The **simulation** base class will automatically call these function as the **controller** calls the *updateTimer* function, which performs some minor additional logic for continuous timespans. In terms of functionality, the simulation is considered more visible than the models. Since models are driven by simulations, the model class need export no additional functionality. If a model requires additional external functionality (i.e., an *updateModel* function) then it may accomplish this in a subclass, and a corresponding simulation subclass must respect these requirements.

Simulations contain additional protected constructor functions, as well as the ***MakePlaceholder*** static function which constructs a pseudo-abstract object given a set of properties describing a sensor. These functions are a relic of the initial desire to specify a simulation as a detached entity, and then instantiate it once it starts running. (This is also the reason that the simulation class contains no *initSimulationRun* function; the simulation object is re-created for each new simulation run.) In retrospect, an argument can be made that this method of managing simulation lifecycles is somewhat unwieldy, and could probably be handled more simply by virtualizing the *restart* function and just relying on that. That said, the current approach to simulation creation has no flaws beyond a minor increase in complexity, so refactoring it was not deemed critical to the health of the framework as a whole.

4.3.2.6 Agent

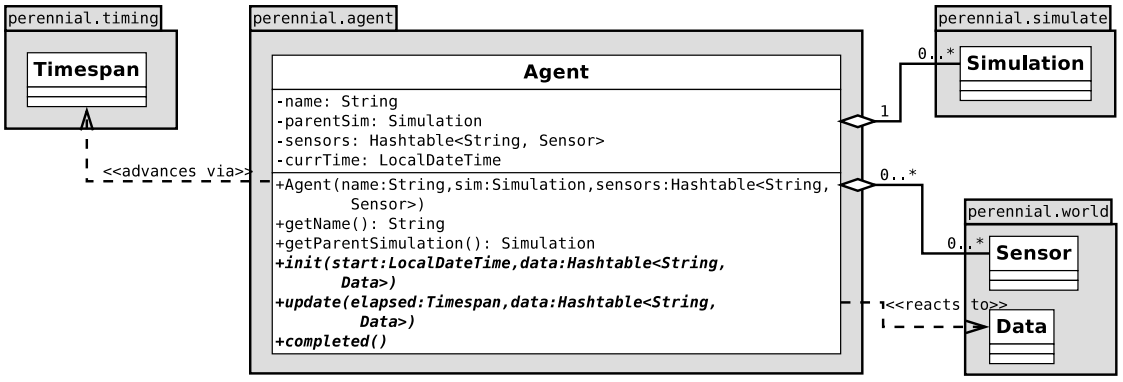

FIGURE 4.9: UML Class Diagram for Agents

An **agent** is an active, self-contained entity within a single simulation. Like simulations, agents achieve their breadth of diversity through subclassing. Each agent maintains a list of sensors, although this is primarily for bookkeeping purposes. Agents do not read data directly from sensors, but rather receive data objects within the *update* function which is called by the parent simulation of each agent. (If greater parallelism is required, agents may negotiate with the simulation and read directly from the sensescape, but this

is the exception rather than the rule.) The *init* function is called once whenever the agent is restarted in a given simulation environment, and may include initial sensor readings. The *completed* function is called either when the simulation is done, or when the agent has completed all of its tasks within the simulation. Regardless of its source, this function instructs the agent to return any borrowed resources and quietly exit.

As stated above, each agent distinguishes itself from its peers through object inheritance and polymorphism. There are other options; for example, an agent may receive **events** that tell it the simulation has started or is updating. These events would then be processed by an **event handler**, and agents would distinguish themselves via the combination of event handlers registered to respond to each event type. Although equally valid, we chose not to use an event subscription model, as it is somewhat less cognitively cohesive than inheritance. For all its verbosity, a subclass of agent called "HelbingSocialForcesPedestrian" very clearly self-documents its intended functionality [148]. In addition, relying on an object-oriented technique (inheritance) for agents allowed us to continue using the same abstraction for this component of the simulation, thus keeping the overall design simpler.

4.3.2.7 Remaining Simulation Components

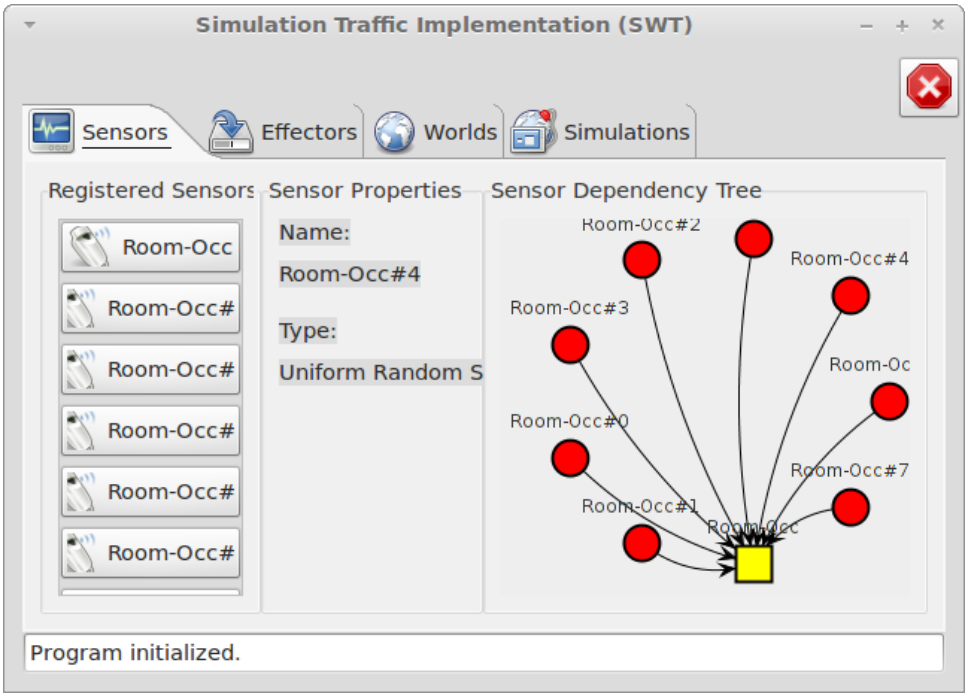

FIGURE 4.10: Screenshot of the GUI component, showing a simple sensor dependency tree.

All of the components in Figure 4.2 are present in the framework implementation to some degree. Some have been explicitly modeled in the previous series of UML diagrams. Others, such as the *converter/agglomerator* or the *access layer* are present implicitly in the software. The GUI component of the implementation, shown in Figure 4.10, was considered non-integral to the framework as a whole, and its UML diagram has not been presented. Please contact the author if you are interested in the interaction between the GUI and the simulation component.

4.3.3 Local Implementation Decisions

The entire implementation as presented is self-contained and functional. All globally-relevant decisions have been presented, and the UML class diagrams shown have reflected these decisions. That said, there are several locally-relevant decisions which can ease the development of any perennial simulation. These will be covered briefly here, and can be considered antecedent to the development of an implementation in any given language.

4.3.3.1 Measuring the Extent of a Crisis

In order to properly simulate the management of an arbitrary crisis, there must be some way to measure the severity of that crisis. This is even more important for symbiotic simulation, which depends on an accurate measurement of performance to auto-validate selected strategies from its "What-If?" simulations. Unfortunately, historically symbiotic simulation has focussed on domains with a very clearly defined or limited role for human decision-making, such as manufacturing and shipping. Where humans are given "full freedom" to act in the system, their available interactions are often narrowly defined. For example, work by Low *et al* uses sophisticated ontologies and behavior models to represent humans, yet limits their actions to simple goal-directed movement [44].

An accurate performance metric is necessary from a simulation standpoint alone. However, the essence of the problem is even more nuanced, since the nature of this metric differs slightly depending on whether one is running a *hindsight* or *foresight* study versus *symbiotic decision support*. Regarding the former, the full breadth of available solutions should be probed, and those that have any deserving qualities should be presented and categorized by those qualities. In this environment, only truly bad mitigation strategies should be discarded. Conversely, when one is running "What-If?" simulations during a crisis, the need for guidance outweighs the need for precision[6]. Recall that, for crisis

[6]Note that, while *precision* wanes in importance, *accuracy* is nonetheless critical

management, making a decision quickly is often better than waiting for additional data at the risk of making no decision [24]. One final problem remains for all three study categories: the fact that, even in a field such as crisis management, different situations and different cultures prioritize performance parameters differently [149]. With this in mind, an approach is needed that deals with all of these problems and is adaptable with respect to the different ways of delineating a crisis.

As a solution to the local problem of measuring crisis performance, we propose using a utility function for *decision support* studies, and estimating a Pareto front for *hindsight* and *foresight* studies. Both will rely on the same partial performance metrics, such as loss-of-life and property damage. Several additional items, such as "technology", "resources", "infrastructure", and "culture", were identified by [25] as broadly significant categories in emergency response situations. Expanding on this, we arrive at our partial performance metrics for crises:

- **Loss of Life**
  - *Number of Human Lives Lost* (N_H) — Total number of human lives lost during a simulation run.
- **Loss of Land**
  - *Arable Land Lost* (N_{LA}) — Total land lost (km^2) which was suitable for farming and agriculture.
  - *Habitable Land Lost* (N_{LH}) — Total land lost (km^2) which was suitable for humans to live on.
  - *Traversable Land Lost* (N_{LT}) — Total land lost (km^2) which allowed for human movement. Generally excludes deserts, dense jungles, etc.
- **Loss of Necessities and Public Services**
  - *Loss of Power* (N_{BP}) — Maximum electricity (kWh/day) lost on any one day during the crisis.
  - *Prolonged Power Loss* (N_{BPp}) — Average electricity (kWh/day) lost during the crisis.
  - *Loss of Potable Water* (N_{BW}) — Maximum amount of water (L^3/day) lost on any one day during the crisis.
  - *Public Transport Loss* (N_{BT}) — Average down-time ($hours/day$) of buses and trains.
- **Environmental Loss**

  - *Animal Deaths* (N_{EA}) — Total biomass (kg) of all animals that died.
  - *Loss of Biodiversity* (N_{EB}) — Total number of species that lost sufficient biodiversity to drop below replacement levels.
  - *Habitat Loss* (N_{EH}) — Total land (km^2) which lost all ability to harbor natural life.
  - *Increase in Greenhouse Gases* (N_{EC}) — Total weight (kg) of increased greenhouse gases caused by the crisis.

- **Financial Loss**
  - *Loss of Home* (N_{FH}) — Total number of homeless after the crisis.
  - *Loss of Property* (N_{FP}) — Total damage done to homes, in the local currency unit.
  - *Loss of Business* (N_{FB}) — Total days of business lost, estimated from the number of businesses forced to suspend operation during the crisis, and the expected economic output of the area under normal circumstances.
  - *Medical Expense* (N_{FM}) — Total medical expenses incurred due to the crisis.

This list captures all of the main damaging factors of a crisis. We assume that a reasonable range can be defined for each partial performance metric, and use an overline to represent a normalized version of each parameter. Thus, $\overline{N}_{FM}$ represents the normalized medical expenses of a crisis. Both Pareto analysis and our utility function are capable of handling additional performance metrics (the latter by means of normalized values and scaling factors). For example, a school board might devise a new metric for "days of school closure", or might incorporate this amount into the N_{FB} sum. Similarly, a corporate executive might try to extend our model to cover business crises by adding a category for "PR-related losses". Certainly, these are reasonable decisions. However, the former is an example of a category that is too specific for our generalized framework, while the latter was explicitly excluded from our work in Chapter 3.

For *decision support* studies, each factor X is multiplied by a *weighting factor* W_X and combined to form a **utility function** which can be used as an overall grade of the crisis, as shown in Equation 4.12. Each weighting factor can range from 0.0 to 1.0, with 0.0 implying that a given factor has no relevance to the study at hand. Two of the factors do not follow this rule: W_H and W_{HS}. The weighting factor for human life, W_H, is held constant at 1.0. An additional factor, W_{HS} is called the *human scaling factor*. This must be at least 1.0, and has no upper bound. The human scaling factor is useful to prioritize human life heavily over a combination of other parameters. The decision was made to introduce W_{HS} —instead of artificially raising W_H above 1.0— because it

serves a different ontological purpose than all the other weighting factors. Even when set to its lowest possible value, it still ensures that human life is placed above any other *single* factor. In addition, the value of W_{HS} tends to be based on considerations, such as job risk, which do not themselves change with any rapidity. Thus, most applications can simply decide on a permanent value for W_{HS} early into the study, and then proceed to change only the other weighting factors.

$$\begin{aligned} P = {} & W_{LA}(\overline{N}_{LA}) + W_{LH}(\overline{N}_{LH}) + W_{LT}(\overline{N}_{LT}) + W_{BP}(\overline{N}_{BP}) + W_{BPp}(\overline{N}_{BPp}) \\ & + W_{BW}(\overline{N}_{BW}) + W_{BT}(\overline{N}_{BT}) + W_{EA}(\overline{N}_{EA}) + W_{EB}(\overline{N}_{EB}) + W_{EH}(\overline{N}_{EH}) \\ & + W_{EC}(\overline{N}_{EC}) + W_{FH}(\overline{N}_{FH}) + W_{FP}(\overline{N}_{FP}) + W_{FB}(\overline{N}_{FB}) + W_{FM}(\overline{N}_{FM}) \\ & + W_H \cdot W_{HS}(\overline{N}_H) \end{aligned} \tag{4.12}$$

For *hindsight* and *foresight* studies, the list of performance factors must be explored through "What-If?" simulation to produce an estimation of the **Pareto front** for that scenario. The Pareto front —sometimes called a *Pareto set* or *Pareto frontier*— is a set of choices for which no other point strictly dominates an existing choice. This concept, known as *Pareto efficiency*, implies that performance can not be increased for one parameter without a consequent reduction in another. Figure 4.11 illustrates a Pareto front estimation for habitat loss (N_{EH}) on the x-axis and property loss (N_{FP}) on the y-axis. The diamonds represent data points which are non-dominated and thus contribute to the Pareto front, while circles represent points which are strictly dominated and thus do not contribute. The estimated Pareto front is drawn between all points in the Pareto set. The concept is similar in three or more dimensions; the only challenges are an increase in the time required to compute the Pareto front and an increased difficulty visualizing the results.

Computing a Pareto front takes a large amount of time, and usually requires the use of some form of converging algorithm such as a genetic algorithm. For the purpose of discussion, the PISA framework is a fully-functional Pareto estimation package that meets our needs [150]. Once computed, a Pareto front's value lies in its ability to fully capture the tradeoffs between various decisions. For example, in the case of Figure 4.11, one might wish to choose a solution which maximizes N_{EH} at the expense of N_{FP}, or one might wish to ensure that both remain above a given threshold. Finally, as observed in [151], in certain cases a utility function can be used to approximate a Pareto front, but it will only capture points near the extremes of each parameter (the far left and far right diamonds in Figure 4.11). This justifies our decision to use a utility function for

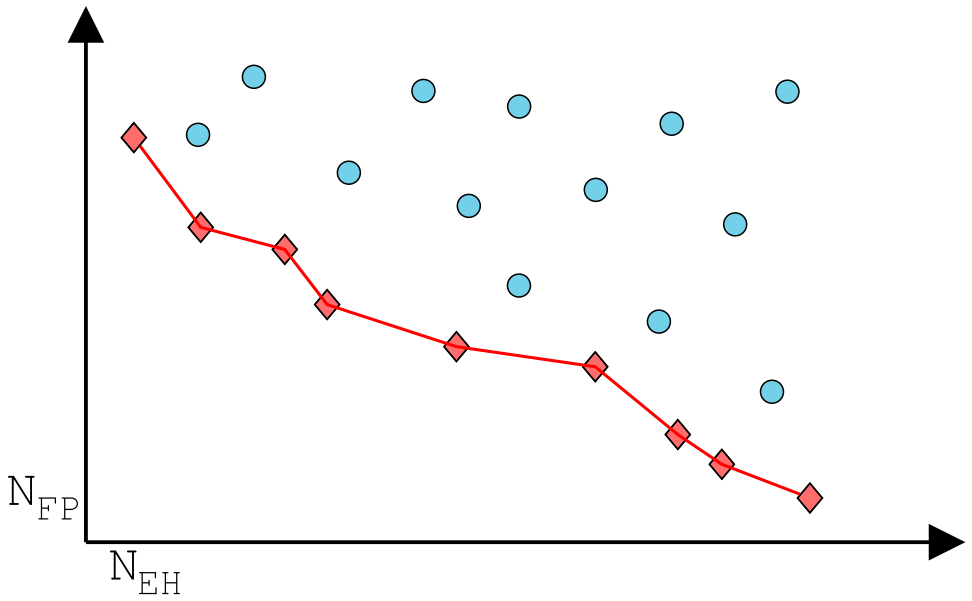

FIGURE 4.11: An example of a Pareto front estimation on the minimum values of N_{EH} and N_{FP}.

decision support studies; it can be used to quickly explore portions of the Pareto set while still allowing administrators to prioritize different partial metrics.

The sum of weighted partial metrics and the appeal to Pareto analysis together provide the necessary level of functionality required by *hindsight*, *foresight*, and *symbiotic decision support* studies. First, both techniques allow policy makers to identify and mark only those parameters which matter to a given crisis scenario. Second, the Pareto front method can be used to generate various graphs that capture the interactions of the numerous partial metrics. This allows researchers to ask informed questions such as "How much more environmental damage could have been prevented without drastically increasing the number made homeless?". Third, the utility function allows calculation of the most important crisis-related factors quickly and with less precision than (but similar accuracy to) a Pareto analysis. Finally, a Pareto front may be approximated after the fact, to determine if the solution captured by the utility function during the crisis actually performed as well as expected. This is a normal consequence of *hindsight* studies, and is one of the benefits of using a combined perennial system.

4.3.3.2 MMOHILS Considerations and Incentives

MMOHILS is a powerful technique that captures data using real humans participating in simulations of virtual environments. Motivating users quickly becomes challenging

with virtual worlds, especially when users are physically removed from simulation coordinators. The temptation to view the simulation as a game and try to "win" it, cheat, or just playfully misbehave increases, since most existing virtual worlds are currently presented as online games. In [135], we proposed borrowing *incentives* from the field of microeconomics for the purpose of encouraging good behavior. We also recommended splitting MMOHILS into two categories, **experimental** and **unannounced**, as a way of diversifying the MMOHILS ecosystem and allowing better targeting of incentives. Both types operate on real users in virtual environments; the difference is in their degree of trust and participation, and the capability to coordinate and monitor them. In addition, *experimental* MMOHILS was designed to be simple and inexpensive to implement, while its *unannounced* counterpart aims to achieve a maximum number of connected users regardless of the cost.

Experimental MMOHILS is suitable for small groups comprising less than 100 participants, all of which are knowingly participating, with good intention, in an experiment. Users are connected through a local area network (LAN), which features round-trip network delays of 1 to 2 ms [152]. They will likely be located at workstations in the same computer lab, and can therefore be monitored to reduce the chance of misbehavior. In addition, social pressures can more easily be applied to reduce cheating. From a software design point of view, discrete, time-stepped simulation is the best choice, as it co-operates nicely with a technique used in online gaming called the *lock-step protocol* which waits for updates from all users before processing each time step. Since users are trusted, much of the processing can be shifted from the server to the client programs themselves. This reduces the processing requirements of the server, helping to make MMOHILS more affordable on commodity hardware. Due to the small number of trusted users, cash handouts are acceptable as a motivating factor.

Unannounced MMOHILS was designed to leverage the full power of massive numbers of users, and proceeds without them knowing the full extent of the study. Ideally, the simulation would be concealed within an existing, popular online game as a means to increase its potential user base. As such, users are connected through the Internet, with round-trip delays of 150 to 300 ms, thus obviating full-locking techniques, which introduce too much unnecessary latency. Client updates must be processed when received by the server, and then immediately broadcast to all users in the virtual vicinity. Unlike traditional online games, a MMOHILS must ensure that all users see the same world state at all times, within the degree of precision required. A policy must be maintained for ejecting users that lag behind in network speed. Moreover, techniques such as interest management must be employed to handle the massive increase in world space and user base that massively multiplayer online games cater to. The software side of an unannounced MMOHILS will likely be much more complicated than that of

the experimental variant. In addition, unannounced MMOHILS deals with users who receive less trust. This reflects on the server, which cannot offload critical tasks to the client programs and thus requires more powerful hardware. Extra care is also required in the verification step, as the client program will be running on an unknown variety of hardware configurations.

Monitoring of clients in the unannounced setup is infeasible for a number of reasons, and the general goodwill of players cannot be assumed. Assuming that client programs can be secured, the validity of each user's *role* and *psycho-social state* must be assessed at all times during the simulation. Methods to accomplish this are discussed in the next section. Finally, incentive management takes on a much higher level of importance in unannounced MMOHILS, as cash handouts become prohibitively expensive. It is vital that as many participants as possible finish each experiment, and equally important that they remain immersed lest they become bored and leave the simulation prematurely. We propose combining an incentives system with regularly scheduled *events* to overcome this problem. Daily events reward all players who complete the event with in-game items, and give particularly special rewards to those players who "finish first" in each event. At all non-event times, the online world is running normally as a game, and the items won during events prove to be rather useful. We conclude with a paragraph, from the original MMOHILS paper, describing a sample MMOHILS incentives setup.

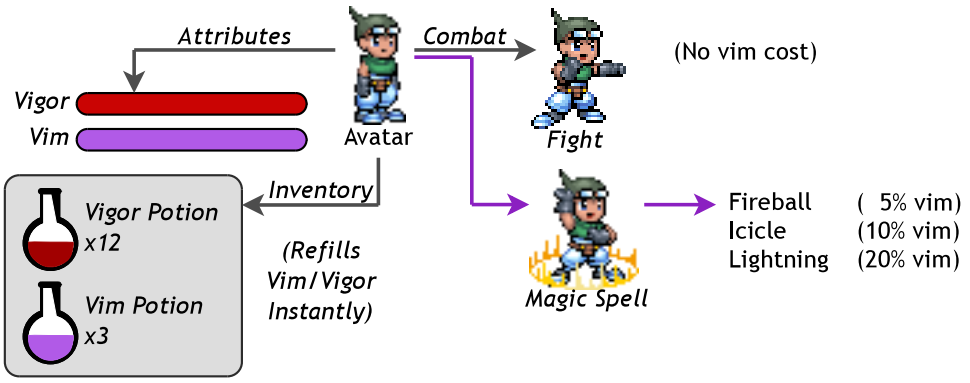

FIGURE 4.12: A sketch of an online system designed to create a strong incentive to complete the simulated egress quickly.

Suppose that players can explore the virtual world, fight each other, or cast magical spells. Fighting reduces an opponent's *vigor*, and a battle is lost when one player's *vigor* reaches zero. Spells reduce the caster's *vim*, and perform a variety of effects. Both *vim* and *vigor* replenish gradually over time, and they can be restored to 100% immediately by consuming *vim potions* and *vigor potions* respectively (see: Figure 4.12). This setup induces a huge demand for *vim* and *vigor* potions, as they allow players to immediately

> refresh their avatars. Every day at noon, all players are transported to a new location and informed that they will be rewarded with 1 of each potion type if they can reach a target destination —10 of each if they're one of the first five players to arrive.
>
> This example event is an egress simulation in disguise. The event's location is a virtualized floor plan of a building, and the the goal locations are exits. Rewards enforce the desire to exit the "building" as quickly as possible [135].

4.3.3.3 Validation Techniques for MMOHILS

Given their reliance on human participants, MMOHILS present some unique challenges in terms of simulation validity. These are compounded by the use of incentives in particular and virtual environments in general, so a thorough treatment of the subject is pertinent. Since the majority of validation tasks performed for MMOHILS are the same as those for any other simulation, a brief introduction to the process of validation will be given. This topic is covered extensively by Law [15] and Sargent [131], and a succinct overview is provided in [59]. While Law focuses on the practical side of simulation, Sargent focuses on statistics-based approaches and provides a full overview of all available techniques. Both authors' work is considered definitive and time-tested as a means of validating simulation studies.

A model is said to be valid if it is "an accurate representation of the system *for the particular objectives of the study*" [59]. Validation should be performed as the simulation is being developed, as it will be considerably more difficult if delayed until after deployment. "The most definitive test of a simulation model's validity is establishing that its output closely resembles the output data that would be observed from the actual system" [59]. The phrase *would be observed* is used because most simulations are commissioned to study a system that does not (yet) exist. The method Law proposes requires considering both the *proposed* system and the most similar *existing* system. The simulation is first configured to run experiments in the existing system, and one concludes that its outputs match. Following that, the simulation is modified to reflect the proposed system. "The greater the commonality between the existing and proposed systems, the greater our confidence in the model of the proposed system" [59]. An example of this two-step validation technique is presented in Section 5.1.3.1, where we validate pedestrian movement behavior in comparison to an existing physical movement study performed by Daamen and Hoogendoorn.

MMOHILS encounters additional difficulties due to its placement within a real-world system. Law in particular is concerned with an over-emphasis on statistical methods,

stating that nearly all real-world systems produce *non-stationary* and *autocorrelated* output[2], and thus traditional statistical tests are not directly applicable. In addition, the classic *null hypothesis test* is flawed, as its query into whether or not the real system and the model are the "same" is clearly false: the model is an approximation *by definition*. One could argue that Law is over-stressing the contention, but we find his criticisms particularly warranted for the complex world of online virtual simulation. *Turing test validation*, *degenerate tests*, *comparison to a second model*, and *animation* are also presented as tools to aid validation efforts, but we emphasize that results comparison is the most *definitive* test of validity. In terms of our experiments, we attempted to demonstrate the validity of human elements whenever possible. In particular, Section 5.1.3.1 covers our attempts to validate via Law's notion of *proposed* and *existing* systems.

Next, MMOHILS introduces several additional validation requirements that are specific to its use of human agents in online virtual worlds. First, the *physical distributions* of each user over time must be analyzed and compared with that of real humans in similar environments. This ensures that humans have the capacity to move in a mechanically similar way in the virtual environment as they would in the physical world. Second, the *social characteristics* of agents must be validated. This step involves ensuring that emergent social patterns (such as queueing) exist in the virtual world as well as the physical. Third, *role validity* must be confirmed. All participants have a role in the simulation, and it is essential that they never violate this role during the course of the study. Finally, the *psycho-social state* of the users must be tested. Any psycho-social factor, such as stress or threat level, must be confirmed through the use of surveys. Studies may attempt to minimize the importance of psycho-social factors, but at the very least *immersion* must be tested to determine if users are identifying with their avatars. There are several methods to detect immersion, such as scanning chat logs for phrasing that indicates identity. For example, participants referring to their own avatars as "me" and "I" indicates a moderate level of virtual presence. Our previous work discusses these considerations in greater detail [153].

One challenging area in model validation of MMOHILS is *sensitivity analysis* of input parameters. To use an example from our work: agents in the library experiment discussed in Section 5.1.2.1 are modeled as ellipses, having a major axis of 57.9cm and a minor axis of 33cm. Despite coming from Fruin's established work, even these measurements have been questioned [84]. Sensitivity analysis would ask if this difference has a significant impact on model output. Can we say, for example, that our model is not

[2] *Non-stationary* means that observations of the same system over time will produce different distributions; *autocorrelated* means that the observations within a real-world output process are correlated with each other.

sensitive to the choice of an agent's breadth and depth varying by 5 to 10 cm? Sensitivity analysis is particularly difficult for a MMOHILS. If time, money, and manpower permits, the simulation can certainly be run multiple times with different input distributions. However, each run of a MMOHILS is considerably more expensive than that of a software agent simulation. We propose dealing with this problem by validating input parameters using traditional means when applicable, and relying on animation and the MMOHILS's greater commonality to the *existing* system (accomplished by leveraging real users) when traditional means do not apply. We consider MMOHILS to offer a trade-off between higher accuracy due to human participants, and the ability to run detailed sensitivity analysis via multiple replications. This is similar to the approach taken by the PDQ hospital simulation discussed earlier [54], which also uses strong system validity to overcome high data variance. If the purpose of a given MMOHILS is to create new agent models, then these models can be used to perform sensitivity analysis after the fact. We did not attempt to create such models, so this method was unavailable to us.

Chapter 5

Experimental Studies and Results

Given the size and the generic nature of the perennial simulation framework introduced in Chapter 4, it is highly unlikely that any single study will exercise all components of the framework simultaneously. Thus, three studies were undertaken which utilize the framework in a variety of ways. The first of these takes place at a high level of abstraction, assuming that all needed sensors are available for the purpose of testing a specific hypothesis via the MMOHILS niche configuration. The third study exists at an antipode, mirroring a physical system as closely as possible in an attempt to demonstrate construction of a credible prototype. The middle study exists both as a classical experimental study and as an appeal to the concept of legacy simulations. Combined, these three studies test a majority of the framework on the most relevant levels, and lead to a clear understanding of its implications and potential utility.

Figure 5.1 presents a complete breakdown of the various regions of the perennial simulation framework and which studies exercised each one. These regions are categorical, and combine to form the complete *feature-space* of the perennial simulation framework as a whole. Each region is sized according to its importance to the end goal of transdisciplinary crisis management simulation; hence, the "Simulation Focus" (which can be *foresight*, *hindsight*, or *decision support*) is the largest. The three studies detailed in Sections 5.1, 5.2, and 5.3 are listed below the feature-space, and have corresponding color-coded entries in the side panels that indicate their individual contributions to exploration of the feature-space. For example, **Study 1** exercises *decision support*, while **Study 2** exercises *hindsight* and *foresight*. Several notable features unique to each given study are also listed, such as **Study 1**'s scalability tests. Finally, note that a small percentage of the feature-space was intentionally left black. This represents the unexplored potential of the framework, which future work may attempt to fill in.

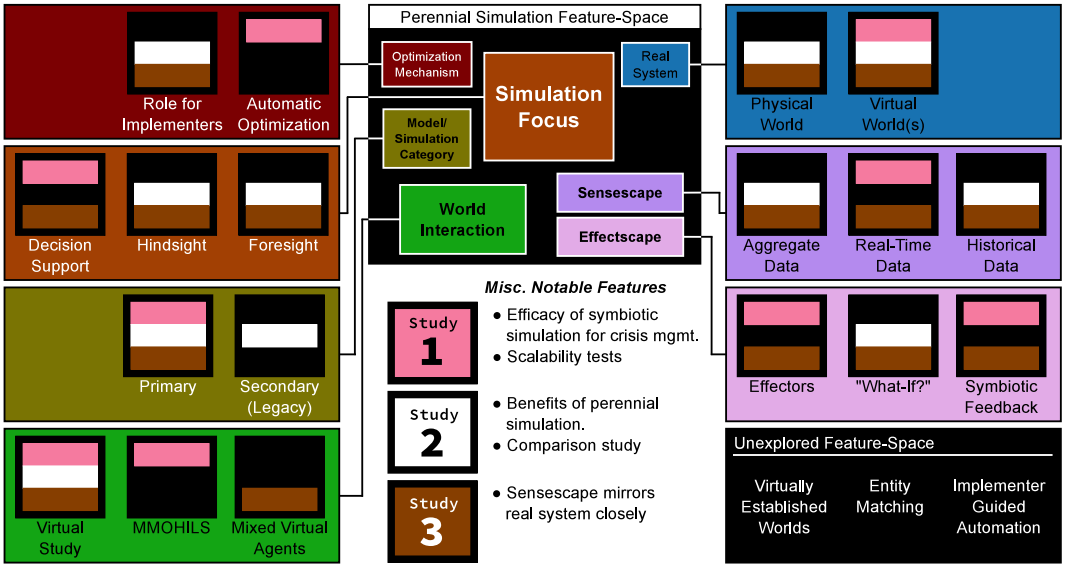

FIGURE 5.1: Coverage of framework concepts by experiment. Larger boxes in the feature-space represent more important concepts. Color-coded by experiment.

The feature-space of perennial simulation is considered to be sufficiently well explored for the purpose of this thesis. Of the unexplored regions, three of the more interesting elements are explicitly identified. First, as mentioned in Section 4.2.5, the notion of *virtually established* worlds is outside the scope of this thesis. We find the idea of virtual-to-physical world interaction fascinating, but to even begin to attempt this requires a strong understanding of the fundamental concept of *virtual studies*. Second, the notion of *entity matching* has not been explored. To match entities means to preserve their identities over a long period of time and across several simulation runs. This might be used, for example, to provide different symbiotic advice to agents who received community training than those who would not benefit from it. Although potentially useful, it is not clear just how much benefit such a technique would bestow, nor is it clear exactly how to implement matching in a robust and generic way. As such, we consider this esoteric to the core framework. Finally, the notion of Implementers guiding the automation process was not studied. Normally, the "optimization mechanism" listed in Figure 5.1 is either managed automatically, or it requires the Implementer to play a strong role. A third option exists: Implementers can create protocols based on simulation data, thereby automating various mundane but non-trivial decisions. The perennial simulation system would then perform the role of a knowledge-based expert system, aiding the implementers by freeing up time for them to focus on the truly novel aspects of a given crisis. Such a sophisticated idea is too many abstraction levels removed from the design goals of the perennial simulation framework for us to consider, although one would be remiss not to explore its potential at some later point.

Naturally, the use of the perennial simulation framework also simplified construction of the studies themselves. This will be covered in greater detail as each study is presented, but a few general remarks will help to set the context. **Study 1** was performed using an early prototype implementation of the framework, so it missed several opportunities (such as mixed physical/virtual agents) offered by the framework in its final form. Nonetheless, this study benefitted from the formulation of MMOHILS and the interactions between the constantly running model and the "What-If?" analyses. **Study 2** benefited greatly from the history window and sensor dependency graph, as well as the technique of encapsulating legacy models so that they could interact to some extent with perennial components. Finally, **Study 3** made heavy use of socket sensors as an abstraction to facilitate interaction within the sensescape. Sensor hierarchies and interaction with virtual agents were two other key elements that facilitated construction of this study. The aforelisted benefits are only the most prominent examples; the reader will be informed of additional advantages throughout the remainder of this chapter.

Each study will now be presented in a similar format, starting with a brief introduction to the purpose of the study, followed by its perennial setup and that of any experiments, and ending with observed results.

5.1 Library Egress Study

For our first study, a pedestrian movement MMOHILS was created and used for symbiotic-guided evacuation of a virtual building environment undergoing a mild hypothetical crisis. This "library egress study" was a complete, self-contained simulation study, and was the inspiration behind the perennial framework as a whole. It is described in its original context in [154].

The library egress study was performed to assess the effect of introducing a symbiotic feedback loop into a simulation of pedestrian egress. The benefit of symbiotic feedback was measured by comparing an egress simulation without this feedback to one that includes it. We expected that symbiotic feedback would improve both total evacuation time and average evacuation time, as well as improving the cohesiveness of egress on the whole. A secondary goal was to experiment with MMOHILS as a niche configuration of our perennial framework, since the concept of MMOHILS was still conceptual at that point in time. We were originally attracted to this niche configuration as it was uncertain how accurate software agents would be in representing human decision-making in response to symbiotic feedback. A tertiary goal was to examine the construction of perennial simulations in general. As an enabling first step, the library egress study was our largest and most comprehensive study.

5.1.1 Concise Overview

Most pedestrian experiments gather statistics based on a bird's-eye view of the world [84] [89] [91] [155]. In particular, Fruin's model of pedestrians models them as ellipses viewed from a top-down perspective [81]. Hence, we created a similarly-structured two-dimensional virtual world for players to explore. Users are represented as ellipses, their breadth and depth determined by Fruin's measurements. Players use intuitive mouse-centric controls to move around by clicking in the direction they wish to travel. Each avatar has a heading which rotates to match the direction of movement. Avatars collide with environmental obstacles and with each other, which halts their forward movement. Collisions due to velocity are resolved before those due to rotation, allowing users to "shuffle" around other players. Visual consistency is maintained by always showing the "current" player's avatar with pink hair; unintentional collusion is mitigated by portraying all other avatars with the same brown-haired model. A screenshot of the virtual world at an arbitrary point in time and from an arbitrary player's perspective is shown in Figure 5.2.

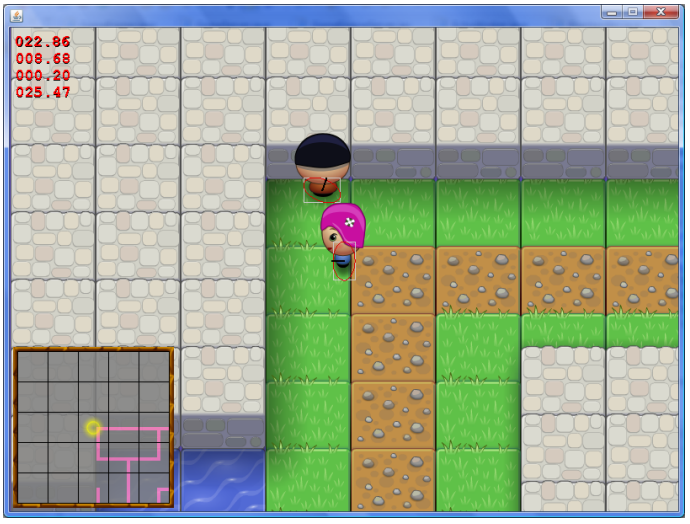

FIGURE 5.2: Two players exploring an online virtual world.

A suitable virtual world was created by studying existing university library blueprints. The final virtual construct was non-specialized (that is, it was not engineered to favor symbiotic egress), and was reasonably scaled with respect to the occupants. The use of MMOHILS allowed us to create the specific virtual world that we wanted to study, rather than be confined to the set of existing physical buildings (see: Section 4.2.6).

5.1.2 Perennial Components and Organization

The study just described was an inspiring force behind the perennial simulation framework, and as such it was only retrospectively incorporated into something resembling Figure 4.2. As a primitive study, it differs from the final system in three key ways.

- There was assumed to be only one virtual world. The concept of physical and virtual worlds co-existing and cross-influencing had not been clearly defined yet.
- Automatic sensor management was only introduced late into development. Historical sensors were programmed by hand and had to be managed manually. At least one simulation run was disturbed by human errors in regards to data gathering.
- Humans were not strictly modeled as Agents. It would have been difficult to incorporate software-controlled Agents into the environment without them seeming obvious and out-of-place.

The differences listed here are essentially historical artifacts and are considered non-critical to the operation of a successful perennial simulation. Indeed, the remainder of the system can be modeled using the latest component specification.

The library egress study exercised a small, vertical section of the perennial framework. The *real system* had a single (virtual) world with a single set of agents. These agents represented the building's occupants, and their positions and headings were sensed by the constantly-running model at regular intervals throughout each simulation run. A network flow model was used for "What-If?" analysis, and a single effector was used to modify and optimize the virtual world. This effector manipulated the simulation by giving group-optimized goals to the agents; an existing physical analogue might accomplish this using connected devices like mobile phones. The Implementers simply approved every plan, and a visualization was provided in real time. The rest of the framework remained unutilized.

As per the requirements in Section 4.2, two simulations were used to manage the library egress MMOHILS. The first of these maintained a constantly running model of the current state of the world. This was quasi agent-based, and included the positions of all users at all times. The second was a simulation of the possible routes users may take to exit a building, and was expected to provide "What-If?" analysis and symbiotic feedback to the real system. The latter served as a secondary model, and was chosen for its speed and simplicity. The following subsections will detail each component of the perennial system in turn.

5.1.2.1 Real System

The real system was composed of a single virtual world representing a two-story library. Book stacks were located on the second floor, with a variety of media (movies, periodicals) on the ground floor. Each floor measured 45.7 m by 29.2 m, with actual walkable space limited to 983.09 m^2 on the first floor, and 651.98 m^2 on the second floor. The real system was based off a series of university library schematics, but it corresponds exactly to no specific physical system.

The second floor, shown in Figure 5.4, is where all agents begin at the time of crisis. Following the announcement of an evacuation, they must proceed to one of four staircases (labeled A through D) and descend to the corresponding staircase on the first floor. As depicted in Figure 5.5, the first floor has one main exit in the front of the building and a second, smaller exit near the back. A legend describing the coloring scheme of these diagrams is shown in Figure 5.3.

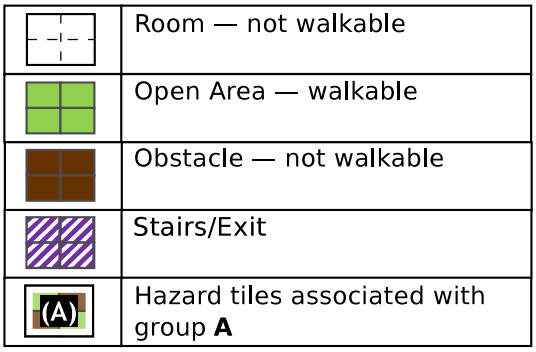

FIGURE 5.3: Legend of library diagram structures and obstacles.

The crisis event was considered non-specific, and the focus was on building egress. As this study was not concerned with perceptions of stress or peril by the agents, the cause of the egress cannot be something like a fire or earthquake. Rather, we informed users that the announcement had been given to evacuate due to several ceiling tiles that had collapsed on the first floor. They were told that the situation was not dangerous, but that they must evacuate with all due speed. The location of the collapsed ceiling tiles is shown in Figure 5.5, and will be discussed in greater detail later.

5.1.2.2 Egress Model

In addition to the constantly-running model (which approximated an agent-based system), an egress model is triggered at various intervals to perform "What-If?" analysis of the building's crowd behavior. At the time, we decided that a **secondary model** was sufficient for modeling egress, and we settled on the EvacNET network flow simulator.

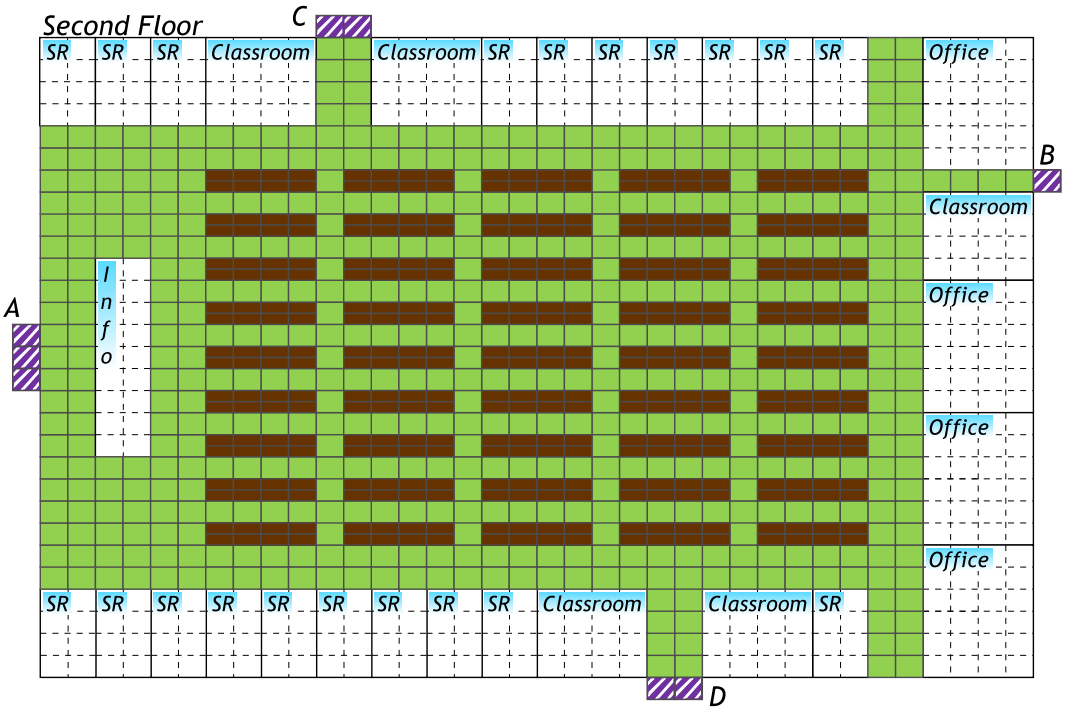

FIGURE 5.4: All agents start on the 2nd floor of the library.

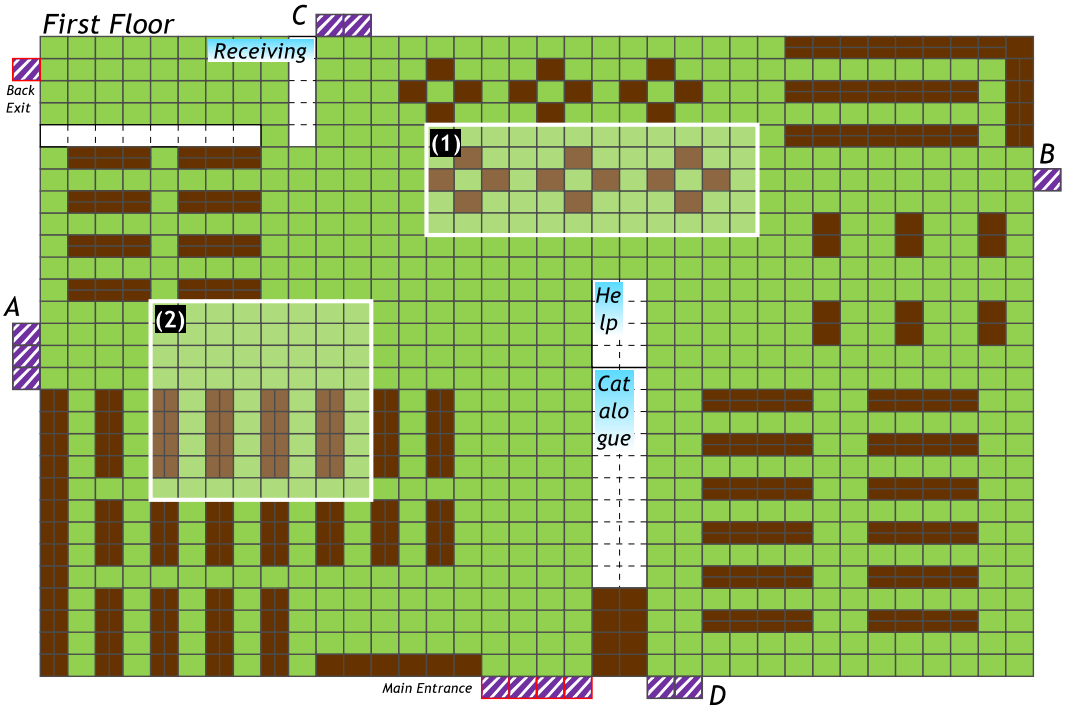

FIGURE 5.5: All exits and hazards are located on the 1st floor of the library.

EvacNET is a fast, powerful modeling software. It treats agent egress as a network flow problem, and was calibrated to Fruin's measurements of pedestrian capacities. EvacNET makes no appeal to behavioral modeling, so it always arrives at an optimal egress strategy. As described previously, our system coordinates with the real system by informing each user of this optimal escape route. Thus, it is necessary to have a model which guides users optimistically towards a "best case" egress scenario.

A filter was used to convert the virtual library world into a format that EvacNET could parse. Figures 5.7 and 5.8 depict the full graphical model of our library using the standard EvacNET syntax, which is summarized in Figure 5.6[1] [82]. The actual EvacNET input file generated by this filter is presented in full in Appendix A.

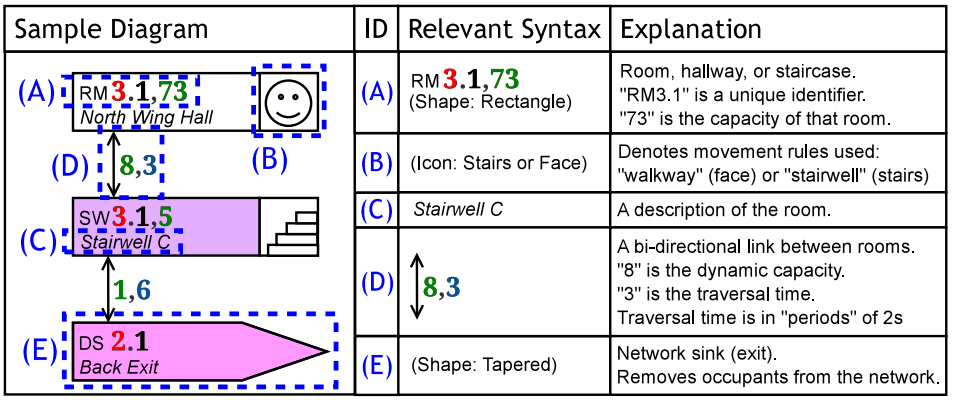

FIGURE 5.6: Example EvacNET syntax with descriptions.

Normally, bi-directional flow between two rooms would reduce the uni-directional dynamic capacity of each transit. However, we both expected and observed very little actual cross-flow, so there was no need to modify the dynamic capacity of each transit. This was consistent with the single-directional nature of traffic in egress studies in general.

Two hazard sets were introduced to restrict flow in the building and thus hinder egress. Hazard set 1 was intended to only slightly reduce flow, while hazard set 2 would severely impede movement. Both hazard sets only affected a small part of the model, thus allowing informed users to skip the congestion entirely. Specifically, each hazard set modified the EvacNET model in the following manner:

- Hazard set 1 reduced the dynamic capacities of **RM7.1↔RM6.1** and **RM7.1↔RM2.1** from 3 to 1, and increased their traversal times from 6 and 5 time intervals to 14 and 13, respectively.

[1] We have extended the official EvacNET specification with the following syntax: **RM1.2↔RM14.2** represents a bi-directional link between **RM1.2** and **RM14.2**.

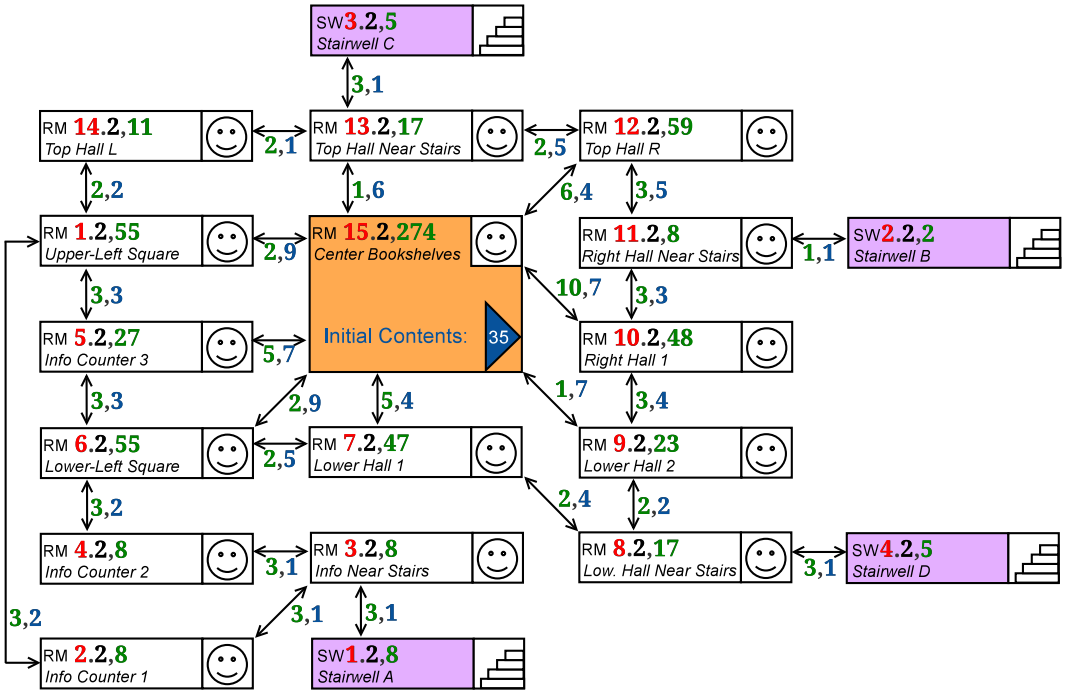

FIGURE 5.7: EvacNET model of our library, second floor.

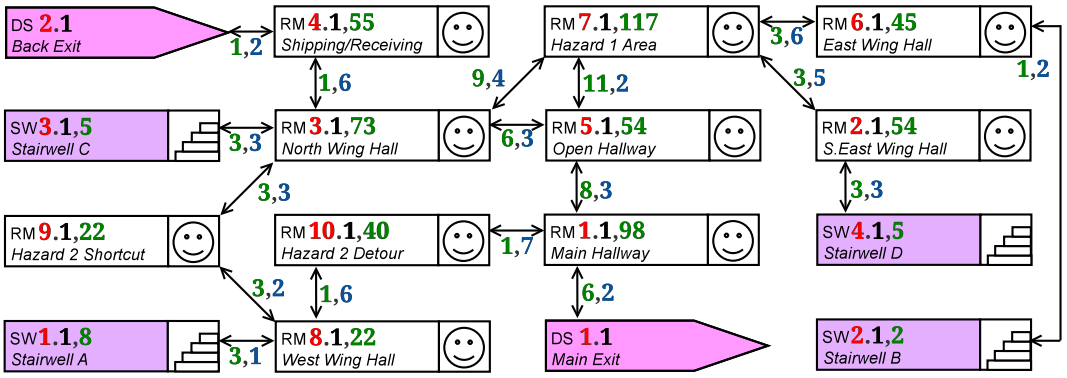

FIGURE 5.8: EvacNET model of our library, first floor.

- Hazard set 2 increased the traversal time of **RM1.1↔RM10.1** from 7 to 12 time intervals. In addition, transits **RM8.1↔RM9.1** and **RM3.1↔RM4.1** were removed from the model, to represent areas made inaccessible due to obstruction.

5.1.2.3 Practical Modeling Considerations

The previous sections presented the system as it was modeled. During the implementation stage, several practical alterations were required. First, "What-If?" analysis was limited to two trigger events: the start of the egress study and the traversal by a user of a staircase. Although the EvacNET model could run instantaneously, it was

essential that user guidance was only updated at points of increased uncertainty, lest participants become confused by constantly changing advice. Second, validation was not performed automatically, as Figure 2.2 might suggest, but was performed manually, and occurred offline after the experiments had concluded. Although automatic validation would have been more convenient, resources were limited and more time was spent running validation studies than adding optional features to the perennial system. Third, sensitivity analysis determined that congestion did not noticeably affect egress for the small number of users in our study. Thus, only one simulation was run for each triggered "What-If?" analysis. Fourth, an attempt was made to give distinct groups of users the same initial evacuation advice. This was based on the observation in [156] and [157] that people respond strongly to social networks involving people they are physically close to. In other words, directing a group to split up would likely result in users ignoring that particular piece of advice, and placing less confidence in future advice from the system. Finally, whenever possible, results from previous EvacNET runs were cached and reused. Several representative evacuation scenarios were computed offline and pre-cached. This optimization was not necessary for a small network like ours, but may become more pertinent as the cost of "What-If?" analysis increases appreciably.

All of these exceptions were superficial in nature, and most were motivated by considerations of performance or parsimony. None of them had any adverse effect on the accuracy or precision of the study.

5.1.3 Verification, Validation, and Calibration

A major problem faced by any MMOHILS developer is how to verify the client program on a potentially enormous diversity of client machines. Subtle inconsistencies can occur on different computers, due to things like 32/64-bit incompatibilities, variable behavior of floating-point arithmetic, or different versions of shared libraries. Our MMOHILS was carried out in a lab environment, so it would have been possible to individually verify each unique hardware configuration. However, we were also concerned with the general case of unannounced MMOHILS, in which the client machine is almost never known beforehand. To anticipate such a possibility, we decided to target a virtual hardware specification rather than a physical one. The Java Virtual Machine was chosen as the most robust, powerful, and readily available virtual machine at the time. We deployed using the JVM distribution from Sun Microsystems (now Oracle), since it was the most mature at the time of this study's inception.

By implementing our client (and server) in the Java Programming Language, we reduced the amount of verification required from potentially limitless hardware configurations to

one: the abstract hardware specification of the Java Virtual Machine. However, we also lost some amount of performance by relying on interpreted code. Several optimizations discussed in [158] were used to overcome the inherent performance hit caused by running interpreted code. In addition, we used ProGuard [159] to obfuscate the contents of the packaged JAR file beyond recognition; this could be used to limit cheating in unannounced MMOHILS, and it also boosted performance slightly.

Validation was more complicated than verification, and it required the MMOHILS-specific considerations described in Section 4.3.3.3. The following sections will detail the validation steps necessary for our pedestrian egress study.

5.1.3.1 Experimental Validation

Daamen's pedestrian movement exercises were used to validate the progression of users through the virtual world of our library egress study [155]. Figure 5.9 depicts the various virtual rooms created to mirror Daamen's real-world locations. The *rectangular* room set (A, B, and C) was used to study pedestrian movement on walkways. The *square* room set (D, E, and F) captured more complicated movement patterns which only emerge when cross-flow is present. Finally, the *bottleneck* room set (G and H) was designed to showcase certain non-optimal behavior which emerges only around bottlenecks.

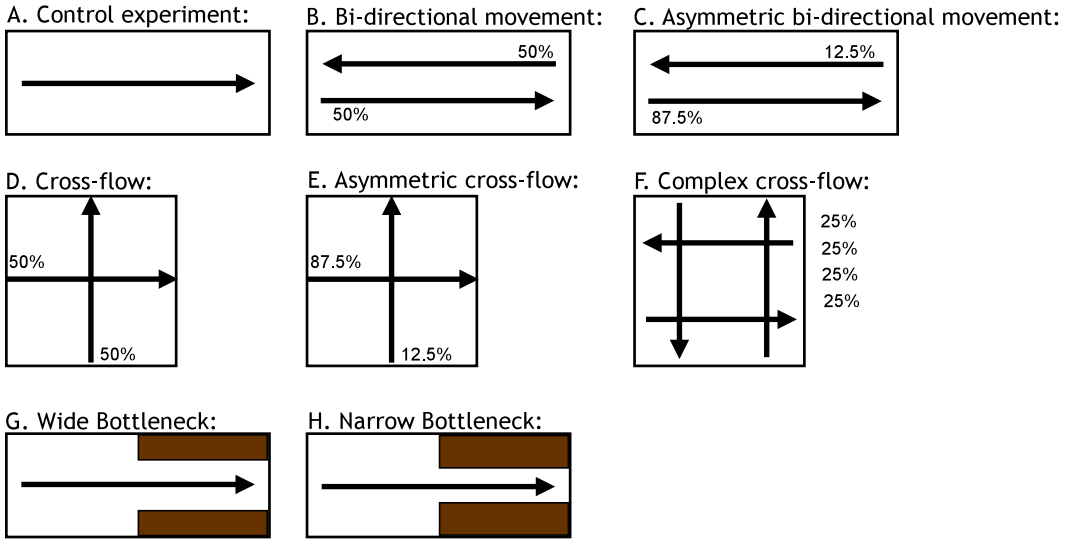

FIGURE 5.9: Dimensions of the rooms studied by Daamen, which were also used in our calibration exercise.

In total, 38 users took part in 16 simulation runs, each of which was located in one of the virtual rooms from Figure 5.9. Rooms A, C, D, and E were each simulated once; Rooms B, F, and H were simulated twice; and Room G was simulated 6 times. At the beginning

of each simulation run, users were divided into proportional groups and placed at their starting locations. They were informed to walk in the direction they were facing, at a normal movement speed, avoiding obstacles and other users similarly to how they would in the real world. If a user disobeyed these instructions, she was informed in real-time by a warning light. Continually misbehaving users could be automatically culled from the simulation; fortunately, no user ever crossed the threshold for total removal.

Movement data from the "wide bottleneck" simulation runs were compiled to form a profile of average user movement, which was then compared with the measurements of Fruin [81] and Togawa [83]. The bottleneck data sets were considered the most representative, as the narrowing room width forced users to anticipate and avoid collisions, much like the pedestrians in Fruin and Togwa's studies. The raw data were considerably noisy, so a five-point, centrally moving average was applied to smooth it. Culling of all data before a user began moving and after he reached his destination was also necessary, for obvious reasons. The number of "warning" points was low enough to be considered irrelevant. Plotted versus free-space-per-user, our data appear next to Fruin's in Figure 5.10; plotted versus crowd density, our data appear next to Fruin and Togawa's in Figure 5.11. In each plot, an exponential curve is fitted to our data to enhance visual identification of trends.

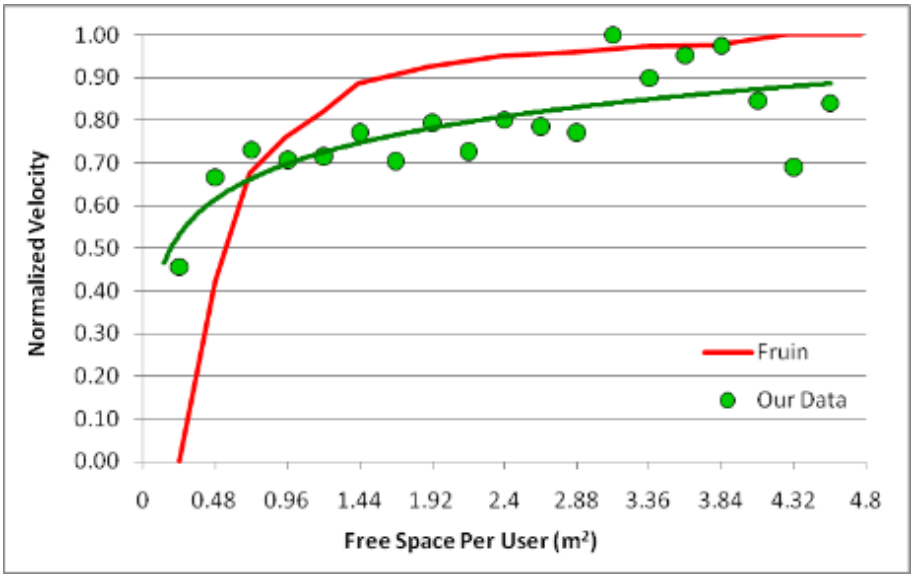

FIGURE 5.10: Normalized velocity versus free space per user.

Figure 5.10 shows a close match between Fruin's measurements and our own. User velocity increased with respect to surrounding free space, leveling out at 90% normalized velocity, compared to an expected maximum of 100%. Figure 5.11 helps to explain this disparity by showing that our data match Togawa's measurements much more closely than they match Fruin's. This is fortuitous, as Togawa's measurements are generally

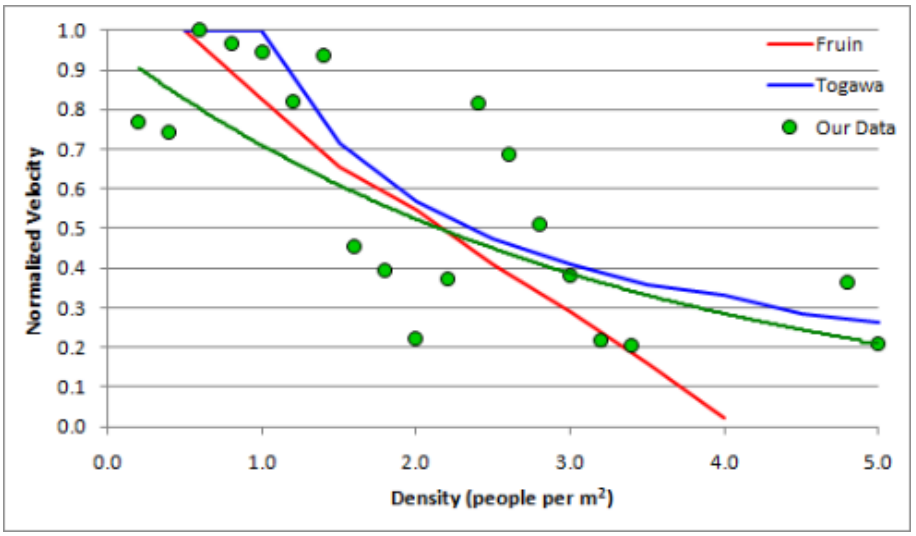

FIGURE 5.11: Normalized velocity versus crowd density.

Property	**Observed Value**
Average Forward Distance (m)	0.628 ± 0.606
Average Interleaved Agents	0.786 ± 0.413
Average Parallel Agents (D<1.5 ag/m²)	2.936 ± 1.663
Average Parallel Agents (D>3.0 ag/m²)	3.707 ± 1.703

TABLE 5.1: Social patterns of pedestrians.

held to be more accurate than Fruin's in regards to movement speed [84]. We consider minor differences at minimum and maximum velocities to be negligible. Between 2.0 and 3.0 people-per-square-meter, however, the data oscillate wildly. This is due to the sparsity of data points which fell within this range. Fortunately, this trend carried over into the movement study itself, as occupants tended to either cluster in slow-moving packs or move unimpeded at full speed. We monitored pedestrian density at all times to ensure that this value never fell into the 2.0 to 3.0 range; hence, the model was considered acceptably valid for our study.

Perhaps with a larger virtual space and longer simulation runs, more data points could have been gathered and greater confidence ascribed to our model within the middling density ranges. This would be mandatory if our library simulator were ever to be used for generic pedestrian movement studies. We stress, however, that data points gathered outside these ranges are perfectly valid for the library egress study, and that any potential sources of invalidity never presented themselves during our egress exercises. Thus, our simulation was valid with respect to its intended usage scenario.

Validation of social patterns, specifically those observed by Daamen [155], was done through analysis of re-runs of the simulation data. In particular, Daamen observed

that participants tended to spread out and fill all available width, even at low densities. Second, Daamen noted that pedestrians approaching a bottleneck would fan out in an attempt to get closer to their goal locations. Thirdly, Daamen observed that pedestrians tended to shuffle through a bottleneck at half-step interleavings, in part due to the increased congestion. We observed all three of these phenomena; two of them are depicted in Figure 5.12. Daamen also observed micro-lane formation. Although this behavior was present in our simulation, it was largely suppressed by the short duration of each simulation run.

Table 5.1 lists several social properties of our agent population. As agents funnel through the bottleneck, they tend to look straight ahead to what is termed the "forward agent". The *average forward distance* is either the distance between these two agents, or zero if no forward agent exists. A good indication of the half-step interleavings observed by Daamen is the presence of a separate agent between the current and forward agents. This is the *average interleaved agents*. To measure the tendency of pedestrians to fill the entire room's available width, we measured the *average parallel agents* count for both low and high density values. (Middling densities, as previously discussed were not considered due to noisy data.) One pedestrian is considered in parallel to another if the latter is within 50 cm (forwards or backwards) of the former, and agents count themselves into this total. For the most part, the data match what we expect, although it should be noted that visual analysis is valuable in cases such as this, where patterns are difficult to extract from the raw data. One exception was the average micro-length, which proved difficult to isolate automatically and was also not readily evident from a visual review of the data. Future work may consider longer experiment times or more complex room layouts in an attempt to extract this particular social pattern.

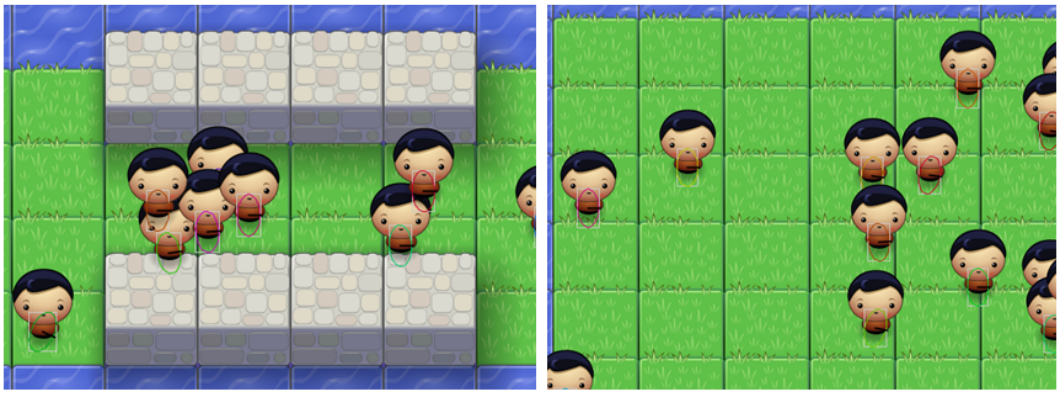

FIGURE 5.12: Screenshots from our live MMOHILS experiment. *Left*: users overtaking each other by half-steps in narrow corridors. *Right*: all available passageway width is used, even at low densities.

Role validation used case-based reasoning similarly to [80]'s approach. By populating a *case base* with known "misbehaving" entities, we were able to match users against this case base in real time to determine if any of them belonged in the set of misbehavers. If so, we warned them. Our case base was a simple *classification case base*, and used the agent's speed and directional heading as parameters. This configuration could detect bad behavior like abrupt stopping or sudden changes in direction. Detection of such behavior could, of course, be performed after the simulation runs. However, we felt that warning users in real-time improved the robustness of our data by allowing users to correct their own errors and thus avoid long-term data degradation.

Psycho-social validation was straightforward; we consciously chose an environment where stress was not a factor, and which minimized other psycho-social constraints. As pointed out by [64], however, humans tend towards "optimistic bias" when asked to estimate chances applied to themselves. More importantly, this tendency does *not* apply when humans are asked to estimate the same probabilities applied to others. Thus, if a user in our simulation did not consider his avatar to represent *himself*, but rather saw his avatar as some *other* entity, then the validity of his decision-making process would be in jeopardy. For the purposes of our study, which contained very few probability-based decisions, a simple debriefing session was sufficient to determine how each player viewed himself in the online world. See Section 4.3.3.3 for a discussion of possible extensions to psycho-social validation.

5.1.4 Structure of Experiments

All experiments excluding those discussed in Section 5.1.3.1 took place in the virtual library world described in Section 5.1.2.1. Users were granted time to explore the virtual building before the simulation began, and they performed a mock egress with no hazards in place. Both exits and all four staircases were made known to all users during this phase, and a floor plan of the library was displayed in a central location. Hazard sets were not disclosed beforehand. Figure 5.13 shows several users initially exploring the online virtual world.

A total of 38 participants took part in our study. Not including the validation runs described earlier, there were four *comparative* and four *progressive* simulation runs. Participants in the *comparative* runs were split evenly into two groups, one of which received symbiotic guidance on the first run, the other on the second. This was intended to counter the natural learning effect from one run to the next. The *comparative* runs all used hazard set 1, and were designed to objectively measure the benefit that symbiotic feedback lent to egress. We expected symbiotic simulation to offer a definite advantage.

FIGURE 5.13: Photograph of users exploring the virtual world in real time.

Conversely, the *progressive* runs used hazard set 2, and required all users to be online at the same time. The purpose of these runs was to see if symbiotic simulation had any noticeable long-term effects on egress time. Our hypothesis was that learning and a general increase in confidence would lead to reduced egress time overall.

All participants' computers were connected to the same local network, with <3 ms round-trip network latency between each computer and the server. Two neighboring computer labs were used to hold all the participants and the server; combined, the experiments took four hours to complete.

5.1.5 Discussion of Results

Generally speaking, total evacuation time was 25% shorter with the introduction of symbiotic feedback. The EvacNET model suggested routes that avoided potential bottlenecks, and real-time goal-directed instructions helped guide occupants out of the building quickly and efficiently. Total evacuation time is sensitive to outlying data points (i.e., a single slow user) so the average evacuation time should be considered more indicative of the real system's performance. (Total evacuation time would be more indicative for cases with a source of peril, such as building fires.)

Figure 5.14 graphs the average time taken to evacuate the library building with and without instructions generated by the symbiotic simulation. For groups A and B, and then the two combined, the left-hand bar shows the average time to evacuate the building with no guidance, and the right-hand bar shows the average time to evacuate with

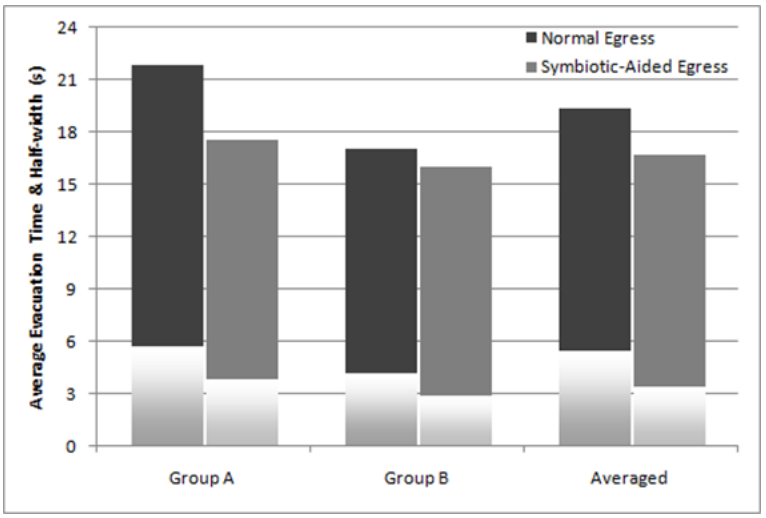

FIGURE 5.14: Average evacuation time, with and without symbiotic simulation.

symbiotic-induced guidance. The white segments towards the bottom of the graph represent one standard deviation of the averaged values. Figure 5.14, makes it clear that symbiotic simulation helps to reduce average egress time. Improvements of 14% with a 37% reduction in standard deviation were observed. Put simply, the symbiotic runs were more cohesive than the normal runs, in addition to being faster. This implies that symbiotic simulation helped to bring individual evacuation time closer to some optimum value.

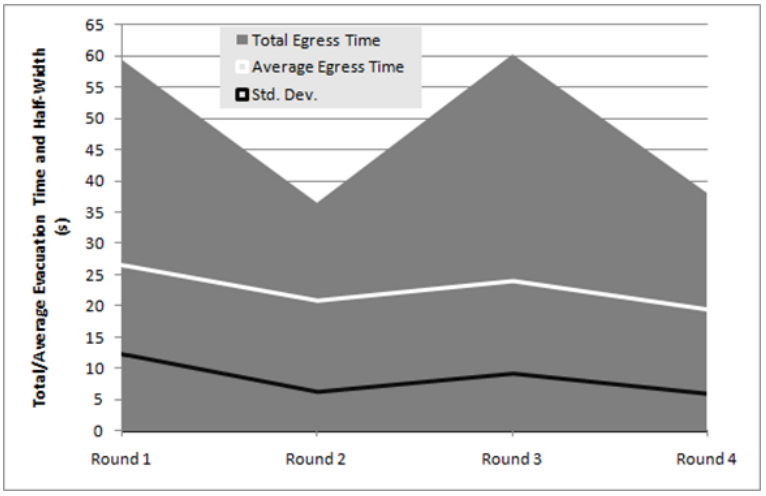

FIGURE 5.15: Evacuation time ordered sequentially, exhibiting some evidence that symbiotic simulation enhances learning.

Figure 5.15 shows the egress times of all users after several iterative runs of the second hazard set. Rounds 1 and 3 use no symbiotic guidance, while rounds 2 and 4 feature symbiotic simulations. Total egress time is graphed in the background; the white and black lines represent average egress time and one standard deviation of the average time

respectively. Although total egress time remains roughly unchanged between pairs of like rounds, the average time to evacuate the building clearly decreased after repeated runs. Normal egress times dropped from 26.6 ±12.3s to 24.1±9.2s, and symbiotic egress times dropped from 20.9 ±6.3 to 19.5 ±6.0s.

The difference from one symbiotic-guided egress to another is only 7%, with virtually no change in standard deviation. However, the normal egress times showed a much more significant improvement of 10%, with a reduction in standard deviation of 25%. Average egress time decreased, and standard deviation lessened. These four rounds were carried out after those in Figure 5.14, after users had become very familiar with the environment. The only change in the environment was in the set of hazardous tiles. We expect that the first symbiotic simulation (Round 2) had the unanticipated effect of *training* users in how to exit the building given this new hazard set, thus indirectly affecting egress times in Round 3. This is not conclusive, but the data indicate a strong likelihood.

The crucial observation relating to our results is that *any* decrease in evacuation time denotes success. The environment in which the evacuation exercise took place was neither very complex nor heavily congested. It was not designed with a bias towards symbiotic egress. Hence, our results clearly demonstrate the power of symbiotic feedback, if only to enforce and enable occupants to expedite egress in an environment they are already somewhat familiar with. To that end, our results also directly imply that users will, *ceteris paribus*, accept guidance from an authority figure in times of uncertainty. This is consistent with other research in the field, such as [156] and [157], and is crucial to symbiotic simulation when humans are active agents.

The results observed relied on the fact that sensing within the virtual world was totally accurate. Some amount of uncertainty would be expected in a physical analogue of this building. Sensing each human participant's position, determining if each user was following his instructions, and actually informing the user of his target staircase or exit are three key areas where noise should be expected. In addition, sensing the total number of users would certainly be prone to noise; normal building occupants might not have a tracking device (i.e., their phones) turned on, and they might crowd together, complicating approaches based on infrared sensing or facial recognition. We expect that as the inaccuracy of the sensors and effectors increases, the total benefit gained from symbiotic simulation would decrease. Fortunately, the virtual library environment was both very simple and sparsely populated. Truly disastrous egress conditions tend to arise when neither of these conditions hold, so there is every reason to expect that most physical environments would benefit enormously from symbiotic feedback. In short, there are

considerations that will both decrease and increase the efficacy of symbiotic simulation in actual applications, but the net result is likely to be overwhelmingly positive.

5.1.6 Scalability

In the interest of examining the scalability of perennial simulation, we performed a series of experiments which stressed the simulation we had constructed for the library egress scenario. We were primarily concerned with determining the per-agent overhead of experimental MMOHILS. In addition, we attempted to measure the space-wise overhead of our perennial simulation, as well as the cost of symbiotic exploration via "What-If?" analysis.

The per-user overhead of the simulation comprised two parts: the overhead of each human participant in the virtual world, and the overhead of each agent in the EvacNET model. The latter will be covered separately, as it partially indicates the cost of providing symbiotic decision support. The former relates directly to the cost of each agent in the constantly-running model; therefore, it can be used to provide an accurate predictor of the overhead of maintaining a perennial simulation for crisis management at a *Virtual Study* location. For practical reasons, agent scalability was tested using **headless clients** —randomly-walking game clients which performed no graphical processing, thus allowing a large number to be run on the same machine at the same time. An Intel Core i7 MacBook Pro (8 effective cores) with Java 1.6, 64-bit server edition was used to host both the headless clients and the server. Figure 5.16 depicts the performance with a varying number of agents. For the sake of discussion, a frame refresh of 90ms corresponds to about 11 fps, which is approximately the limit for acceptable real-time immersive experiences. The virtual (upper) data points were obtained with headless clients, while the real (lower) data points were measured in our previous work [158]. Our actual experiments in [154] note that, when all 38 test users were connected, latency was 45.5 ms. This is consistent with the "stepping" pattern that occurs as demand on the server increases. The cause of this stepping pattern is well-known to video game developers: when a frame update deadline is missed, the current update is delayed until the next frame, leading to a net loss (in this case) of 30ms regardless of the actual increase in latency.

Several aspects of Figure 5.16 merit further discussion. First, although the machine used for testing virtual demand was much more powerful than the machines used to measure real demand, there is about 10 to 15ms of overhead on the virtual data points. At the same time, CPU utilization (not shown) only peaked after 80 connected clients. We expect that the difference in performance is due either to overhead imposed by

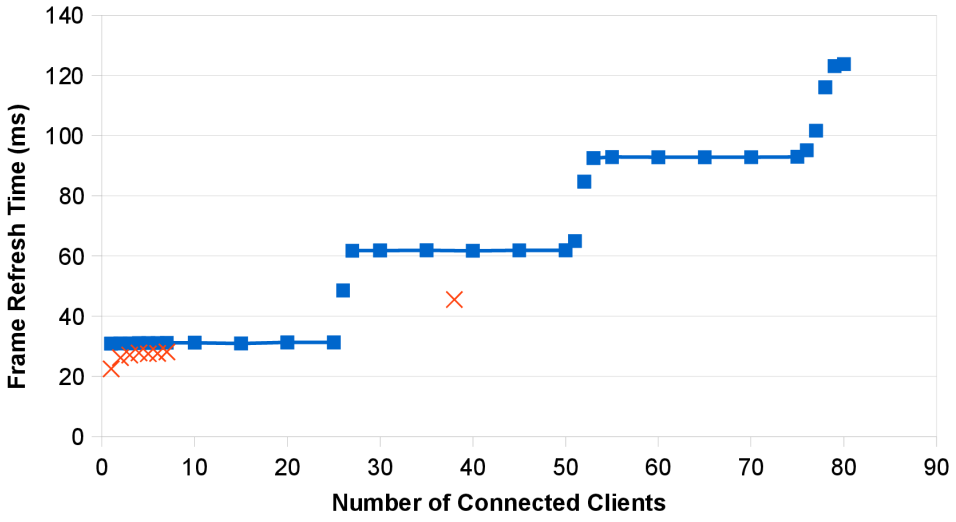

FIGURE 5.16: Performance of our MMOHILS as client connections increased. The upper data points (blue squares) were measured using headless clients, while the lower data points were gathered on physical, networked machines.

task switching at the operating system level, or by performance differences in the JVM on Windows versus OS X. Second, we observe that the plateaus in Figure 5.16 are evenly-spaced: the first occurs after 25 connected users, the second after 50, and the third after 75. When a borderline number of agents was connected, updates would only occasionally miss the frame boundary, leading to values outside the plateaus. Overall performance was not affected by Java's threading model: the 60-user study was re-run with half the agents forced onto a separate JVM (and thus a separate core), and performance was observed to be the same. As a result, we can reasonably expect that using more powerful servers would increase the number of agents supportable at each plateau, leading to a cumulative boost in performance and allowing more simultaneous participants in the MMOHILS. At some point, the bandwidth of the network would also have to be considered —all of our tests used commodity hardware, but proper game servers typically exploit **teaming** of multiple network cards and feature high bandwidth, low latency connections to a shared intranet. Finally, we reconfirmed our observation in [158] that the slowest client limits the height of each plateau on our graph. This height difference accumulates, such that extrapolating our real-world experiments to 60 connections would theoretically equate to 66ms. Compared to our virtual refresh rate of 93ms, this is an effective improvement of 4.4 fps, gained solely by targeting more appropriate hardware.

Given the "stepping" performance in Figure 5.16, combined with our observation that the network, client machines, operating system, and JVM used can greatly affect the height and breadth of each plateau, we would make an educated guess that the limit on

the number of connected users in an **experimental** MMOHILS study (with reasonable fps) can easily reach 100, and is likely somewhere between 200 and 300. Estimating an upper bound on **unannounced** MMOHILS is more difficult, as these simulations are typically embedded in online MMO games which feature several hundred thousand simultaneous online players. (Recall that **experimental** MMOHILS aims to be easy to develop and suitable for small, trusted groups, while **unannounced** MMOHILS attempts to achieve a maximum amount of scalability regardless of development cost.) Commercial games invariably use information management techniques similar to those found in the HLA, thus partitioning the user base and reducing the per-player bandwidth to an acceptable level. As a result, the fps of each client may never dip below a suitable level of immersion; rather, the limit of the MMOHILS will manifest as a result of exhausting the simulation's capacity to generate valid data with a user base full of inconsistent world views. Commercial MMOs, as described in Section 3.8, often achieve tens of thousands of simultaneously connected players. As a caveat to the anecdotal evidence of MMO games, note that scalability is achieved by making "assumptions about user distribution, update frequency, and read and write operation in the environment", and that such constraints ignore the true cost of "creating undesirable restrictions on how users should behave and what functionalities they may expect" [104]. As a result, although MMOs provide a good example of the absolute limits on agent connectivity, *full-featured* virtual environments and practical simulation-specific considerations may reduce this value. For example, [104] lists no more than 1000 and as few as 50 connected users as the limits for research-oriented virtual world technology. We would therefore cautiously estimate that unannounced MMOHILS should aim to incorporate several hundred to several thousand simultaneous online human agents.

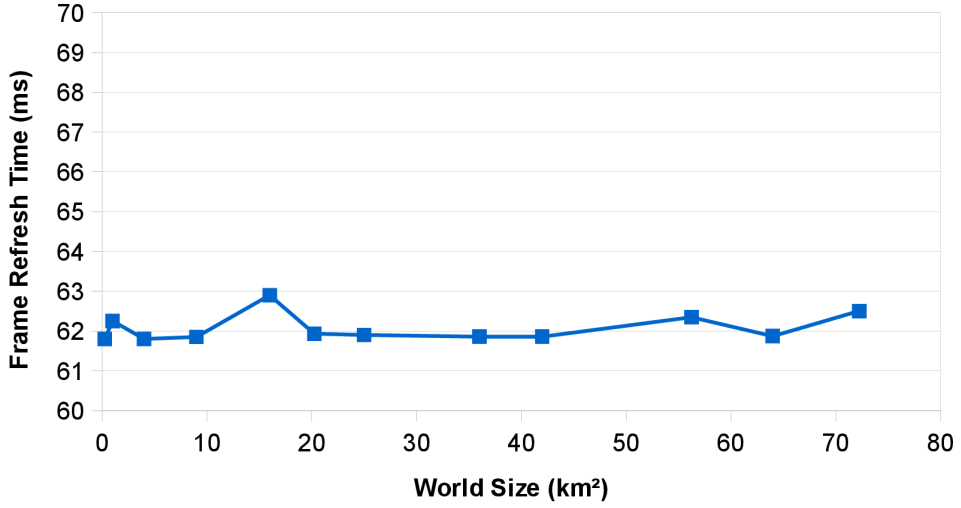

FIGURE 5.17: Performance of our MMOHILS client with 50 connected users as world size increases.

Our second performance measurement concerned the scalability of the size of our virtual world. Figure 5.17 shows our results. The smallest world measured was 0.25 km^2 — roughly the size of 35 soccer fields. A population of 50 headless clients was randomly scattered throughout the area. The choice of 50 agents corresponded to the upper limit on the second plateau measured earlier, and was intended to test the sensitivity of the simulation to changes in world size. The world scaled successfully up to 72 km^2 with no significant loss of performance, but the client machine froze for larger worlds. This occurred because the clients were programmed to keep each square meter of terrain in memory at all times, leading to an overhead of 0.6 GiB[2] per client, or 30 GiB overall, at world sizes above 80 km^2. Most of this memory was paged out at any given time, but the sheer volume of data eventually overwhelmed the system and led to thrashing. This problem could be easily remedied through obvious mechanisms, but 72 km^2 was considered sufficient, as it represents an area larger than Manhattan Island, and is thus more than enough space for the several hundred agents that **experimental** MMOHILS can support.

Regarding the maximum world size for **unannounced** MMOHILS, we can arrive at a reasonable estimate by considering a similar online game, also programmed in Java, with the same world building-block resolution of 1 m. *Minecraft* is an immersive 3-D game released in 2011 with a theoretical maximum world size of 157.5 square megameters[3]. Typical *Minecraft* multiplayer worlds range from 300 km^2 for mid-range, amateur setups to 850 km^2 for high-end, professional servers [160]. The longest straight-line distance perambulated in *Minecraft* without a noticeable increase in lag is 292 km, leading to a justifiable maximum world size of 85264 km^2, which is roughly 120 times the size of Singapore [161]. The game is still mostly playable beyond this point, but the increased reticule lag threatens immersion validity; thus, we consider this to be the upper bound on world size for unannounced MOHILS.

Finally, we attempted to measure the overhead of providing symbiotic decision support at the time of crisis. An important aspect of this is the quality of information that one can extract from the simulation within a short timespan from the occurrence of the crisis event. Gauging an upper bound on this is important, as effective crisis management is characterized by strong information superiority. The elemental component of performance for symbiotic simulation is the time taken to run a single "What-If?" scenario. This can be seen as the atomic unit of performance, as running multiple "What-If?" scenarios at the same time is trivial with sufficient hardware, but improving performance of the unit itself is challenging and often requires re-designing the simulation engine to

[2] 1 GiB is equal to 2^{30} bytes, or about 1.074 GB.

[3] As its name implies, *Minecraft* allows exploration beneath a world's surface, but only to a depth of 255 meters. From a top-down perspective, therefore, it functions exactly like our 2-D simulator, and one would reasonably expect the performance to be no better than that of a comparable 2-D world.

respect specific parallelization assumptions. Measuring this elemental performance was challenging in our case, since our EvacNET model resolved instantaneously.

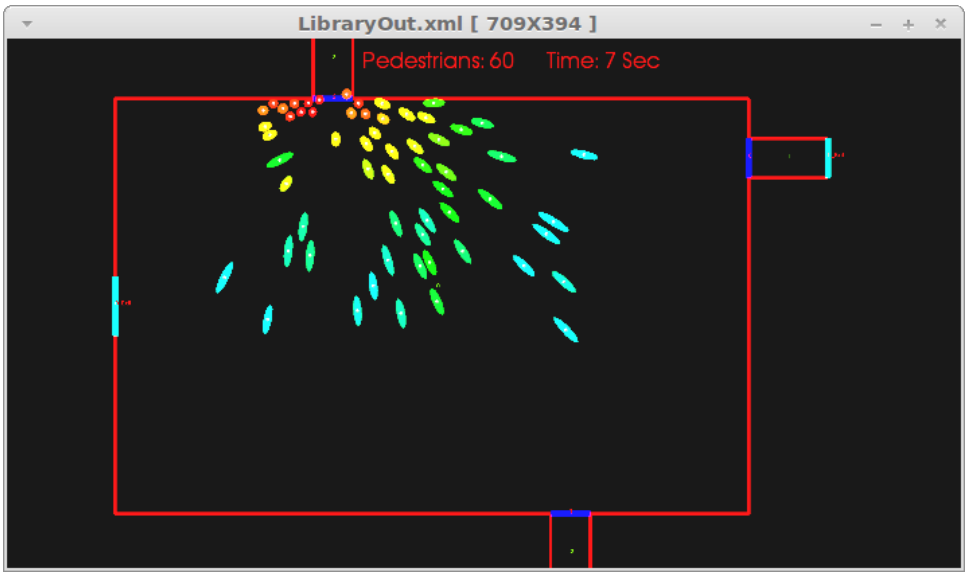

FIGURE 5.18: Snapshot of our library virtual world running in OpenPedSim. The velocity of each agent is indicated by its color and length: cyan ellipses for fast-moving agents, and red circles for agents that have stopped.

In the interest of obtaining a meaningful measurable result, we re-implemented our egress environment using the OpenPedSim simulator. This pedestrian dynamics simulator uses a **generalized centrifugal force model** (gcfm) to simulate pedestrian movement at a microscopic level. The gcfm is somewhat similar to Helbing's social-forces model, in that it treats agents as Newtonian particles and moves them through time via numerical integration on a set of force equations. The gcfm attempts to improve upon Helbing's original model by providing more accurate results for highly congested rooms [162]. Figure 5.18 shows a screenshot of the visualizer running a version of the second-floor library environment with obstacles removed. Seeing as it builds on Helbing's work, OpenPedSim is subject to the criticisms in Section 3.6 and should therefore not be considered valid in the context of providing symbiotic feedback for our users. Rather, it was chosen because of a few key properties, listed below.

- The model is microscopic, individualistic, and continuous in both time and space. It therefore represents one of the highest resolutions of information one might need for egress modeling.
- The simulator runs at a very fine-grained resolution of 10 ms, thereby representing the most computationally intensive egress models.

- The software design is similar to EvacNET, in that it was programmed in C++ and requires the use of external resources (such as pipes) to process output in real time. Thus, any slowdown will likely be due to the complexity of the model, rather than, e.g., performance differences between programming languages.

For these reasons, OpenPedSim is considered to be a good representation of the worst-case performance of any complex, agent-centric simulation system. To further stress the performance of OpenPedSim, we created a model of the library world's second floor with all obstacles removed. (Removing obstacles allowed more agents to be in close proximity to each other, thus adding to the complexity of the **gcfm** at each time tick.) An increasingly large group of agents was scattered randomly throughout the navigable space, and the order was given to evacuate. Based on observed data with real users, we limited the simulation time to 30 seconds, as the first notification that users received generally removed the most uncertainty and provided more symbiotic benefit than any single other message. Figure 5.19 depicts the results of our performance measurements.

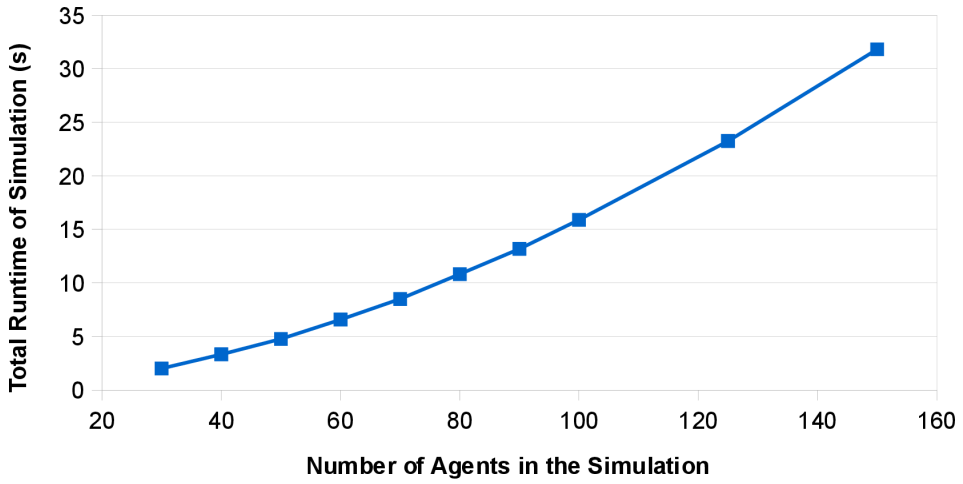

FIGURE 5.19: Performance of OpenPedSim operating as a symbiotic smiulation as number of agents increases.

Our results allow us to arrive at a reasonable estimate of elemental performance. It is clear that the simulation becomes impractical for symbiotic-guided egress after 140 users, as a prediction of the world state 30 s into the future takes at least that long to compute. This does not completely eliminate the utility of the simulation; for example, one might use the difference between the OpenPedSim results and observed values to tune a historical case-base in real time. For pure symbiotic feedback, however, we would expect a delay of no more than 10 s, or 80 agents as per Figure 5.19. As the elemental performance factor, this is quite low, even by the restrained expectations of **experimental** MMOHILS. Even worse, this must be considered alongside the generational

nature of search algorithms such as the Pareto or utility function approaches mentioned in Section 4.3.3.1. The potential for slowdown increases as we demand more accurate, long-running results for ever-broadening areas of interest.

Fortunately, there are several optimizations and mitigating factors that allow us to reasonably expect better real-world performance. First, the poor performance from Figure 5.19 emerges only when 80 or more agents are gathered in a single room and simulated at the operational level. For various hallways, stairwells, and corridors, a simple network flow model like EvacNET will suffice; thus, as agents exit the main room, they are removed from the calculation of the **gcfm** for the remaining agents, leading to an exponential drop in complexity. Second, we expect that a high-resolution technique such as **gcfm** will only be used to "zoom in" on a particularly troubling area. For example, if several dozen agents are taking longer to exit a room than intended, then a small **gcfm** can be used to determine if their physical motion is the root cause of this delay, as in [67]. In such a case, the runtime behavior of agents is already outside the scope of what is expected, so the ability to extract any information at all overrides symbiotic decision support's normally strict requirements. Third, partial or low-resolution results can be streamed to Implementers or users in real-time, while the slower models resolve. This can be accomplished by running at a larger time step (within the acceptable range of the **gcfm**) or running a second, faster model and then attempting to improve the preliminary results. This particular approach is useful for its flexibility: if the deadline for action has arrived, then the most-accurate model which has fully resolved may be used. Finally, the worst-case performance for OpenPedSim is nonetheless compartmentalized. In the case of a large-scale, **unannounced** MMOHILS setup, each individual room will have its own population of users. Large rooms can be migrated to separate hardware, and corridors can manage supply and demand between these numerous independent models. Instead of having an upper bound of 80 agents in total, such a system would have an upper bound of 80 agents per room, which is far more viable. For these reasons, we expect that the elemental performance of symbiotic simulation can reach an acceptable level with a modest amount of development effort.

Evaluating the performance of symbiotic decision support for **unannounced** MMOHILS is significantly more difficult in this case. For optimistic users, low-resolution models such as EvacNET have nearly infinite capacity, and should allow a full exploration of state space even with thousands of users. More pessimistic users might observe the slow performance of the (high-resolution) OpenPedSim model. Based on our measurements of real users, we found that symbiotic egress feedback need not be sophisticated to have a profound effect (see Section 5.1.5), so we tend towards the optimistic opinion. To this end, we conclude that, unlike MMOHILS in general, reliable symbiotic decision support will be bound not by the users' perceptions of consistent world views, but by a

trade-off between accuracy and resolution. At one end of the spectrum will be models such as EvacNET, which feature excellent confidence regarding very general, high-level tactical egress decisions. At the other end will be resource-hungry models such as **gcfm**, which provide extremely detailed information about low-level specifics such as physical movement at the expense of precision. A full breakdown of the real-world performance characteristics of **unannounced** MMOHILS is beyond the scope of this thesis, but at the very least an accurate, coarse-grained level of symbiotic feedback is feasible.

5.1.7 Significance and Conclusions

The library egress study described above was the first simulation study to use MMOHILS, as well as the first use of our nascent perennial framework. Regarding the former, this study helped to confirm the strengths of using real humans in a virtual experimentation environment. This approach allowed us to be flexible in defining the environment for our study and to maintain a homogeneity with respect to the various validation environments. Validation succeeded on all levels, confirming that users in the virtual world both maintained their roles throughout the simulation and exhibited the physical properties of actual pedestrians in real-world crowds. Although each run of the MMOHILS required significantly more resources than a traditional simulation, the data gathered were of far greater validity. Reconfiguring the simulation to study a different building or hazard set would also be far easier than re-running a data-gathering exercise with real users in the physical world.

In addition to its function as a pioneering MMOHILS study, the library egress study was also a full-fledged perennial simulation of a virtual crisis management exercise. The purpose of this perennial simulation was to study the effect of symbiotic simulation and agent-specific advice on humans exiting a building. In particular, we showed that, consistent with our expectations from the literature, users responded positively to advice in a virtual environment similarly to how they would in the physical world. Moreover, the consistent decrease in both average egress time and standard deviation of egress times indicates that symbiotic feedback can optimize even a simple pedestrian movement exercise such as our experiment. This result was not obvious *a priori*, and represents a significant positive aspect of symbiotic simulation when applied to human egress. Without the perennial simulation framework, it would have been difficult or impossible to demonstrate this effect. Finally, we performed a series of stress-tests to gauge the performance of MMOHILS and symbiotic decision support in general. Based on these measurements and reasonable assumptions, **experimental** MMOHILS was found to scale upwards of 100 users, and **unannounced** MMOHILS could reach several thousand. World size had no effect on this number. Symbiotic feedback was somewhat

more limited, depending on the fidelity of information required. High-level tactical information could be provided at no cost, while microscopic movement details required clever optimizations after 80 connected users. Middling resolutions could likely scale to the full size of **unannounced** MMOHILS; therefore, we conclude that MMOHILS and symbiotic decision support scale to acceptable levels.

As was previously mentioned, a MMOHILS is merely a perennial simulation set up to exhibit certain properties. This setup is formally called a *niche configuration* after its use in highly targeted studies. Thus, the benefits of MMOHILS reflect positively on the perennial framework as a whole. In particular, the library egress study demonstrated the framework's *flexibility* in analyzing new scenarios. Sensors, effectors, and simulations had clearly defined roles, and virtual worlds were used to captilize on the ability to study a hypothetical building environment rather than requiring a corresponding real-world location. Operating costs were high compared to a software agent-based simulation, but were much lower than that of a similar real-world experiment. In addition, the use of a virtual environment allowed data to be gathered and processed in a simple, streamlined manner. Re-using the same software at multiple different virtual locations (i.e., when validating to Daamen's environments) required little effort and could be carried out instantaneously. All of these positive aspects come from the specification of MMOHILS as a niche configuration of the perennial simulation framework.

5.2 Incident Response (Traffic) Study

The second major use-case of the perennial framework was a traffic simulation applied to investigate incident response on a university campus [163]. This built off our previous perennial study and provided several key improvements and novelties. Most importantly, it served as a multi-level study, capable of running *foresight* or *hindsight* simulations, while exhibiting potential for *symbiotic decision support* via its reliance on vast amounts of processed real-time data. Additionally, it provided a robust method for interfacing legacy simulation components with the perennial framework, something which had been raised as a concern when the framework was first reviewed. This study also provided some direct inspiration and refinements of the framework proper, such as the history window and the sensory dependency trees. Finally, a minor point of interest is the automated image processing algorithm used to mine traffic data from security cameras in real time (with some caveats, see Section 5.2.3.1).

The simulation created for this study was designed to estimate the time taken by security vehicles to respond to incidents under different levels of congestion and at different locations on the campus. Real-world traffic data captured from security cameras were

used to test the efficacy of dispatching security vehicles to incidents based on their locations. It is assumed that security officers might alternatively be dispatched on foot, so the ability to extrapolate current traffic conditions and estimate the time required to drive to an area of incident is valuable to security personnel. This may serve as a form of decision support, or may operate through foresight to prepare for possible traffic incidents. The benefit of using real-time sensor data was offset by the burden of supporting a legacy traffic simulator which was not designed to interface with our perennial framework. We demonstrate a method of encapsulating legacy components which allows them full access to the sensescape while requiring only minor modifications in the form of boilerplate code.

The incident response study combined many aspects of traditional simulation, such as a vehicular traffic model and a sensor capable of providing real-time data. The goals described previously are mostly specific to the perennial framework, particularly the task of handling legacy simulations. Therefore, this simulation exercise will play out much like a traditional traffic management simulation, rather than exploring an entirely novel concept such as MMOHILS. This was an intentional decision, and reflects the nature of "transitional" perennial simulation projects. As the perennial simulation framework slowly becomes commonplace, the typical simulation project will treat this framework more as an "add-on" rather than a centric element of methodology. Even given full acceptance, most studies will be much more prosaic than Figure 4.1 seems to imply. Such systems are nonetheless capable of capitalizing on the full benefits of the perennial simulation framework, and thus it is useful to view this and other transitional studies with this in mind.

5.2.1 Concise Overview

Responding to incidents on a small, congested road network with a highly variable capacity is exactly the kind of event that can suffer from a "conglomerate effect" —a buildup of smaller, non-critical factors that leads to catastrophic failure unless prompt action is taken [27]. Campus road networks tend to comprise a handful of arterial roads with a limited number of lanes. If construction or a traffic accident reduces capacity, then incident response cannot proceed with adequate efficiency, allowing a mostly benign event to worsen. Dispatching a security vehicle during peak congestion is ineffectual at best; in the worst case, an incorrect dispatch will cause an officer to become occupied attempting to service a request he cannot possibly reach. In the meantime, another (reachable) request may arise, stressing the typically limited resources of security response teams. Making the right decision based on the current state of the road network is crucial.

This combination of mundane incidents with a potential cascading effect is exactly the kind of scenario that can benefit from symbiotic decision support. When a situation arises, implementers can run several "What-If?" simulations to determine how best to respond: by vehicle or by foot. These simulations can capture difficulties such as congestion due to construction or poor weather conditions. Ultimately, the implementer must make the final decision, but providing her with a wide range of useful information about the current situation allows her to make the best possible decision given the current information. In addition, this decision can be validated later in a *hindsight*-focused study.

5.2.2 Perennial Components and Organization

Unlike the previous study, which was required to proceed without a complete framework, the incident response study had the benefit of a fully-functional framework implementation. This greatly eased development by clarifying the roadmap and allowing a clean separation of components. No historical artifacts were introduced by this study.

Perennial components fell broadly into two categories: those which would be part of any perennial simulation study, and those specifically required for legacy simulation. The real system comprised a physical world and its *Virtual Study* counterpart, with camera feeds used as data sources. A road network model was prepared using a mature traffic simulation package, and full "What-If?" capability was imparted through the use of perennial components. The main effector on the system was the implementers' ability to dispatch vehicles to response sites; thus, a form of symbiotic feedback was possible. As stated before, the incident response study featured a strong transitional element. Were the system to continue in development, one would expect most legacy components to eventually be rewritten as first-class perennial components so that full symbiotic decision support could be performed in a streamlined fashion. In this case, symbiotic triggers would include actual incidents, with the possibility to run *preventative* simulations to detect dangerous levels of congestion.

The following subsections will detail each component of the perennial simulation in turn.

5.2.2.1 Traffic System Components

Despite focusing on legacy system compatibility, our traffic system featured several components which interacted with the remainder of the system in a manner typical of perennial simulation elements. These include the world and sensors, as well as their associated history window and dependency tree.

Target	***Description***
Upper	North-west entrance, upper main road and small roundabout
Round	Large roundabout; connects upper, east, and south main roads
East	North-east entrance, east main road
South	Southern entrance, south main road, major traffic light intersection leading to south-east entrance
SEast	South-east entrance, south-east main road, intersection to two other campus locations (which are not modeled)

TABLE 5.2: Targets for the world "virtual.1".

Our traffic system consists of two worlds, "physical.1" and "virtual.1". The latter was a virtual study of the former, and contained five targets as described in Table 5.2. These targets cover several thoroughfares, intersections, entrances/exits, and roundabouts, and serve to completely describe the major traffic influences on campus. (Several minor but essential sources of traffic, such as shuttle buses and city transit, are modeled using schedules and route maps.) Figure 5.20 depicts each target represented as a combination of several sensors. These components will be described in more detail in a later section.

Additionally, the traffic study makes use of the history window and sensor dependency tree —in fact, this is the study which first introduced these two items as necessary components of the reference implementation. Figure 5.20 depicts a subset of the sensor dependency tree; the mechanics of these two elements were introduced in complete detail in Section 4.3.2.3.

5.2.2.2 Traffic system legacy model

The virtual road network was modeled using the "Simulation of Urban Mobility" (SUMO) package [101]. Running a predictive simulation in SUMO required a road network, a list of vehicle types, and a list of routes. The road network listed all **nodes** (intersections, lane splits and merges) and **edges** (paths between nodes) in a simple XML format. Rules on U-turns, number of lanes, lane priorities, and lane usage (e.g., car lanes versus bus lanes) are all specified as properties of edges or through the use of **connections** (specialization templates describing what happens when two edges meet). In addition, bus stops are represented by automatically segmenting an edge at a given point. Buses themselves require additional logic in the vehicle types list to make proper use of bus stop edges.

To obtain the aforelisted road network data, a variety of GIS data sources were combined, with Open Street Map providing the bulk of the road network [164]. Particularly tricky sections of the road network were sketched, paced, and modeled manually. Figure 5.20 presents just such an example: the roundabout connecting the campus's three major roadways, with cars represented in silver, taxis in blue, and shuttles in orange. Since SUMO expects traffic to flow on the right (and in Singapore the reverse is true), the road network was flipped about the Y-axis before being sent to the simulation engine. Care was taken to ensure that this did not lead to an invalid model; see Section 5.2.4 on validation. Only one intersection had a traffic signal; its timing was extracted from camera footage at that location.

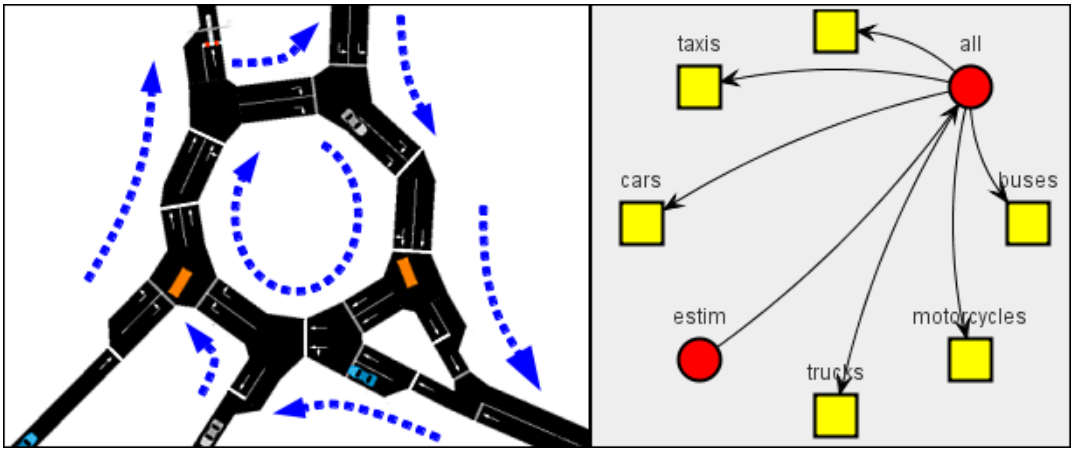

FIGURE 5.20: *Left*: The main roundabout on campus, with various vehicles navigating it. Dashed arrows indicate the direction of traffic flow. Small 2-lane segments complicate this location. *Right*: The roundabout camera's sensor dependency tree. Squares (colored yellow) represent "leaf" sensors, which must be processed first.

Vehicles were grouped into categories with similar properties. Cars, taxis, motorcycles, and trucks each had different driving behavior characteristics as defined in [99], which were found to be consistent with our observed traffic data. City buses and campus shuttles functioned as larger vehicles with the additional property of periodically stopping to pick up and release pedestrian passengers. A vehicle's behavior was modified by the driver's *imperfection* parameter; this is a common technique for introducing driver variability into traffic models.

The route list was generated each time a "What-If?" simulation was requested by the controller. As explained below, each simulation run could avail itself of a series of arrival times and route choices for each sensor location. Buses and shuttles were generated first, as they appeared on a regular schedule. For each bus or shuttle arrival, the nearest arrival time from the input data was tagged as being a public transport vehicle, and given a predefined route including bus stops. Then, the remaining entrance arrivals were tagged

as one of the remaining vehicle types. Each of these types was given a route based on the observed node choice probability at each intersection.

Before continuing to describe the system, it is worth reflecting for a moment on SUMO's role as a legacy simulation component. As previously mentioned, the SUMO traffic simulation toolkit was chosen over more prominent traffic simulation packages for a variety of reasons. First, Krauss's work is sound, rigorous, and fully validated. Second, it fits both requirements of a "legacy" model: it is not an agent-based model, and it cannot be run in real-time in discrete time-steps[4]. Therefore, Krauss's model is an ideal example of a high-quality legacy model that one might wish to incorporate into a perennial simulation. Using a model which is specifically incompatible as a first-class perennial element proves the point that our legacy integration approach works.

5.2.3 Introduction to Image Processing

Video footage from security cameras was mined for traffic data, using techniques borrowed from the fields of image processing and information retrieval. As this is an ancillary activity and tangential to the field of simulation, a brief introduction to *image information extraction* will be presented in this section. This should not be taken as a representative literature review, since it will only cover the specific technologies we chose for our implementation and a few prominent alternatives. The actual image processing pipeline used for information extraction will be detailed in Section 5.2.3.1.

Several independent steps are required to extract vehicle data from a series of video frames. Background elimination, edge detection, and object labeling are universal, although various intermediate filtering steps are also commonly used to refine the process [165]. The simplest approaches are **region** based, and perform well on low to medium levels of congestion. For heavy or overlapping traffic, a more powerful algorithm is required that tracks the **features** of an individual vehicle [166], or attempts to project 3-D models of existing vehicles onto the video footage and thus form a match [167]. Both of these approaches, while certainly more sophisticated, have their own caveats as well; the former is difficult to implement and time-intensive to tune, while the latter requires the modeler to have existing 3-D models of all potential vehicles in the system.

Early work in automated traffic-related image processing focused on identifying traffic violations and congestion, with little thought given to using the mined data as input into a real-time simulation. As the field matured, mined traffic data were used in a broader

[4]In fact, SUMO does use a discrete timer, but the dynamic user assignment strategies require multiple simulation runs to reach equilibrium, and thus cannot be performed in lock step with real time.

range of endeavors, especially automated signal control. A good example of this subdomain is [168], which studies the possibility of controlling traffic signals dynamically in response to actual traffic data. DDDAS is leveraged to predict the optimal traffic signal patterns in a series of complex intersections. The author implies that traffic cameras will provide the input data, but it is unclear how much of this is actually occurring in real-time. The study is notable for its emphasis on modern techniques such as DDDAS and the SOA; the latter was introduced in Section 2.1.1 as a means of achieving simulation interoperability. Unfortunately, modern studies will often suffer from a form of myopia: [169], for example, attempts to dynamically adjust traffic signals, but makes no appeal to real-time data or symbiotic response.

Our work relies heavily on the ViBe background elimination algorithm [170] and Perreault's median filter [171]. ViBe is notable for its performance, resistance to camera jitter, and modest requirement of only a single frame of startup data. Perreault's approach uses an array of past kernel values to compute each pixel's filtered value in constant time with respect to kernel size. Using these techniques, we chose to create our own region-based tracking algorithm. Region-based tracking seemed acceptable, as our traffic network experienced only moderate levels of congestion. Unlike most related work, our footage comes from security cameras, which are often badly positioned with respect to traffic. This presents a challenge to our object tracking algorithm, and ultimately puts a limit on the quality of the data that can be provided to our simulation.

5.2.3.1 Image Processing Pipeline

One week's worth of traffic footage, recorded daily from 3pm till 4pm, was processed to retrieve the arrival time, estimated velocity, and on-screen entrance of each vehicle that passed within range of each camera. Available to this algorithm was a list of entrance locations (in screen co-ordinates), the distance of each possible route through that location, and a mask image which was generated by hand to help reduce the impact of static noise sources, such as nearby highways, trees blowing in the breeze, or busy pedestrian crossings.

The image processor was implemented as a series of GStreamer plugins. GStreamer is a multimedia processing framework built around the idea of a software pipeline. Audio and video streams flow from **sources** through a series of mutative **nodes** until they reach a **sink** and terminate. This approach is highly flexible and, as we shall see later, capable of being parallelized.

The complete image processing pipeline, presented in Figure 5.21, is broken into four phases. First, the *input* phase reads a source MPEG file, removes the audio track, and

balances the video stream. Next, the *filter* phase applies a mask to each frame, followed by the ViBe background eliminator and finally Perreault's median filter. The filtered video stream is then duplicated. The *display* phase operates on one copy of the stream and simply presents the candidate objects to the user. Figure 5.22 shows a sample frame from the *display* phase. Finally, the *identification* phase converts the colorspace from YUV to RGB, and performs segmentation, object identification, and object tracking in that order. After each object exits the camera's field of view, its path and traversal information are saved to a flat file.

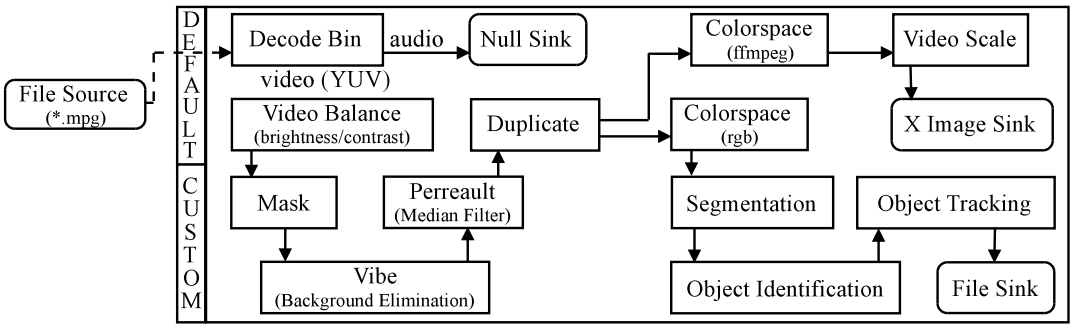

FIGURE 5.21: GStreamer pipeline for our image processing algorithm, with sources and sinks as rounded rectangles. "Default" plugins are provided by GStreamer; the "Custom" plugins were programmed as part of development.

The segmentation algorithm we used grouped together pixels which shared a common minimum threshold number of neighbor pixels. Object identification was performed by first checking if each segment's area was above some minimum value. If so, the segment was promoted to an object. If not, the segment was conditionally promoted depending on the histogram of color values it contained, under the assumption that vehicles tended to contain narrower histograms than other moving objects like pedestrians. Finally, the object tracking algorithm operated in two steps. First, new objects were detected by checking if they overlapped one of the entrance locations. Second, objects from the previous time-step were compared against objects in the current time-step by size and position; the nearest match became part of a "tracked" object. When a tracked object overlapped an exit location, the tracking of that object was considered complete.

When operating on real-world data, the system exhibited a few weaknesses. First, the traffic cameras had a tendency to shake in the wind, and were prone to temporary fluctuations in their histograms. ViBe was capable of filtering out either of these individually. If both occurred at the same time, however, they resulted in a small number of frames consisting solely of numerous fragmented artifacts. Second, a car's windows tended to be marked and eliminated along with the background, which would occasionally cause the segmentation algorithm to split a car into four or five separate pieces. Figure 5.22 depicts an example of this false segmentation; in this case, the object tracking algorithm

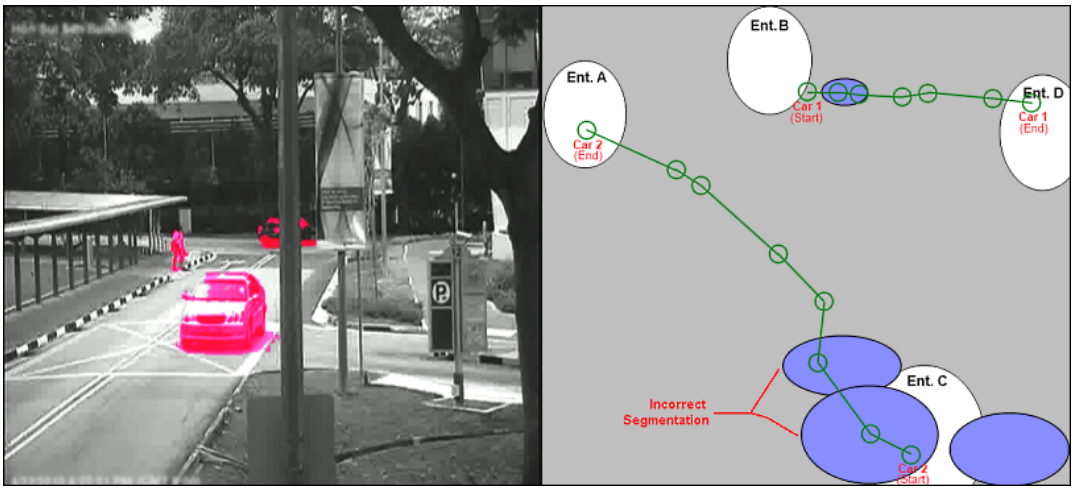

FIGURE 5.22: *Left*: Output of the GStreamer X Image Sink node; possible objects are highlighted in pink. A pedestrian (left) will be removed in the "object identification" phase. *Right*: Screenshot of image tracking system, showing a false segmentation: the two marked nodes are actually part of the same car.

was able to recover, but if the false segmentation continued for several frames, or if two cars overlapped during a timespan with high background noise, then the image tracking information would need to be manually corrected.

5.2.4 Verification, Validation, and Calibration

The incident response study required a great deal of manual verification effort. One potential source of uncertainty was the use of a right-side driving traffic simulator (SUMO) with a left-side driving road network. To this end, the output of several traffic runs at each congestion level was analyzed at one-second intervals to ensure that vehicles did not skip road segments, take abnormally long paths, or engage in other incorrect behavior. In addition, SUMO's capacity to operate at high levels of congestion was confirmed by checking that vehicles did not enter the road network until space was available for them, and that vehicles did not exhibit "collision-like" behavior at intersections. Many traffic simulators including SUMO introduce a collision-free driving model that force-brakes any vehicle which would otherwise engender a collision. This leads to rapid deceleration which is beyond the physical capacity of a real vehicle's braking system, and represents an error.

Another time-consuming area of verification involved the automated image processing algorithm used to extract traffic patterns. The potential for inaccuracy was first quantified automatically by counting the number of frames with fragmentation artifacts and classifying them by their severity. An *artifact* in this context refers to a segmented region

Classification	**Threshold**	**% Frames**
Negligible	$N_I \leq 5 \vee N_O = 0$	89.14%
Mild	$5 < N_I \leq 10 \wedge N_O \leq 2$	9.0%
Severe	$N_I > 10 \vee N_O > 2$	1.86%

TABLE 5.3: Frame artifact errors by category

	Category	**% Objects Affected**
Erroneous	Tracking	3.42%
	Split	11.12%
	Total	14.53%
Correct	Initially Correct	65.99%
	Recoverable Errors	19.48%
	Total	85.47%

TABLE 5.4: Percent of object identification errors by category

of one input frame which has corresponding region in the previous or following frames. Table 5.3 lists the percentage of frames with **negligible**, **mild**, and **severe** fragmentation, as well as defining each category. $\mathbf{N_I}$ and $\mathbf{N_O}$ in this table refer to the number of artifacts existing in an isolated area of the image versus overlapping a non-artifact, respectively.

Any frame with **mild** or **severe** artifact fragmentation could conceivably interfere with the detection of zero, some, or all of the objects which appeared in that frame. A sample frame with this issue is shown in Figure 5.22. Thus, although Table 5.3 provides a good approximation of the scope of the error in the system, extrapolating this data to arrive at the actual errors in object detection was impossible, and verification proceeded manually for the next phase. For each one-hour video stream, the number and types of vehicles tagged by the image processing algorithm was compared to a list enumerated by hand. We observed two primary types of errors. A **tracking** error occurred when an artifact was incorrectly identified as an object for one frame, thus affecting the perceived route of the vehicle through the recorded area. A **split** error occurred when an object was artificially split into several artifacts for one frame. In this case, the image processing algorithm was unable to detect that multiple artifacts should be aggregated to form one resultant object. Table 5.4 lists our results; in total, 85.47% of vehicles were correctly identified. In some case, a single-frame error was detected and corrected (by skipping that frame and interpolating the frames before and after); these do not count towards the error total and instead are listed as "recoverable". All errors were manually corrected prior to running the simulation.

Once it had been established that our image detection algorithm and mirroring of the

road network performed as intended, validation efforts could proceed. Estimations of input data, where applicable, were accurate with $p<0.01$. Vehicle inter-arrival time was exponential in nature, while intersection choice behavior tended to be uniform. To ensure that SUMO processed the road network accurately, a scenario matching the recorded data points was set up and run to completion. As expected, individual vehicles did not exactly match the recorded data at all time points, but overall congestion levels were closely matched on road segments which fell within the recorded zones.

Validation of the incident response zones described in Section 5.2.5 was also performed. Since we relied on existing vehicles rather than injecting patrol cars directly into the network, there was a risk of sparse data leading to inaccurate response radii. To check this, we ran a series of simulations that flooded the network, and generated a count of the maximum number of vehicles that could exist on each road segment. Then, we compared this number to the actual number of vehicles per segment during a simulation run, ensuring that roughly the same proportion of maximum capacity was utilized per road segment at any point in time. This provided a small but helpful level of validity in asserting that incident response zones were not adversely affected by outliers.

5.2.5 Structure of Experiments

Three areas were analyzed for congestion. **Region A** was centrally located near the main campus roundabout, while **Region B** and **Region C** were located near the edge of the campus. To alter congestion levels, the inverse of the mean inter-arrival time was modified by a constant factor at each entrance, while route choice probabilities at each node remained unchanged. We refer to this scaling of origin-destination pairs as a "congestion factor". Twenty experiments were run at each region, with a total of four congestion factors analyzed. For each of these experiments, an incident was introduced into the simulator at the same point in time.

When an incident entered the system, the controller immediately responded by spawning off a series of "What-If?" simulations with the goal of measuring the time it would take for a vehicle to respond to that scenario from any point on the campus road network. Approximately thirty of these simulations were required to reach an acceptable error threshold. As the traffic network continued to update during this period, it was essential that "What-If?" analysis concluded speedily. Thus, each simulation scenario was run in its own thread, and its input and output were encapsulated into a scenario file that allowed it to run on any computer in the local network. Fortunately, SUMO can process road networks fairly quickly, so a single multi-core machine was sufficient to complete the "What-If?" activity within a few minutes.

The "What-If?" experiments were then combined to form a "response radius" for each scenario. Any vehicle which could reach the area of incident within a given time limit was considered within the response radius. For example, Figure 5.23 depicts the average (plus variance) response radius around all three areas, with a graphical representation of **Region A**'s response time of 60s at a congestion level of 1.0. Based on our assumption that incidents will be responded to by security personnel who are currently on patrol, we do not inject cars into the simulation when an incident occurs. Rather, we simply track all existing vehicles throughout the time limit, and flag which ones pass through the area of incident. This creates an approximation of the response radius, and repeated simulations under similar conditions increase the accuracy of this approximation.

5.2.6 Discussion of Results

Figure 5.23 shows the effect of congestion on the response radius. Region A's response radius decreased slightly as congestion increased, but remained relatively unaffected overall. This is likely due to its central location near the intersection of three major roadways. Regions B and C each noticed a steady drop in response radius as congestion increased, reflecting the remoteness of these locations and their subsequent vulnerability to increased congestion. Region B's response radius plummeted under a congestion factor of 2.0, which is particularly troubling as 2.0 only represents a medium-high level of congestion. Recall that the default congestion factor of 1.0 was extracted from non-peak traffic patterns from 3pm till 4pm.

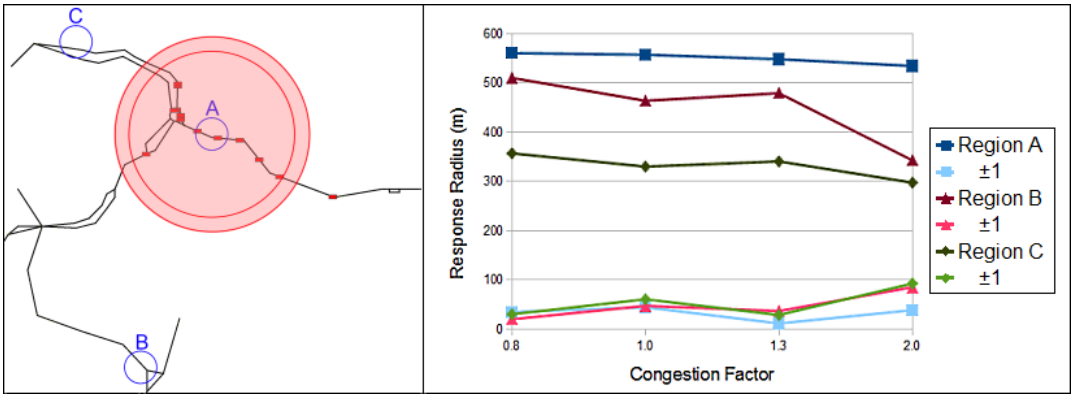

FIGURE 5.23: *Left*: Campus road network, with three regions marked A, B, and C. Region A's incident response radius and variance are depicted for congestion factor 1.0. *Right*: Mean response radius (± 1 standard deviation) for each region under different congestion factors (scaled O/D arrival rates).

The data gathered express their value in their applicability to various knowledge acquisition tasks. Predicting that a central location like A is easier to reach than the others is obvious, but discerning between Regions B and C (which both seemed equally

remote, but had different response radii) was a valuable addition to the pool of total information available to dispatch personnel. As the complexity of the road network increases, the ability to back up "obvious" statements with actual experimental evidence becomes essential. Regarding the image processing algorithm, performance was acceptable for real-time operation, and all of the image processing pipeline's shortcomings have straightforward workarounds which are discussed in [163]. Thus, it is reasonable to assume that it could run unattended in real time. This combined with the very nature of the perennial simulation framework would allow campus security administrators the ability to run "What-If?" simulations on the traffic network during any given period of congestion. This would be invaluable for training, where the presence of a simulation-based "arbiter" removes a substantial amount of guesswork [22].

In addition to the data gathered for reasoning about the road network, the integration of a legacy simulation into the perennial framework was also a marked success. The legacy model was cleanly incorporated into the framework through the use of inheritance, thus giving it access to all real-time and historical sensor information. For foresight or hindsight studies (including "What-If?" analyses), this was sufficient. Non-symbiotic decision support is possible with minor modifications to the image processing algorithm. The only feature unavailable would be real-time symbiotic decision support, and this is a limitation of legacy models in general rather than a weakness of the perennial framework. In the case of SUMO, for example, absolute positioning of cars is technically possible, but there is a risk that inserting cars directly might invalidate some of the modeling assumptions of the various traffic movement models. In addition, placing cars directly would require guessing their routes based on their previous locations. These problems are by no means intractable; rather, they serve to illustrate the boundaries that one invariably ends up pushing against when attempting to shoe-horn a legacy model into a modern framework. Were these limitations to be overcome, SUMO could exist as a first-class perennial simulation component.

5.2.7 Comparison to Non-Perennial Methods

The traffic incident system was further enhanced in an attempt to answer the question of how perennial simulation compares to traditional simulation-guided preparedness. As noted in Section 4.2.7, traditional simulation is the technology which fulfils the most similar role to perennial simulation without being encumbered by unnecessary restrictions (such as with symbiotic simulation or the HLA). We judged that perennial simulation's real-time decision support capabilities should provide the greatest benefit and thereby confer the strongest justification for the framework itself.

The basic structure of our comparison study was a simplified version of the incident study from Section 5.2.5. Traffic accumulates and propagates through a road network. At some point an incident occurs, and security personnel (playing the role of Implementers) must decide whether to dispatch officers in a vehicle or by foot. Walking is generally slower than driving, unless congestion is throttling the network. Thus, the ability to dispatch the correct mode of transport is critical to the officers arriving on time and pacifying the situation. For simplicity's sake, we estimated an arrival time of 240s as the maximum time an officer would take to arrive by foot. The Implementers are assumed to have a large amount of historical data on network congestion and its effect on trip time. Perennial simulation adds the ability for Implementers to perform runtime queries at the time of incident using up-to-the-minute road network data. The quality of information added by perennial simulation over traditional simulation is judged by its ability to avoid false positives (dispatch by foot when vehicles would have arrived on time) and true negatives (dispatching a vehicle that won't actually arrive on time).

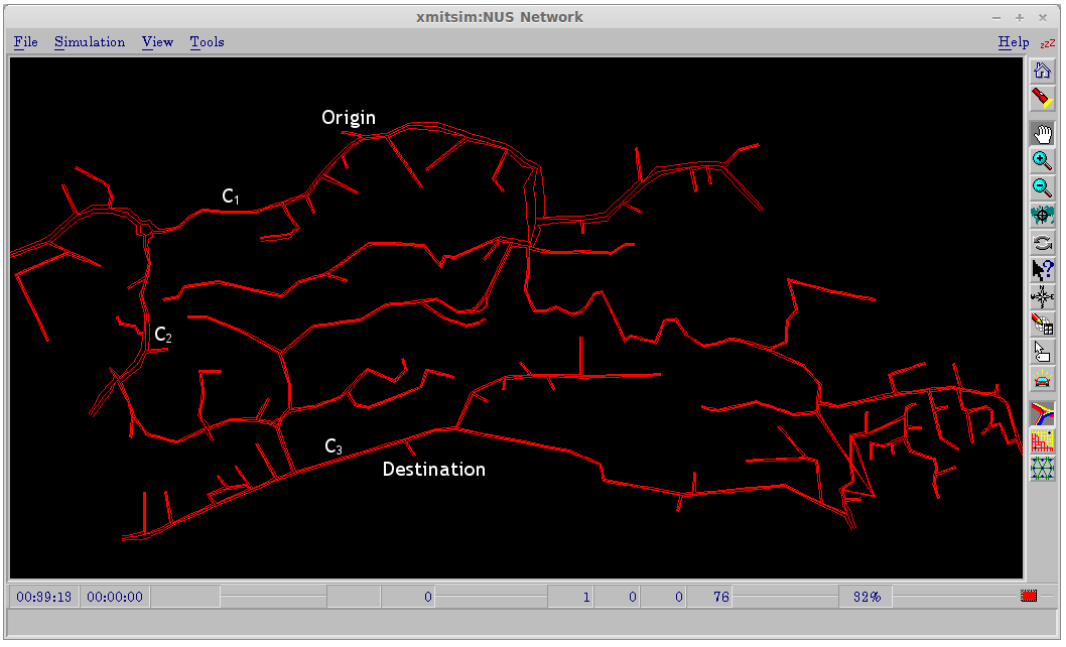

FIGURE 5.24: Screenshot of the enlarged area network running in the MITSIM X11 visualizer, rotated to 45°. Various key roads have been labeled.

The road network from our previous study was enlarged to better capture the cascading effect of congestion over time. This was accomplished using geo-spatial data retrieved from OpenStreetMap [164], with cleanup performed against a variety of third-party sources such as Google Maps. Figure 5.24 depicts our expanded network in the visualizer for MITSIM. This traffic simulator, which is part of the MITSIMLab simulation package discussed in Section 3.7, features a powerful microscopic movement model that

incorporates driver reaction time, allows individualistic route assignment, and has advanced traffic management system controls. MITSIM contains a much broader feature set than SUMO; thus, we considered it more appropriate for our task of performing predictive analyses in a complicated, realistic scenario. Traffic demand was increased proportionally for the origin and destination nodes which were present in the original study, while the remaining candidate nodes received a fraction of this value based on the given roadway's lane count and total length. MITSIM uses "congestion factors" to increase demand proportionally; demand was thus adjusted so that a congestion factor of 1.0 represented the mild traffic observed from security camera feeds. Congestion was increased up to a factor of 3.8, at which point traffic jams rendered the entire roadway unnavigable. Compared to our original study, several simplifying decisions were made. First, traffic lights were not modeled explicitly; instead, demand was decreased in timed intervals to account for their presence. Second, security vehicles were dispatched from a single origin to a single destination, in order to maintain consistent, comparable data on congestion levels. These locations are marked on Figure 5.24, in addition to the three roadways (C_1, C_2, and C_3) which present congestion information to Implementers. Third, security personnel moving by foot were advanced using a linear time movement model, assuming that such personnel would be unaffected by sidewalk crowds. Several distributions of officers' starting locations were simulated to determine the upper bound on arrival time of 240s, at which point this value was simply used as a threshold on the drivers' arrival times. Finally, the MITSIM road network required the same horizontal axis modification used earlier in Section 5.2.2.2.

Evaluating the benefits of perennial simulation required us to measure the effect of new information on **Implementers** at the time of an incident. From existing research, we know that Implementers would likely maintain their decision-making capabilities even through stressful situations [138]. In addition, we feel they are likely to respond positively to simulation-generated advice, based on our own research into symbiotic decision support as well as existing research such as [136] and [137]. Based on this, we chose to model the Implementers' role in this comparison study using **decision trees**. A decision tree is a directed, acyclic graph used to formalize concepts in decision-support analysis. Decision trees represent decision points and the criteria for moving between them using traditional graph vertices and edges. They are a useful tool for presenting decisions in a way that experts can grasp while still remaining readily accessible to programmers. As such, they typically manifest a high level of credibility, and ease validation of expert roles within a simulation study [172] [173]. Figure 5.25 depicts the decision tree used to represent the Implementers' decision-making process. At the time of incident, an Implementer must choose whether to dispatch security personnel by foot or by vehicle. The current congestion levels for two major roadways along the shortest

driving route are known to Implementers. These are expressed as vehicle density values C_1 and C_2, and are sufficient for determining the appropriate reaction. As shown in Figure 5.25, values of C_1 outside the range of 27 to 45 present only one option. For the remaining values, C_2 can completely describe a solution unless it is between 54 and 72. For the slim number of cases in which C_1 and C_2 fall within both ranges (which account for only 8.27% of our training data), an appeal is made to perennial simulation. Several "What-If?" scenarios are run, and are classified according to the summed squared difference between C_1 and C_2 in the current situation versus the "What-If?" analysis. The resultant travel times of the various "What-If?" scenarios are weighted by this sum, and are further scaled by a small amount to represent humans' tendency towards optimistic and emotional bias. Scenarios which predict success are weighted higher, while failure predictors receive less weight. The scenarios are then averaged to provide a general recommendation: either dispatch drivers ($P_S(D)$) or walkers ($P_S(W)$). The Implementers will follow this advice, unless it is unavailable ($P_S(\emptyset)$), in which case they will use the best estimate from historical data, and dispatch a driver if C_2 is less than 69 and a walker otherwise.

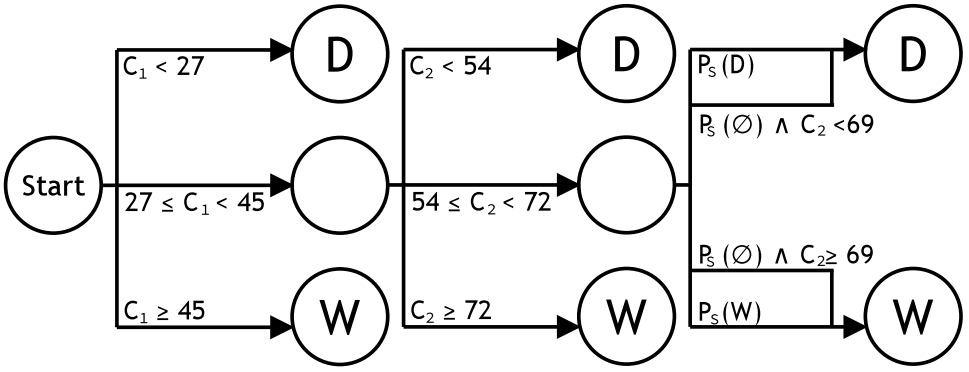

FIGURE 5.25: Decision tree representing incident response strategy on the larger road network. Nodes marked with a "D" or "W" indicate a decision to drive or walk to the location of incident.

The constants in Figure 5.25 were estimated from a set of 3000 vehicle traces at various congestion factors ranging from 1.0 (normal congestion) to 3.8 (very congested). For each vehicle trace, the values C_1, C_2, and C_3 were known, as was the total trip time in seconds. Although undesirable, false positives are considered less damaging than true negatives, as the latter leads to a situation where no security personnel reach the area of incident in time. As such, false positives received half the weight of true negatives when assessing the severity of an incorrect incident response. Figure 5.26 contains a plot of trip time for each congestion parameter, as well as for the combined sum of congestion, for reference. C_1 is a strong indicator of trip time, and C_2 is nearly as accurate. C_3 suffers from artifacts typical to MITSIM for short, straight road segments, leading to a

striated pattern. In addition, its predictive power is comparatively weak, most likely due to its location far downstream from the security vehicle's origin. Further analysis of the data led us to conclude that C_1 provides the most stable indicator of trip arrival time, with C_2 covering the areas where C_1 is not consistent. For some combinations of C_1 and C_2, it was impossible to predict success or failure with an accuracy 10% better than chance. These situations were expected to benefit the most from perennial simulation.

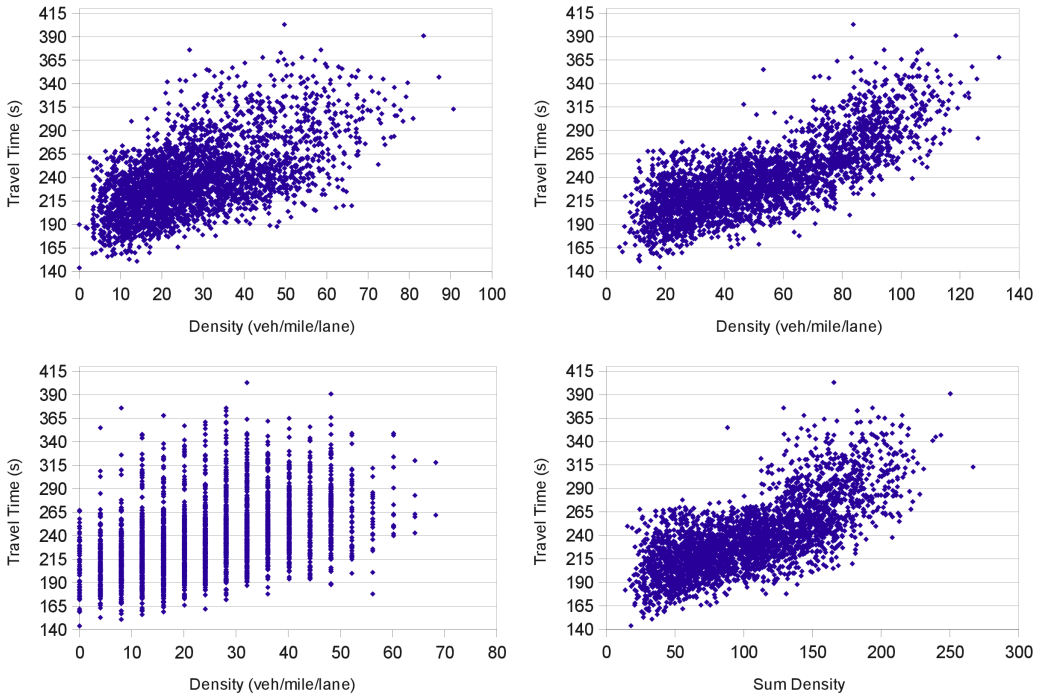

FIGURE 5.26: Data used to train the decision tree. From left to right, top to bottom: congestion parameters C_1, C_2, and C_3, as well as the sum total of all three parameters.

The data used to estimate the parameters in the Implementers' decision tree are assumed to be free from error, as any real-world application of such a system will be required to manually ensure that training data are correct. At the time of crisis, however, uncertainty will perturb the data presented to the Implementers and thus disrupt their ability to make accurate decisions. We assume that an error estimate will be known to the Implementers at runtime. This assumption is reasonable, since building a decision tree in the first place will require manually correcting data, so the amount of error can be estimated based on the error of the data that was cleaned for this purpose. For most scenarios, the error value will not perturb the congestion levels enough to affect the results of the decision tree. At times, however, multiple conflicting decisions will be presented to Implementers, who must then rely on their intuition to make the final decision. For traditional simulation, this means that they will choose to employ what worked well

in similar past scenarios. In a perennial environment, however, several "What-If?" simulations can be run, using a Monte Carlo sampling method across the possible values of each congestion parameter with the given error value. Once this is done, the scenario results are combined into a single recommendation (to dispatch drivers or walkers) which the Implementers will put into action. Regardless of their path through the decision tree, their choice will then be evaluated by the network simulator, to confirm whether or not their expectations matched reality.

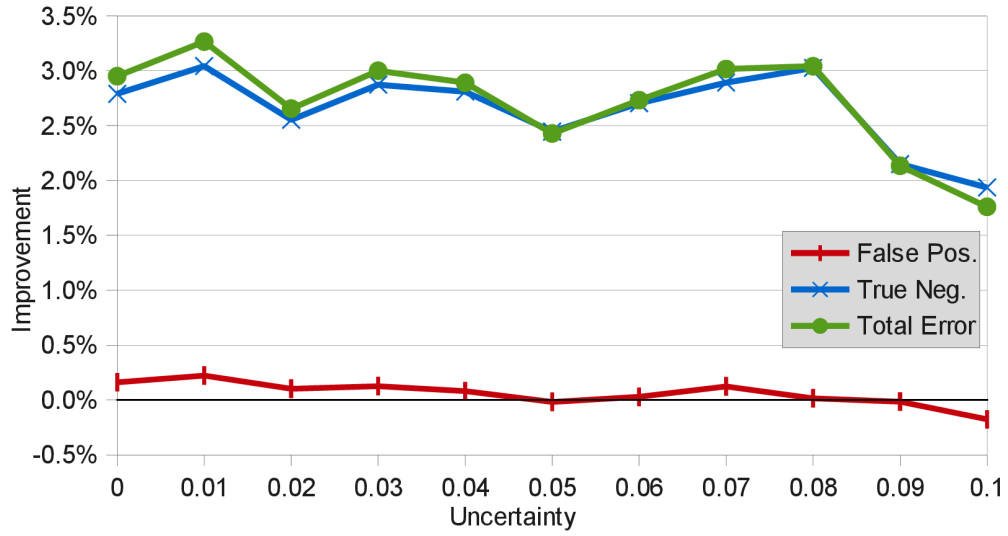

FIGURE 5.27: Improvement offered by perennial simulation in the *Control* scenarios.

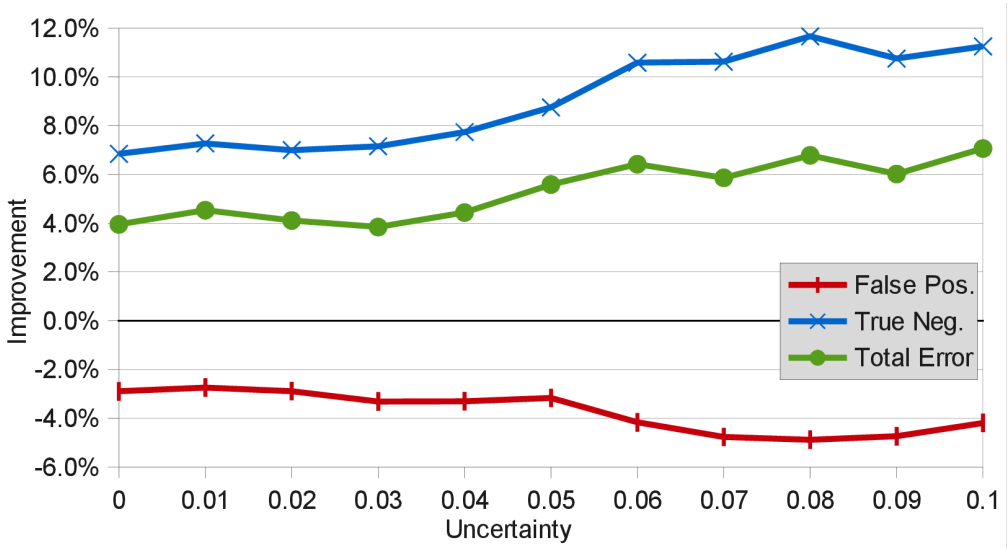

FIGURE 5.28: Improvement offered by perennial simulation in the *Testing* scenarios.

Figures 5.27 and 5.28 depict our results for the *Control* and *Testing* scenarios, respectively. Each scenario varied the uncertainty from 0.0 up to 0.1, distorting the reported values of C_1 and C_2 with uniform likelihood. If these two parameters led to a definitive

answer from the decision tree, they were placed in the *Control* category —and were not considered likely to benefit from perennial simulation. All other cases were expected to benefit, and were placed in the *Testing* category. 47,000 simulation traces of dispatched vehicles were sampled, and the decrease in false positives and true negatives were recorded, as shown in the graphs. In addition, the combined improvement is represented as a green line with circles, and is considered the most accurate measure of the perennial system's efficacy.

As shown in Figure 5.27, the *Control* group featured a general overall improvement and better true negative avoidance, both of which hovered around 1.8 to 3.2%. False positive avoidance generally remained positive, and had no real impact on the overall performance of the system. These results are typical of what one would expect with a control group —the use of perennial simulation in cases where it is not strictly needed does not offer a significant improvement over traditional simulation methods. As uncertainty increases, both systems experience the same degradation in predictive power (not shown in the graphs), so the benefit remains roughly unaffected. Incidentally, we were surprised to find that perennial simulation improves over traditional methods at all in this case. A 3% improvement does not justify the cost of setting up a perennial simulation, but if the system is already present for the *Testing* cases, then Implementers may take advantage of the system in mundane cases as well.

Figure 5.28 captures the essence of perennial simulation's improvement in the *Testing* group. Recall that entities in this group are necessarily difficult to reason about, even if information is perfect. In this case, perennial simulation incurred a slight penalty in terms of false positives, but countered this with a large improvement in true negative reduction. The overall trend line is positive, and increases from 4.0% to 7.2% as uncertainty increases. In other words, the perennial simulation is more than twice as good at reducing error rates than it was in the *Control* group, and this predictive power functions acceptably when faced with data uncertainty. It compensates by dispatching more vehicles than are strictly necessary, but as stated before, true negatives are sufficiently more damaging than false positive to justify this, and the overall trend line remains positive. Eventually, too much error will perturb the results and lead to far more false positives, dragging the trend line downwards. This is not shown, as an error rate of 0.1 is already considerably noisy, and simulation should generally not be relied upon if the input data does not satisfy a minimum required level of precision.

The nature of the improvement gained by perennial simulation merits further discussion. First, the *Control* group exhibited greatly improved control over true negatives, with no corresponding effect on false positives. This indicates that the decision tree used to

represent the Implementers' decision-making process could benefit from the incorporation of feedback from a perennial simulation. This reflects well on perennial simulation: even for clear-cut, seemingly-obvious decisions in a highly-correlated decision space, the application of perennial simulation techniques allows one to glean slightly more information and make an overall better decision. Next, we consider the trend of worsening false positives in return for better true negatives and overall better predictive performance. In this case, the weighting factors chosen for positive and negative information were determined statically, mostly because the Implementers' reasoning process was necessarily abstracted. However, in a real system perennial simulation offers Implementers the ability to *reason* about the consequences of their actions and change their partial utility metrics dynamically. For example, in a high-risk security scenario, the Implementer may choose to ascribe no cost to false positives up to a given threshold, in which case our system could reasonably be expected to boost performance by 12 to 15% or more with only minor tuning. The ability to incorporate Implementers in the loop is one of the key strengths of perennial simulation. Finally, we would stress that, even though the results obtained were clearly positive, the abundance of perennial simulation's benefit is difficult to accurately quantify. Our attempt with these scenarios was to try to compare perennial simulation to the "next best thing" available to crisis managers, and to that extend these experiments succeeded. However, a gain in predictive performance is only a small portion of the true potential benefit that perennial simulation has to offer. Future work may yield better quantifiers, but it would be unwise to depreciate the qualitative aspects of the framework.

5.2.8 Significance and Conclusions

The perennial simulation framework was again leveraged to provide increased modeling flexibility, this time for an incident response study carried out on a traffic network of a university campus. Similar to the previous example, the cost of development was slightly higher using the perennial framework, but the time spent was far outweighed by the benefit in creating opportunities for hindsight and foresight, as well as easy management of "What-If?" analyses. The model chosen for this study was a "legacy model" not originally designed to work within a perennial context, yet this limitation was partially overcome using a very simple inheritance abstraction.

The benefits of enabling legacy interaction with a perennial system were manifold. The legacy model now was able to access the complex network of sensors that the perennial framework organizes, which simplified the reuse of historical input data. In addition, the incorporation of real-time video data was now possible, although in practice our image processing algorithm lacked the required precision for this. Interaction between the

physical world and its *virtual study* equivalent was clearly delineated and presented no conceptual obstacles. Unfortunately, legacy models cannot directly be used for symbiotic simulation within the perennial framework. By definition, their lack of agents and a discrete time step complicates their application to decision-support tasks. However, if a non-legacy model were to be connected to the system, that model could benefit from spawning "What-If?" analyses using the traffic legacy model for cross-validation. This is the surest way to enable symbiotic decision support, thus striking a balance between accuracy and development cost. One final benefit was noticed during the design stage: the system was significantly easier both to plan and develop, since the perennial framework handled most of the difficulty of managing sensors and worlds.

The traffic study was further expanded to provide a comparison between perennial simulation and the most similar existing technique: traditional simulation. The decision-making process of the Implementers was modeled, and and increase in predictive power was recorded for the perennial scenario. This predictive power functioned well regardless of the uncertainty level, and even provided some benefit to seemingly-trivial cases in the control group as well.

The traffic simulation study we performed demonstrated the tradeoffs between high congestion and response to security incidents. The system designed for this task used traffic data extracted from cameras, which leads to the possibility of estimating a response radius during the event of an actual incident. Overall, our approach for automatically extracting traffic data from an existing network of security cameras was a success. Manual processing of camera data was required, but most of this was due to minor issues that the system could, in time, be tuned to automatically avoid. Processing could be performed efficiently, and using a pipelined system opened up the possibility of parallelizing each processing step, most likely leading to a boost in performance.

5.3 Building Monitor Prototype

The third study carried out with the perennial simulation framework was a prototypical application of an *automatic building monitor* system (or "smart building") and its use to aid full building sweeps in the event of a crisis [174]. Using "sweep teams" to search every room in a building is a common tactic when all building occupants must be informed of the need to evacuate. These teams are often forced to operate under tight time constraints, and must rely on limited two-way radio communication and even simple heuristics such as closing doors once rooms have been cleared [175]. Recent advances in sensor network technology allow easy monitoring of an entire building through communicating **motes**; the **Indriya** sensor network testbed is an example of just such a

system [176]. We targeted this existing physical system to test the construction of a prototype that could leverage the perennial simulation framework to provide an abundance of information to sweep teams tasked with notifying the entire building's occupants.

Unlike the previous two studies, which also served as classical simulation studies in their own right, the building monitor study was intended to function as a prototype application of our perennial framework with actual real-world systems in mind. In other words, the goal of this study was to explore the creation of the system itself, rather than to investigate some ulterior prospect (such as the benefit of symbiotic advice or the use of legacy systems). At the same time, where previous studies assumed perfect knowledge as a means of advancing to the actual experiments as quickly as possible, this study specifically bound itself to existing, working technology. A virtual world was still used to ease development, but the goal was that, by mirroring a real building, the modifications necessary to realize this system physically would be simple and straightforward.

5.3.1 Concise Overview

Interior environments such as buildings offer a unique challenge to crisis management, as they feature both dense populations and relative isolation between groups. This risks reducing the effect of intervention strategies, as many crisis management tasks rely on informing as many people as possible of some key notification. The obvious example would be a notification to evacuate, but as the first experiment involved the task of egress we will focus instead on another facet of building notification. Consider zoned quarantine, a form of social distancing that can work to limit the spread of infectious diseases [177] [4]. Research data indicate that, generally speaking, quarantines are more effective if enacted early, and then only if an overwhelming the majority of individuals adhere to the restrictions imposed by the quarantine [178]. Related research confirms that people are more likely to follow advice in a crisis if it is given by a human authority figure [43]. Thus, using teams of notifiers who physically patrol the entire building informing occupants of the intent to quarantine maximizes the chances that such orders will be received, understood, and followed by all. Despite this benefit, notification teams present their own challenges, mostly due to the time-intensive nature of building sweeps. Personnel posted at building entrances can inform anyone wishing to enter the building, but this will not prevent occupants already inside from blithely changing locations within the building. In such cases, the notification teams must backtrack to ensure that as many people are informed as possible. As the number of buildings or layout complexity of each increases, this task will quickly overwhelm the human capacity for planning.

To deal with the complexities of this task, we propose using a network of low fidelity sensors to detect building occupant presence in real-time, coupled with "What-If?" simulations to optimize the dispatch of notification teams throughout the building. Various sensing nodes in the building's sensor network will relay readings which can be combined to form an overall picture of the building's current occupancy. Small teams will be assigned routes to trace through the building, informing occupants on a per-room basis. If a room's sensors detect unexpected activity, nearby teams will either be rerouted or split and sent to investigate the potential source of new occupants to notify. This symbiotic feedback loop will ensure that as many people are notified as possible, while minimizing the time required to notify the entire building.

5.3.2 Perennial Components and Organization

The elements of perennial simulation included a variety of sensors, models, and simulations. The target environment was the **Indriya** sensor network testbed, an experimental sensor network under development at the National University of Singapore (NUS). The building monitor prototype was targeted at a *virtual study* of this environment, which would allow swapping of physical sensors with their virtual equivalents, and the introduction of estimations for rooms with no sensors. Thus, the concept of a virtual study proves its viability for a third time, as it enabled rapid construction of a prototype while still remaining true to the original system.

The **Indriya** network operates across three floors of the **COM1** building at NUS. We chose to focus on the first floor of the building, because of its greater sensor density compared to the basement, and its more interesting geometry compared to the second floor. The layout of this floor, marked with the locations of SBT80 sensing nodes, is depicted in Figure 5.29. The most interesting area is the winding thoroughfare which extends from the entrance past the Technical Services Desk to the back staircase. Sensors for this room were simulated individually and used to build a picture of the room's occupancy at any given time. This combined indication of a room's total occupancy is a type of *agglomerated data*, which was first introduced in Chapter 4. Such information exists as the property of its own unique sensor, which can be loaded from history independent of its constituent sensors as a means of increasing performance. This is exactly what we did for the cluster of sensors (61, 62, and 65) near the south-east side of the building. Finally, several estimation sensors were created for the rooms with no sensors, which may or may not have contained Telos B motes; these rooms are shaded in Figure 5.29. The **Indriya** sensor network is under continuous expansion[5], so it seemed reasonable

[5]The latest update increased the sensor count from 127 to 139 nodes; however, at the time of publication, the corresponding documentation had not been similarly updated.

to assume that more rooms would receive presence-sensing SBT80 boards in the future. In the event that this does not happen, the current presence estimators could easily be removed from the sensescape.

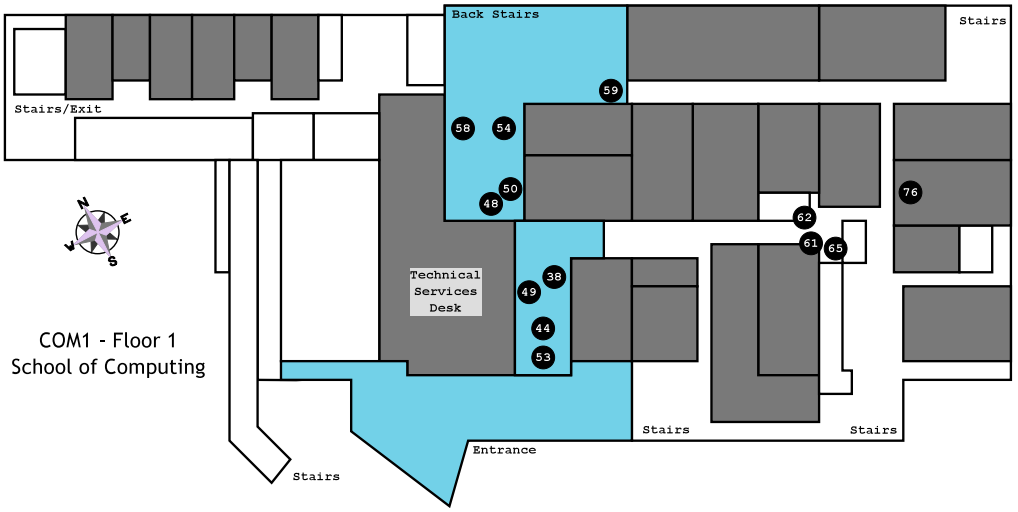

FIGURE 5.29: Layout of sensors in COM1, floor 1. Not shown are non-sensing nodes, which are used for processing or passing data through the sensor network.

5.3.2.1 Real System

Several different categories of sensors are used to form the overall picture of building occupancy; they will now be described in turn. The most logical place to start is with the Telos B motes which form the bulk of the **Indriya** network's sensing capacity.

The sensor network is composed of 127 *Telos B* motes connected through the *TinyOS* operating system. Each mote may have an attached sensor: either a *Wi-Eye* (infrared) sensor or an SBT80 multi-modal sensor. For the purpose of this study, we chose to focus on the SBT80 sensors, which offered a more interesting opportunity to combine sensor inputs, and were also more numerous within the **Indriya** framework. Figure 5.29 depicts the locations of Telos B motes with SBT80 sensors within the first floor of the target building.

The SBT80 sensors provide output for six distinct sensor types [179]. Table 5.5 lists these types along with each sensor's output range and effective sensing range. Internally, each sensor outputs a value between 0 and 4096, which sets a bound on the precision one can expect from these sensors. Sensors can update at 30 ms intervals, although propagation through the sensor network itself will limit this frequency. Data were captured from the following four sensors. The visual light sensor can detect electromagnetic radiation with a wavelength of 350 to 750 nm, and tended to vary between its maximum and

Type	***Axes***	***Units***	***Range***	***Used***
Visual Light (VL)	1	A/W	[0.0, +0.3]	Yes
Infrared (IR)	1	A/W	[0.0, +0.6]	Yes
Acoustic (MIC)	1	dB	[-45, -39]	Yes
Temperature (TEMP)	1	°C	[-25, +150]	Yes
Magnetometer (MAG)	2	gauss	[-6, +6]	No
Accelerometer (ACC)	2	g	[-1.5, +1.5]	No

TABLE 5.5: Properties of the SBT80 board's individual sensors

minimum value depending on whether the building's lights were on or off. The passive infrared sensor can detect wavelengths between 700 and 1100 nm, and was affected by the number of people in the room. Microphone sensors picked up various sources of noise, and the temperature sensor was generally unaffected by the occupancy level of the main room. Some of the noise may have been due to the sensors being situated near the ceiling. The remaining two sensors —the magnetometer and accelerometer— were not deemed useful for sensing occupants. Existing research often uses the former in outdoor environments to detect vehicles, while the latter might have some interesting application in determining building health during, e.g., an earthquake. Overall, though, the relevance of these two sensors toward human occupancy sensing was negligible.

Several publicly available sets of SBT80 sensor data were analyzed to determine how they might be used to detect occupants. Figure 5.30 depicts sample data for our four sensors of interest, with a 200 ms centrally moving average used to smooth out noise. We then defined a **presence sensor**, with dependencies on each sub-sensor, which flagged room occupancy using the following set of rules.

1. If the visible light sensor rose sharply from its minimum to maximum values, this strongly indicated the presence of new users.
2. If the passive infrared sensor detected a value near its maximum, this strongly indicated presence.
3. If the microphone's average rose more than 20% above its history-weighted average, this weakly indicated presence.

Each of the three conditions listed above provided a unit of confidence, with the microphone providing only half a unit. Following this, we defined a **combined presence sensor** for each room based on the combination of SBT80 confidence values. For each item listed above, its detection by any sensor in the room counted towards its unit of presence. If more than one sensor detected the same component, the unit of presence

was still only counted once. If the final presence value was above a threshold of 1.5, presence was detected. This is a simplification of an approach listed in [180], and allows multiple sensors to collaborate and cover each other's blind spots.

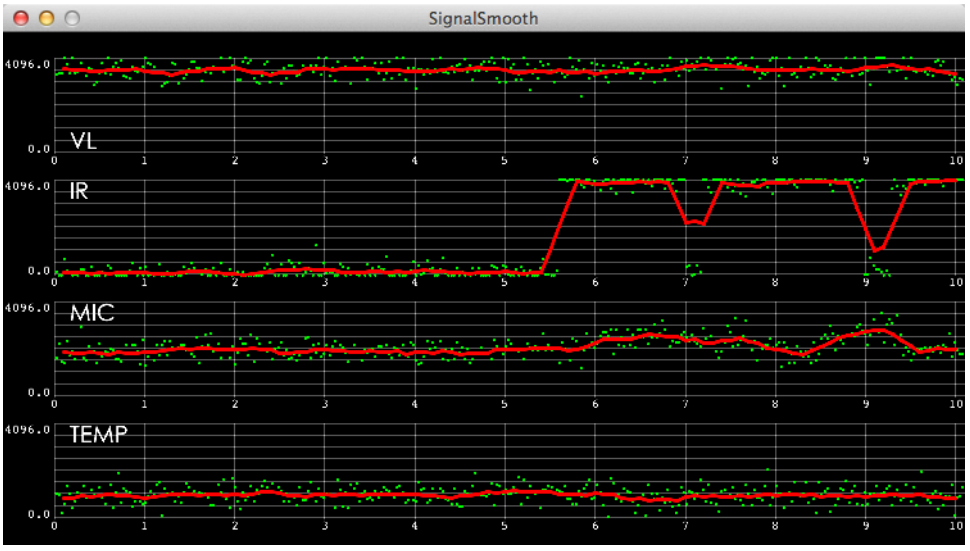

FIGURE 5.30: Sample data points received through socket sensors; the red line represents a 200ms centrally moving average applied as a smoothing function.

According to the requirements of our perennial framework, each sensor type had an associated historical sensor which could be used to access past data. This was particularly useful for groups of sensors which were positioned in under-utilized corridors, such as 61, 62, and 65 in Figure 5.29. For these sensors, we were able to simply load the combined historical sensor, and avoid the need to re-calculate the combined effect at each time tick. In addition to this, we defined an **estimated room presence sensor** which could provide real-time or historical data in the place of a **combined presence sensor** where such a sensor did not exist. This was based on an optimistic assumption that the rooms **Indriya** currently does not sense are excluded primarily for privacy reasons, and that the success of our virtual study might open the possibility of more room-specific nodes like number 76.

In addition to sensing occupants, simulations would require locational data on the teams themselves. This was essential to avoid a feedback loop, with notification teams triggering, e.g., the passive infrared sensors and the system being unable to detect that this was due to the teams moving throughout the building. From the simulation's point of view, the team sensing infrastructure need only operate at the rough granularity of a room. In other words, as parts of the building became notified, the only information required by the simulation was the identifier of the team which notified each room, and the point in time that room was notified. This could be achieved through the building's omnipresent WiFi network or, in the simplest case, through physical coordination using

two-way radios or mobile phones. Given these possibilities, our system assumed that team locational data were available.

The building monitor prototype contained one major class of effectors: the team notification strategies. This comprised the routes that should be taken by each of the teams to minimize total notification time, and could be relayed to each team through whatever communication channel was used for updating locational data (i.e., WiFi or mobile phones/radios). Unlike the library egress study, we had little cause to worry about teams disregarding information, since these were trained response teams instead of arbitrary individuals. There is some research which shows that even diligent, highly trained teams will break down under conditions of extraordinary stress, but for notification and egress this situation does not occur [139].

5.3.2.2 Practical Sensor Considerations

As part of our prototype study, we were required to consider the possibilities for collecting data from the physical system as well as its virtual counterpart. Although the **Indriya** sensor network uses Java for class descriptors, the nodes themselves run *TinyOS*, and it is not possible to communicate with them in a tightly coupled manner. Instead, there are three possible ways to gather data from the network:

1. Data may be read in bulk after the program had completed running on the network.
2. Data may be read at runtime through a TCP connection to the mote's *serial forwarder*.
3. Data may read from the MySQL database which stores data as requested by the program. The update frequency of data in this database is not entirely clear.

As our system dealt primarily with playback of SBT80 historical data, either the first or third item was sufficient for the virtual component of our simulation. That said, we designed the system with real-time operation in mind, so some strategy for extracting sensor readings as they occurred was needed. Here, the perennial reference implementation's **socket sensors** proved extremely valuable, as they logically separated the raw sensor data from the method by which it was provided. In particular, by relying on the second data access pattern, we could write a small utility in C++ which connected to the mote's TCP port and pushed data to a Java **socket sensor**. If the third data access pattern's update frequency were high enough, we could also write a small program in, say, PHP which constantly checked the database and pushed any new data to

a Java **socket sensor**. Using this clean separation, the source of any particular datastream need not be known to the simulation processing it. Moreover, *every* sensor in this prototype utilized sockets; even the combined sensors presented their data via **socket sensors**. This proved useful later, for hot-swapping historical sensors out for estimators and vice-versa.

We were fortunate that the **Indriya** sensor network provided a host of convenient methods for accessing data. Careful readers will note that we could have even written a TCP socket reader in Java, and bound the retrieval of sensor data strongly to our implementation. However, the benefit of the loose coupling afforded by the **socket sensor** approach is that it can be flexibly adapted to less forgiving sensor network setups —as long as there is some way of reading data from the motes in real time, that data can simply be pushed to a socket for the simulation component to deal with. This requires minimal boilerplate code to set up, and no modifications to the existing hardware.

5.3.2.3 Models and Simulations

The vast quantities of sensor data captured by the building monitor prototype were then processed by several models (which were in turn used by a variety of simulations). One of these has already been covered: the **combined presence sensor** mentioned previously functions as a coarse model of presence within a room. Several other models were made available to the system; these will now be covered in turn.

A room occupancy model was used to turn the concept of presence into an estimate of the number of people currently occupying a room. This model comprised two parts. First, for any given room, a maximum occupancy value was obtained. This was simply calculated from the room's floorspace combined with recommendations from Singapore's Fire Safety Codes for public buildings [181]. Following that, the model was also tasked with determining the implications of presence. For rooms with actual sensors, a positive presence value was correlated with a 90% occupancy value. This was done to ensure that sweep teams were always re-routed to deal with actual observances of occupancy. For rooms with estimated sensors, the desirability of a room waxes and wanes over time (either in the shape of a sine wave, or according to a pre-defined schedule), and occupants will leave or enter a room based on that desirability value. This was done to approximate room usage over the course of a given day, and provided implementers a natural abstraction for modifying the system. In both cases, the presence of a sweep team would cause a room's occupancy level to deviate from the physical system and virtually plummet. This was used to signify that the notification teams were getting their message across to occupants.

In addition to occupancy, the detailed path of egress for each user was also modeled. As sweep teams proceeded through the building, they delivered a notification which *may* have also included an order to evacuate. In this case, the desirability of a room was set to zero, and the users contained in that room were told to find egress routes. Modeling user egress was necessary, as users exiting the building would trigger presence sensors on their way out. We suppressed presence values if users could potentially be exiting through that location, so accurate estimates of egress route and time were needed. For this, we made use of **EvacNET**; see Section 5.1.2.2 for an overview of its syntax. Our previous work on the library egress study indicated that EvacNET calculated agent egress rates which were significantly faster than reality; thus, we slowed agent egress times generated from EvacNET by a factor of 2. Figure 5.31 depicts the EvacNET diagram for the first floor of COM1. The full specification is listed in Appendix C.

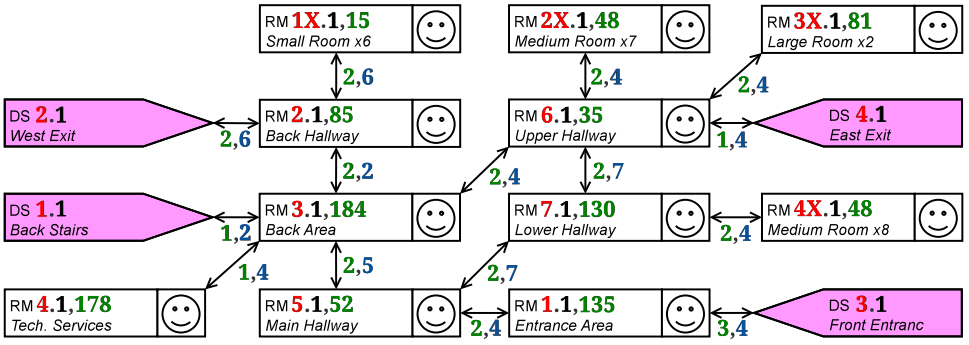

FIGURE 5.31: EvacNET model of the COM1 target world. The "1X" rooms represent a set of rooms with the same properties; e.g., 11.1, 12.1, etc.

The final model handled team movement throughout the building. This model performed estimates of the time required to search and notify a room, as well as the time required to move from room to room. EvacNET had no bearing on these calculations, as the teams were not engaging in straightforward egress, but were instead reacting to building occupants as they encountered them. Teams were expected to notify occupants as they encountered them in hallways, and to answer any questions the occupants might have. Equation 5.1 captures the general form of this model. Table 5.6 explains these parameters, and lists the distributions we attached to each independent variable. Note that this calculation could be performed piecemeal if the time required for part of a notification team's route was required. Also note that normal distributions were always bounded to be non-negative.

Parameter	***Description***	***Value***
T_{Notify}	Time to notify entire route	Derived
$T_{Hallway}$	Time to notify a hallway	Derived
T_{Room}	Time to notify a room	Derived
T_{Delay}	Time cost (s) of a generic delay	30
P_{Delay}	Probability of a generic delay occurring	NORMAL(0.05, 0.05)
TT	Walkway traversal time for a given room	From EvacNET
TG	Time cost (s) to notify a random group	20
LG	Probability of encountering a random group	NORMAL(0.2, 0.1)
NC	Node capacity for a given room	From EvacNET
PC	Percent of capacity currently utilized	From sensors
LQ	Probability of being asked a question while informing a room	NORMAL(0.5, 0.2)
TQ	Time required (s) to answer a question in an arbitrary room	NORMAL(30, 30)

TABLE 5.6: Parameters of the notification model

$$
\begin{aligned}
T_{Notify} &= \sum T_{Hallway} + \sum T_{Room} + T_{Delay} * P_{Delay} \\
T_{Hallway} &= TT + TG * (1.0 + LG) \\
T_{Room} &= NC * PC * LQ * TQ
\end{aligned} \tag{5.1}
$$

These models were combined to form two simulations. First, the constantly-running simulation tracked the team locations and the occupancy levels of each building. This ran in real time, and was scanned periodically by the controller in case symbiotic simulation was required. Second, the "What-If?" analyses simulated the teams' progression through the building and the general state of the notification exercise. These were triggered by the controller for various reasons discussed in Section 5.3.3, and their output data were collected and used to globally optimize the notification strategies of the sweep teams. As with all perennial simulations, the constantly-running models have uses outside the field of crisis management. For example, one might determine which meeting rooms were currently free based on the output of the presence sensors, and broadcast this information in a publicly accessible manner.

5.3.2.4 Usage as a Reduced Framework Tutorial

One of our goals was to use this prototype to introduce the framework and its implementation to new users. This was one of the reasons that a physical location was favored over an abstract virtual one. With this goal in mind, we refactored the building monitor prototype into a simplified version which serves to describe how one might use our framework to build a simulation. The full specification is included in Appendix D.

5.3.2.5 Mixing Human and Software Agents

Another goal with this prototype was to explore the possibility of incorporating actual human users into the software agent mix. We covered this concept briefly in [135], but at that time the framework had not yet been fully formed. Here, we consider the requirements of what we term "agent form transparency". For an agent to exist in either **form** (human or software), the human users must have access to the same input and output mechanisms as the software agents, and may *not* have further means of interacting with the system or each other. For input, this includes all sensors applicable to an agent, as well as the agent-to-agent sensescape. For output, this includes effectors (although agents rarely access these directly), environment interaction protocols, and agent-to-agent communication techniques —both internal to the simulation and external to it. If all of these conditions match exactly, then the potential exists for validly mixing human and software agents. The library egress study violated this constraint by allowing MMOHILS participants to speak to each other during the study; this led to them generating information which could not possibly be captured by the simulation itself.

For the building monitor prototype, agent input and output were kept clean of interference, and cross-agent communication channels were restricted. The motivating use case behind this effort was the notion of allowing *historical walkthroughs* to be performed of past egress events. In this case, our flexible sensor setup allowed us to rewind and replay historical events. By making use of agent form transparency, we were able to inject agents into the historical event to determine how it would have affected total egress times. In reality, these could be software agents with a pre-defined role (usually an additional sweep team), or they could be human users interacting with the system through a MMOHILS-like interface. Figure 5.32, shown in Section 5.3.4, depicts a mix of software agents and human agents performing a historical walkthrough. Unlike traditional MMOHILS users, which exist primarily as points of data, a participant in a walkthrough is generally attempting to explore possible options in a previous event. This can be very

useful for credibility purposes; i.e., to allow Implementers to experience and attempt to modify the course of the past event themselves.

5.3.3 Symbiotic Optimization

During a sweep team exercise, the perennial simulation within the building monitor prototype will periodically trigger symbiotic feedback to optimize the total notification time. This is a form of *preventative* symbiotic simulation, and is necessary due to the inherently unpredictable nature of building sweeps. For example, although an order to evacuate may be imparted to occupants quickly, an order for quarantine could occasionally give rise to lengthy questions about the exact limits of the quarantine. Thus, the Controller will periodically run the team movement model and raise a flag if the current teams' progressions will not lead to egress in the previously calculated time. At this point, multiple "What-If?" simulations are run, which attempt to determine how best to distribute the slower teams' workload to the faster teams. If a substantial speedup can be obtained, this new information is then broadcast to the appropriate sweep teams.

Additionally, the perennial simulation performs *reactive* symbiotic simulation if it detects occupants in an area of the building that previously was thought to be fully unoccupied. In a non-egress context, occupants never actually leave their rooms, so "unoccupied" has a much looser meaning: reactive simulation is triggered only if a fully empty room now has occupants. In addition, there is some leeway with regards to the threshold for actually triggering "What-If?" analyses. In particular, if a sweep team is known to be within visible range of the newly detected occupants, then no symbiotic feedback was deemed necessary.

Team dispatch follows a very simple set of rules: initial dispatch teams comprise either two or four members, with one leader per pair of individuals, and a combined leader for four-person teams. The four-person team may be split into two teams of two either by the combined leader, or as advice from the symbiotic simulation. In both cases, teams are only split as a last resort, to manage some unavoidable delay. Total building notification time is used to determine the value of each "What-If?" simulation, and the best plan is sent to each team via their respective communication devices. This setup provides flexibility without sacrificing speed, and allows the simulation to provide useful advice to the sweep teams without micro-managing them more than absolutely necessary.

5.3.4 Visualization Elements

Several visualization elements were required to support the activities of the implementers. The implementers in our building monitor prototype had a small but far-reaching role. They could override the sweep team notification strategy selected by the controller, and they could blacklist sensors in the event of erroneous results. (The **Indriya** sensor network generally did not have rogue sensors, but the ability to blacklist provided a safeguard against Byzantine behavior.) In addition, implementers could choose to manually modify a room's desirability level, allowing them to redirect teams in the event that knowledge of occupancy reached them via some alternative channel.

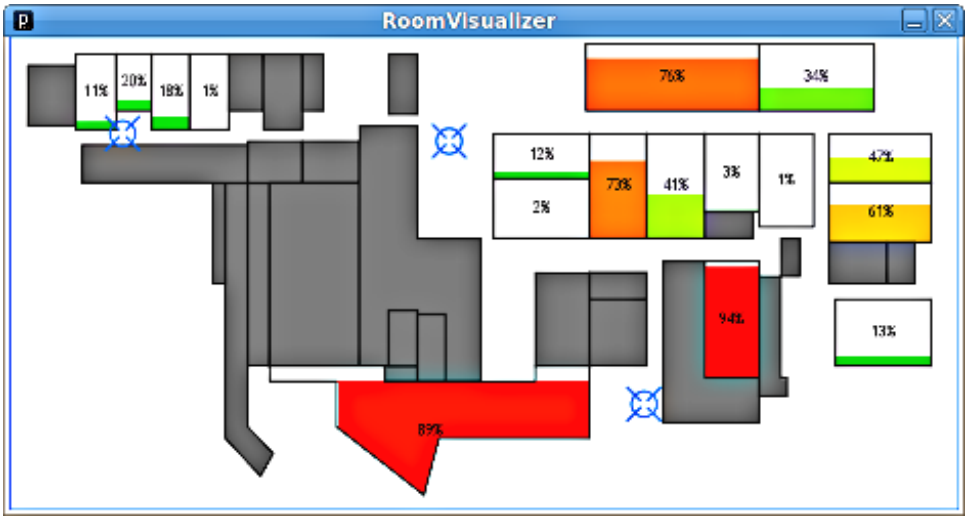

FIGURE 5.32: Screenshot of the building visualization during runtime. Teams are represented with blue circles; they may be software agents or actual humans undertaking a historical walkthrough.

Given the responsibilities of the implementers, combined with the description of sensors and agents, our system defined several visualization elements. First, the *building visualization* depicted a floorplan of the world with team positions and room occupancy values at any point in time. Second, the *strategy visualization* listed the most promising plans generated by the controller through "What-If?" analysis, along with the estimated total notification time and (if relevant) evacuation time. Finally, the *statistics visualization* operated similarly to a more classical simulation visualization component, and showed various calculated statistics along with their actual observed values. Parameters such as the time to notify each room were shown here, as well as any other data relevant to the progress of the evacuation.

Inspired by their success in defining the sensescape, we used sockets to connect each data source to the various visualizations. This proved beneficial, as it allowed each visualization component to operate loosely coupled to each sensor and without the need

to know if it was operating on historical or real-time data, or on the physical environment or its virtual equivalent.

5.3.5 Results and Discussions

Unlike the previous two studies, which focused on experimental results, the results of the building monitor prototype were instructional in nature. This was seen as appropriate, since forcing an experimental study would not have led to any interesting new data. For example, testing if symbiotic simulation improved egress is trivial; the library egress study already showed this under conditions of *greater* uncertainty, as the sweep teams in this example are assumed to be trained and diligent. The sweep team exercise was used as a focal point for this study, but the true research value here comes from its methodical approach to apply perennial simulation to a real-world system. In this sense, it serves as model for the creation of such systems.

Given this distinction, the results of the building monitor prototype were categorized into two major categories: those related to our use of **socket sensors** and those which are not. The decision to use socket sensors led to a cascade of interesting consequences, most of which may be incorporated into a future version of the reference implementation. As expected, none of these improvements affected the theoretical framework, which is considerably more inert than its implementation. The findings inspired by the use of sockets in this framework will be detailed first; following that, several additional points of interest will be covered.

First, we observe that the use of **socket sensors** for all sensor types allowed for an easy abstraction of data by separating the actual data from its source. Socket sensors were originally intended to wrap non-Java sensors in a portable way, but they have now shown themselves to be extremely useful for swapping around the variety of configurations one might wish to study. The reference implementation can be somewhat rigid in regards to the sensor setup, in that it assumes sensors will never change for a given target during a simulation run. This is clearly false; for example, if a virtual sweep team notifies a virtual room to evacuate, that room must no longer read from the real-time sensors, but must instead switch to an estimation model. We were able to overcome this limitation fairly easily by simply switching to a new sensor socket —the simulation was unaware of the change.

The second interesting side-effect of using entirely **socket sensors** was the elimination of the need for a sensor dependency tree. Every sensor maintained its own list of child sensors, and received information from them in a manner perceived as real-time (either actually in sync with reality, or manually advanced by the controller). Thus, each

sensor received all the data it needed in the correct order by virtue of the network itself, thereby removing the need for an explicit dependency tree. There are some downsides to completely removing the tree from the library. First, sensor loops will now risk repeating forever instead of generating an exception. Second, the GUI would now be unable to properly represent sensor dependencies. Nonetheless, it would be an interesting exercise to see how much of the dependency tree could be replaced with a sockets-based approach.

Next, we observe that sockets were a useful means of communicating with the visualizer as well as with the models themselves. This allows the visualizations to share code with the models that read from the sensor network, although the benefit is only apparent if using the same programming language for the simulation component and the GUI. Nonetheless, we might consider modifying the specification of sensors in the perennial framework to account for a default socket on each sensor.

Regarding non-socket results, we observe that the perennial framework and implementation performed well in the task of setting up a prototype study. The value of using a *virtual study* cannot be understated, as it allowed us to quickly prepare a system using the data and models on hand, while the flexibility of the sensescape (even without sockets) enabled updating components as new data became available. The system constructed exhibited the capability to handle real-world systems with only a small amount of overhead attributed to our perennial framework. Historical datasets were saved as they were read from the sensescape, and replaying past scenarios was always possible.

Finally, the building monitor prototype displayed all of the qualities of a modern perennial simulation. The use of *historical walkthroughs* reflected our work on MMOHILS, and it is reasonable to assume that new *niche configurations* could easily be attached to our system as they are developed. Results gathered by our system were used to generate simple, optimizing team notification strategies, leading to the possibility for symbiotic decision support. The system even incorporated a legacy model, in the form of the EvacNET network specification. Thus, we consider this study a late-stage transitional study, with all the modernity of a full perennial simulation system, coupled with a small number of legacy elements to help gather information.

Chapter 6

Conclusions

In this work, we devised a *perennial simulation framework* that captured the benefits of online symbiotic simulation for crisis management in a concise conceptual model. Although previous research has touched on the subject of crisis management simulation, a generalized method that operates across its entire breadth has remained relatively unexplored. Here, we have demonstrated the material benefits associated with moving towards such a system, as well as the risks of leaving crisis management tasks purely to intuition and limited, one-off modeling. The perennial simulation framework eases development of simulations which meet crisis management's demanding standards of allowing studies with a *hindsight*, *foresight*, or *decision support* focus. This system, in turn, can be used to study and react to the tasks before, during, and after the occurrence of a crisis.

Following the framework's formulation, we created an implementation using current technology, and then applied this implementation to investigate the research implications of perennial simulations. Care was taken to ensure that any studies with experimental components were verified and validated to an acceptable degree, as described in Sections 5.1.3.1 and 5.2.4. In addition to serving as experiments, these studies have explored several novel areas of research that the perennial simulation framework enables. First and foremost is the niche configuration termed MMOHILS, a type of simulation which provides the means to mine data from human users interacting within a virtual world. This opened up the possibility of validating new forms of software agents in previously un-observable contexts. A full treatment of validation in MMOHILS was provided; in particular, methods of *role* and *psycho-social* validation were required above and beyond the validation requirements of traditional simulation. In addition to MMOHILS, a form of sensor abstraction was used to allow real human agents to interact with virtual users.

Finally, several means of managing the complexities of performance tradeoffs in crisis management were provided.

The following sections clarify and expand upon the previous summary. Section 6.1 consolidates the methods used and findings observed, with Section 6.2 focussing specifically on contributions and achievements. Section 6.3 provides broader comments pertaining to the fields of simulation and crisis management themselves. Section 6.4 denotes specific areas of interest to future research, and Section 6.5 concludes the chapter.

6.1 Summary

We proceeded methodically from a full examination of the shortcomings affecting crisis management and computer simulation to a consolidated framework that enables construction of what we termed *perennial simulations* designed to overcome the weaknesses inherent in current techniques.

In Chapters 2 and 3, we demonstrated the elements missing from each domain of research. Crisis management encompasses at least five distinct intervention zones, categorized into the three major research categories of *hindsight*, *foresight*, and *decision support*. The level of sophistication apparent in existing research varied by organization and by sub-domain. Some relied on simulation to an appreciable degree, but the overall appeal to simulation was low. We consider this unfortunate, as simulation is a mature field with many powerful techniques —such as agent-based simulation, symbiotic simulation, and human-in-the-loop simulation— that crisis managers could use to achieve the *information superiority* that their own research identified as critical. Viewed from the converse perspective, computer simulation was lacking an acceptably valid means of modeling humans in situations of uncertainty. The state of the art in crisis management simulation was found lacking for all parties involved. To realize the potential of this trans-disciplinary field and address its current inadequacies, we introduced the concept of a *perennial simulation framework* built specifically for crisis management.

In Chapter 4, we moved from an abstract idea of the shape of a solution to the solution itself. This was accomplished by carefully examining the intended usage and lifecycle of the framework. Given this clear picture of how the system might be used, we then created a framework specification with the goals of being *modular*, *powerful*, and *flexible*, while requiring *minimal cognitive overhead* and adequately *capturing reality* as well as meeting several quantifiable goals. This is discussed further in Section 6.2. A high-level component interaction diagram served to present the main elements of the perennial framework and their corresponding communication requirements. These components

were then detailed individually, with first-order logic equations used to clarify potential ambiguities. This formalism helped us identify a promising *niche configuration* we termed MMOHILS, which was further developed through experimentation.

After thoroughly detailing the theoretical framework, development of a *reference implementation* began, with the goal of using current technology to expose our framework to a wider audience. Given proper deliberation, Java was selected as both the language of the library itself and that of its GUI component (coupled with SWT). The implementation relied on object-oriented programming techniques in general, and weak type guarantees in regards to user-defined datatypes [132]. As a tertiary goal, it was decided that the implementation should remain as free as possible from artifacts specific to the field of crisis management. This would ensure that other trans-disciplinary collaborations with the field of computer simulation would be possible, and boosted the value of the framework as a generic method for handling modern simulation studies.

Given a properly conceived framework and implementation, several studies were performed as detailed in Chapter 5. The first of these tested the MMOHILS *niche configuration* to create an egress simulation of a *virtually established* building [135]. Besides serving as a demonstration of the perennial framework, this study also attempted to discern the potential benefit of using symbiotic simulation to aid general building egress [154]. MMOHILS was an appropriate choice for this study, as its virtual sensors could assume perfect knowledge, while at the same time its output could be captured in totality for further analysis. This enabled a rapid development cycle, allowing us to focus on the task at hand —studying the human factors of symbiotic-guided egress— rather than be delayed calibrating a physical building's sensor network. After performing a thorough validation of our perennial simulation, we proceeded to model egress of a two-story library building with two sets of potential detours, using EvacNET to provide optimized symbiotic egress feedback to users. Despite the simplicity of the building's layout, we observed a 14% improvement in average egress time due to symbiotic feedback, with further positive implications for training exercise and non-egress applications.

The second study explored the use of an abundance of real-world traffic camera data to facilitate *foresight* and *hindsight* studies for incident response management on a university campus [163]. The experimental aspect of this study focused on determining the *response radius* within which security personnel would be able to reach an incident by vehicle. A pipelined image-processing algorithm was used to extract traffic data from security camera footage, thus leading to the creation of another *virtual study* of a physical system via perennial simulation. After constructing the system, verification and validation proceeded as per our stated requirements for perennial systems [153]. Experiments

with the system demonstrated that once congestion increased to twice the normal "mid-day" traffic level, the response radius at non-central areas on campus became severely constrained. In terms of the perennial framework, this study demonstrated a simple, effective means of embedding "legacy" simulations into a perennial simulation which allowed them to benefit from most of the perennial components, especially the sens-escape. In addition, the image processing algorithm performed well, with a few caveats that would have to be addressed for it to operate in real time.

The third study involved the creation of a perennial system prototype that was tied closely to a real-world building and its sensor network (**Indriya**). Various physical sensors were mapped to a *virtual study* that attempted to determine occupancy levels and guide "sweep teams" with real-time feedback [174]. In addition, an automatic symbiotic feedback component was triggered if cleared rooms suddenly became occupied, causing the sweep teams to be split up or re-routed so as to minimize total building notification time. Unlike the previous studies, which existed also for an experimental component, the building monitor study focused on presenting a complete, thorough prototype of the perennial system in a *virtual study* of an existing physical location and its technology. As such, its results were instructional rather than experimental. Most interesting was the concept, introduced in [135], that virtual users could interact with each other regardless of whether they were software agents or actual human participants.

Throughout these studies, care was taken to ensure that perennial simulation remained feasible for real-world applications. Our first experiment demonstrated the scalability of MMOHILS and symbiotic simulation, both of which can scale to 100 (for *experimental* MMOHILS) or 1000 (for *unannounced* MMOHILS) connected users in a world of arbitrary size. In addition, our second experiment confirmed the benefit offered by perennial simulation in terms of enhanced decision support, improving the Implementers' predictive accuracy by roughly 6% when faced with sensor-level uncertainty. These quantitative improvements are in addition to the numerous qualitative benefits of perennial simulation.

6.2 Contributions and Achievements

The research carried out for this thesis followed the broad outline of Section 6.1. We will now consolidate this research summary into the primary contributions and achievements that resulted from this thesis as a whole.

The first and most important of the primary contributions is that of the perennial simulation framework. Existing techniques from simulation and crisis management were

improved upon, and new techniques (in particular MMOHILS) fostered, in an attempt to bridge the trans-disciplinary gap and achieve full crisis management simulation. Perennial simulation shares a similar **creation** phase with traditional simulation, and exposes the system to multiple entities during the **perennial** phase [132]. We met our goal of scaling to dozens of agents and city-sized environments (see: Section 5.1.6). Additionally, the perennial simulation framework was shown to provide a benefit to **Implementers** even in the face of increasing uncertainty (see: Section 5.2.7). Symbiotic simulation was also shown to be effective for crisis management activities such as egress, in addition to proving practical within the tight time constraints of symbiotic simulation (see: Sections 5.1.5, 5.1.6). Thus, we conclude that the perennial simulation framework meets our goal of running of *foresight*, *hindsight*, and *decision support* studies, a well as reacting nimbly to unpredictable crisis events.

The primary contribution of MMOHILS arose as a direct result of the perennial framework's first-order logic formalisms. This *niche configuration* combines the strengths of human-in-the-loop simulation with massively multiplayer online games, and helped us to meet our goal of exploring the various physical/virtual and virtual/virtual world interactions implied by the framework. In addition, MMOHILS and its sensor design allowed us to meet our goals of providing validation techniques for agent behavior in crisis-relevant situations, as well as providing a general technique for mixing human and virtual agents in virtual world simulation studies (see: Sections 4.2.6, 4.3.3.2, 5.1.5). MMOHILS is one of many potential *niche configurations*, and it benefits from existing fully within the perennial framework rather than being treated as an exception.

Regarding our minor contributions, the largest contribution in terms of source code was an implementation of the perennial simulation framework (see: Section 4.2). Although the true value of our research exists at the framework level, a workable implementation was required to enable further experimenting with the concept of a perennial simulation and MMOHILS. The implementation was shown to be flexible enough to deal with our three, vastly different studies, as well as providing support for "legacy" simulations. Several minor decisions were made in the implementation phase, including the use of a dependency tree with a shared root node to manage sensor dependencies.

Other minor contributions such as an analysis of physical to virtual world interactions and a study of human response to symbiotic feedback during a crisis have already been covered as part of the major contribution of the framework. One final contribution was the need to present a combined approach to the problem of measuring the extent of a crisis. A full Pareto front analysis would be used for non-time-critical *foresight* and *hindsight* studies, while a utility function would approximate the Pareto front favoring local maxima on a per-parameter basis (see: Section 4.3.3.1). This exhibited an efficiency

that was advantageous for *decision support* studies, where time was at a premium. The demonstrated efficacy of symbiotic simulation is pertinent in that perennial simulation enables the use of such a technique for decision support at a time of crisis.

6.3 General Discussion

The work performed on creating the *perennial simulation framework*, its implementation, and the three experimental studies led to several observed results which were broader in scope than those listed in the previous section. These generally applied to the framework or its implementation *in toto*, or generally to the fields of simulation or crisis management.

- We created a framework which enables simulation of *hindsight*, *foresight*, and *decision support* studies in a way that is flexible and allows component reuse. In particular, the strong hierarchy present in the sensescape, the distinction between physical and virtual worlds, and the ability of human participants to inter-mix with software agents stand out as strong features of our framework.
- We demonstrated the value of creating *virtual studies* of physical systems, in that they allow experimenting on hypothetical locations and situations. We demonstrated a technique (MMOHILS) that thrives in virtual spaces and allows gathering valid human data in circumstances which are otherwise unmeasurable. We demonstrated verification and validation of such systems, which are acceptable except under conditions of extreme stress [139].
- We created a reference implementation that proved its usefulness over the course of three separate studies. The implementation featured encapsulating "legacy simulations" in a way that allowed them to make use of the perennial framework to some degree. This system also quantified the magnitude of a given crisis using either a utility function or a Pareto front depending on the time constraints of the situation.
- We studied the efficacy of symbiotic simulation for crisis management, demonstrating that users tended to respond positively to advice. Such systems might also be used for community training exercises.

The experiments performed covered limited sample points, but there were several key factors which allow us to generalize their results. First, the scalability tests from Section 5.1.6 indicate that MMOHILS can scale up to several hundred or several thousand

users depending on the exact technique used. World size scales almost indefinitely, and symbiotic simulation can generally keep up with the simulation as it increases, assuming some necessary trade-offs are made. Second, the three experiments exhibited a broad spread of abstraction, varying from the highly abstract world of the building egress study to the realistic environment of the notification experiment. This demonstrates the perennial simulation framework's versatility when applied to real-world problems as well as highly speculative niche areas of research. Third, the experiments all feature obvious candidates for parallelism or workload distribution. For example, the pipelined image processing algorithm used in the incident response study can be parallelized at each stage of the pipeline as well as for each security camera feed. Similarly, the "What-If?" analyses used in the library egress study are distinct from each other and can be increased in number with the addition of more hardware. This, combined with information management techniques already present in the field of distributed computing, allow us to conclude that perennial simulations in general can scale to vast sizes with nominal effort. Finally, the distinction between the theoretical framework and its implementation provide flexibility in the event that the latter is found lacking in some regard. The third experiment encountered this to some degree, as socket sensors were found to be superior to integrated (Java) sensors in all regards. Thus, the implementation can be refactored many times without affecting the framework proper, providing a necessary degree of adroitness in tuning the implementation to overcome issues as they arise.

6.4 Limitations and Recommendations for Future Research

The work presented forms a cohesive whole, with each of the three studies branching off to explore various ambiguities or areas of interest. That said, several important limitations to the current approach will now be discussed, followed by possible extensions which merit future research.

The primary limitations to the perennial simulation framework and its implementation are threefold. First, the role of Implementers is more limited than it should be in reality. The ability to guide Implementers acting within the simulation loop via information updates is a great boost to their role compared to traditional simulation, but this nonetheless represents a narrowing of their role from a crisis management point of view. Essentially, the simulation currently *delegates* decision-making to the Implementer, while an ideal setup would have the Implementer delegate tasks to the simulation system. This is a necessary restriction, which can gradually be lifted as the field of crisis management simulation matures. Second, none of the test studies covered the entirety of the simulation framework. This was also a necessary caveat, as building a complete, single study

would have been costly and risky —indeed, improvements from the first study were crucial to the success of the second and third studies. In addition to this, we attempted to introduce overlap between the three studies, and explained in the previous section which key factors led us to conclude that all three together provided a reasonable coverage of the framework. Third, all software components designed for this thesis were written in a single programming language: Java. This risks restricting the applicability of our technique, as different languages favor different programming techniques. For example, we saw earlier that Javascript severely limits the way that object-oriented techniques behave. Fortunately, we consider this limitation to be relatively harmless: Java is a minimalistic language, and several languages (C#, Python, and to some extent C++) contain all of Java's functionality as a subset. Nonetheless, language interoperability remains a limitation of our current framework implementation.

Regarding extensions, the first involves the framework itself, particularly the implications of a world being both *virtually established* and a *virtual study* simultaneously. Such locations would theoretically have the potential to both sense and influence any intersection between the sensescape and effectscape, hypothetically enabling a virtual user to have some simulacrum in the physical world. For example, a mobile phone might be used as a "window" through which one views augmented reality avatars. This idea is fascinating, but far beyond the bounds of what we were attempting to study. As it stands, even the notion of *virtual establishment* has not been fully explored, as it has no directly obvious benefit to crisis management simulation.

The second extension involves the implementation of the framework, specifically the need to converge on a solution quickly. We proposed the use of a utility function, which operates much more efficiently than a full Pareto front calculation, while still approximating its per-parameter maximum to an acceptable degree. Several techniques exist which are even faster than utility functions, such as the "elimination by aspects" referenced by [24]. Such techniques were not relied upon in the current implementation because it is not entirely clear that they are valid substitutes for utility functions in all cases. Assuming these faster methods offer some guarantee of validity, one might use them to guide the convergence of a Pareto front estimation, thereby creating a middle-ground solution. For crisis management, only the two extremes of fast and comprehensive are required. However, other sub-domains of simulation might benefit from a middle-ground solution which is slightly slower than a utility function calculation while also providing a more complete picture of the situation.

A related area of additional research involves the creation of various framework implementations with different goals than our own. The most obvious of these would be an implementation designed to "plug in" to the High Level Architecture and thus maintain

the benefits of our framework coupled with the HLA's robust and extensive research history. More recent improvements in this domain such as **HLA Evolved** feature promising modernizations, such as extensive XML support and IPv6 addressing [182]. The primary benefit of our Java reference implementation comes from the accessibility of the language, but we would not want to limit the use of our framework by needlessly restricting the number of implementations. In addition, having multiple implementations ensures that the entire feature-space of the framework can be explored, including those elements that might be easier to model in languages other than Java.

Finally, the traffic study itself requires improvements to the automated image processing algorithm. In particular, a better pre-processing step coupled with rigorous parallelization of the image processing pipeline would enable operation on real-time data without the need for manual correction. Unfortunately, this by itself has no value unless the legacy model is upgraded to a first-class perennial component. At that point, the possibilities of the system widen greatly; for example, one might incorporate support for pedestrian dynamics, thus allowing the system to demonstrate its full potential for symbiotic-guided decision support across all modes of travel throughout the campus road network. Although interesting from a traffic simulation point of view, this would not add much to our understanding of the perennial simulation framework.

6.5 Concluding Remarks

Over the course of this thesis, we have shown how one might approach the transdisciplinary field of crisis management simulation. Like the bulk of existing research, our work leans more towards the *hindsight* and *foresight* elements of crisis management. This is pragmatic and sensible, as the FEMA diagrams demonstrate that a substantial effect can be gained from preparedness and information superiority. Unlike related work, however, our framework has demonstrated a clear capacity for use in *decision support* at the time of crisis. In particular, we recommended symbiotic simulation for use in this area, and demonstrated its efficacy. Problems with using symbiotic simulation (such as its need for a performance metric) were satisfactorily resolved. Three studies tested our framework at various levels: two with experimental results and one which stressed the framework through the implementation of a prototype. Various exciting elements of our framework, such as MMOHILS, were also explored. The totality of our work brings the fields of crisis management and computer simulation significantly closer together, and enables various new and useful avenues of future research.

Bibliography

[1] Erik Larson. *Isaac's Storm: A Man, a Time, and the Deadliest Hurricane in History*. Knopf Doubleday Publishing Group, 1st edition, 2000.

[2] H.S. Frankenfield. August 1915 monthly weather review. Weather Bureau, 1925. URL http://www.aoml.noaa.gov/general/lib/lib1/nhclib/mwreviews/1915.pdf.

[3] Jeffery K. Taubenberger and David M. Morens. 1918 influenza: the mother of all pandemics. Centers for Disease Control and Prevention, 2006. URL http://www.cdc.gov/ncidod/EID/vol12no01/05-0979.htm.

[4] World Health Organization. Pandemic influenza preparedness and response: a WHO guidance document. WHO Global Influenza Program, 2009.

[5] Seth Solomonow and Scott Gastel. NYC DOT announces historic safety trend continues with sustained lows in citywide traffic fatalities. New York City Department of Transportation, 2011. URL http://home2.nyc.gov/html/dot/html/pr2011/pr11_16.shtml.

[6] International Olympic Committee. Olympic marketing fact file, 2008 edition. International Olympic Committee, 2008. URL http://www.olympic.org/Documents/marketing_fact_file_en.pdf.

[7] U.S. Bureau of the Census. U.S. bureau of the census, decennial censuses of 1980 and 1990. U.S. Bureau of the Census, 1980, 1990. URL http://www.census.gov/prod/www/abs/decennial.

[8] Edwin Rowe. The Los Angeles automated traffic surveillance and control (ATSAC) system. *IEEE Transactions on Vehicular Technology*, 40(1):16–20, 1991.

[9] S. Tufekci and W.A. Wallace. The emerging area of emergency management and engineering. *Engineering Management, IEEE Transactions on*, 45(2):103–105, may 1998. ISSN 0018-9391.

[10] Giorgio Fagiolo, Alessio Moneta, and Paul Windrum. A critical guide to empirical validation of agent-based models in economics: Methodologies, procedures, and open problems. *Computational Economics*, 30(3):195–226, 2007. ISSN 0927-7099.

[11] Merriam webster dictionary. Merriam-Webster, Incorporated, 2012.

[12] R. F. Shangraw, Jr. Beyond the boundaries: Simulation and transdisciplinarity. Keynote, Winter Simulation Conference 2011, 2011.

[13] Ed Lee. BBC – future media standards & guidelines — use of colour and colour-contrast standards v1.7. http://www.bbc.co.uk/guidelines/futuremedia/accessibility/colour_contrast.shtml, 2009.

[14] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, and Gregg Vanderheiden. Web content accessibility guidelines (WCAG) 2.0. http://www.w3.org/TR/WCAG20, 2008.

[15] Averill M. Law. *Simulation Modeling and Analysis*. McGraw-Hill Education, New York, 4th edition, 2007. ISBN 978-0-07-110336,7.

[16] Roger D. Smith. Simulation article, encyclopedia of computer science. Nature Publishing Group.

[17] Juta Pichitlamken, Supasit Kajkamhaeng, and Putchong Uthayopas. High performance spreadsheet simulation on a desktop grid. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, pages 663–670. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[18] S.N. Othman, N.H. Mustaffa, and R. Sallehuddin. Supply chain spreadsheet simulation optimization. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on*, pages 188–193, 2012.

[19] T.J. Schriber. Simulation for the masses: Spreadsheet-based monte carlo simulation. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 1–11, 2009.

[20] R. D. Fujimoto, E. P. Lunceford, and A. M. U. (editors). Grand challenges for modeling and simulation: Dagstuhl report. Technical Report 350, Schloss Dagstuhl. Seminar No. 02351, 2002.

[21] Frederica Darema. Grid computing and beyond: The context of dynamic data driven applications systems. In *Proceedings of the IEEE, vol. 93, #3*, pages 692–697, 2005.

[22] Dennis McGrath, Amy Hunt, and Marion Bates. A simple distributed simulation architecture for emergency response exercises. In *DS-RT '05: Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 221–228, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2462-1.

[23] Grady Venville. UWA & University of Kentucky collaboration. The University of Western Australia in Second Life (Blog), 2011. URL http://uwainsl.blogspot.sg/2011/11/uwa-university-of-kentucky.html.

[24] C.A.P. Smith and S.C. Hayne. A distributed system for crisis management. In *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, volume iii, pages 72–81, january 1991.

[25] Tung Bui and R. Subba. A tale of two disasters: Assessing crisis management readiness. In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1–10, january 2009.

[26] R. Desimone. The application of AI planning technology to crisis management tasks. In *Intelligent Planning and Scheduling Solutions (Digest No. 1996/197), IEE Colloquium on*, pages 6/1–6/3, october 1996.

[27] U. Gupta and N. Ranganathan. Multievent crisis management using noncooperative multistep games. *Computers, IEEE Transactions on*, 56(5):577–589, may 2007. ISSN 0018-9340.

[28] J. Lee and T. Bui. A template-based methodology for disaster management information systems. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, volume 2, page 7, january 2000.

[29] I.J. Martinez-Moyano, S.H. Conrad, E.H. Rich, and D.F. Andersen. Modeling the emergence of insider threat vulnerabilities. In *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, pages 562–568, 2006.

[30] Jr. Nunamaker, J.F., E.S. Weber, C.A.P. Smith, and M. Chen. Crisis planning systems: tools for intelligent action. In *System Sciences, 1988. Vol.III. Decision Support and Knowledge Based Systems Track, Proceedings of the Twenty-First Annual Hawaii International Conference on*, volume 3, pages 25–34, january 1988.

[31] Tamara Kaplan. The tylenol crisis: How effective public relations saved Johnson & Johnson. The Pennsylvania State University, 1998.

[32] Philip Henslowe. *Public Relations: A Practical Guide to the Basics*. Kogan Page, 2nd edition, 2003. ISBN 978-0749440725.

[33] Gene Klann. *Crisis leadership: using military lessons, organizational experiences, and the power of influence to lessen the impact of chaos on the people you lead.* Center for Creative Leadership, 2003.

[34] J.L. Jacobs, M.C.P. Dorneich, and P.M. Jones. Activity representation and management for crisis action planning. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 1, pages 961–966, october 1998.

[35] *Cambridge Advanced Learner's Dictionary.* 3rd edition, 2008.

[36] Tung Bui, Sungwon Cho, Siva Sankaran, and Michael Sovereign. A framework for designing a global information network for multinational humanitarian assistance/disaster relief. *Information Systems Frontiers*, 1:427–442, 2000. ISSN 1387-3326.

[37] USA Federal Emergency Management Agency. FEMA. http://www.fema.gov, 2000.

[38] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki. Managing escalation of collaboration processes in crisis mitigation situations. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 45–56, 2000.

[39] Boumediene Belkhouche, Reda Bakeer, and Salah Al-saleh. An expert decision support system for crisis management, 2002.

[40] Craig C. Douglas, Robert A. Lodder, Richard E. Ewing, Yalchin Efendiev, Guan Qin, Janice Coen, Mauricio Kritz, Jonathan D. Beezley, Jan Mandel, Mohamed Iskandarani, Anthony Vodacek, and Gundolf Haase. DDDAS approaches to wildland fire modeling and contaminant tracking. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 2117–2124. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

[41] Frank Fiedrich. An HLA-based multiagent system for optimized resource allocation after strong earthquakes. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 486–492. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

[42] Sanjay Jain and Charles R. McLean. A concept prototype for integrated gaming and simulation for incident management. In *WSC '06: Proceedings of the 38th conference on Winter simulation*, pages 493–500. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

[43] Ameya Shendarkar, Karthik Vasudevan, Seungho Lee, and Young-Jun Son. Crowd simulation for emergency response using BDI agent based on virtual reality. In

WSC '06: Proceedings of the 38th conference on Winter simulation, pages 545–553. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

[44] Malcolm Yoke Hean Low, Wentong Cai, and Suiping Zhou. A federated agent-based crowd simulation architecture. In *Proceedings of the 2007 European Conference on Modelling and Simulation*, pages 188–194, 2007.

[45] Department of Defense Defense Modeling and Simulation Office. HLA run-time infrastructure RTI 1.3-next generation programmer's guide version 5, 2002.

[46] Steven P. Wilcox. Agentizing the social science of crime. In *WSC '11: Proceedings of the 43rd Conference on Winter Simulation*. Winter Simulation Conference, 2011.

[47] Tim Lant, Megan Jehn, Cody Christensen, Ozgur M. Araz, and John W. Fowler. Simulating pandemic influenza preparedness plans for a public university: a hierarchical system dynamics approach. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1305–1313. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[48] Ali Ekici, Pinar Keskinocak, and Julie L. Swann. Pandemic influenza response. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1592–1600. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[49] Margaret L. Brandeau. Infectious disease control policy: a role for simulation. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1578–1582. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[50] T. Sakairi, T. Tamada, and H. Nakata. GIS crisis-management systems using Ajax technology. In *SICE Annual Conference, 2008*, pages 3043–3046, august 2008.

[51] S.E. Cross, E. Walker, H. Forsdick, and T.C. Kral. Collaboration technology for crisis management. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1994. Proceedings., Third Workshop on*, pages 48–52, april 1994.

[52] L.J. Dunn. HCI factors affecting quality of information in crisis management systems. In *Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on*, pages 324–325, november 1996.

[53] J. Rossmann, M. Schluse, M. Hoppen, and R. Waspe. Integrating semantic world modeling, 3D-simulation, virtual reality and remote sensing techniques for a new class of interactive GIS-based simulation systems. In *Geoinformatics, 2009 17th International Conference on*, pages 1–6, 2009. ISBN 978-1-4244-4562-2.

[54] D. J. Medeiros, Eric Swenson, and Christopher DeFlitch. Improving patient flow in a hospital emergency department. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1526–1531. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[55] Le Yin Meng and Trevor Spedding. Modelling patient arrivals when simulating an accident and emergency unit. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1509–1515. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[56] Ronald E. Giachetti. A simulation study of interventions to reduce appointment lead-time and patient no-show rate. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 1463–1468. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[57] James Fackler and Michael Spaeder. Why doesn't healtcare embrace simulation and modeling; what would it take? In *WSC '11: Proceedings of the 43rd Conference on Winter Simulation*. Winter Simulation Conference, 2011.

[58] James C Fackler, Charles Watts, Anna Grome, Thomas Miller, Beth Crandall, and Peter Pronovost. Critical care physician cognitive task analysis: an exploratory study. volume 13, page R33. BioMed Central, 2009.

[59] Averill M. Law. How to build valid and credible simulation models. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 39–47. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[60] Malcom Yoke Hean Low, Kong Wei Lye, Peter Lendermann, Stephen John Turner, Reman Tat Wee Chim, and Surya Hadisaputra Leo. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Utrecht, The Netherland*, pages 85–92, 2006.

[61] Heiko Aydt, Stephen John Turner, Wentong Cai, Malcolm Yoke Hean Low, Peter Lendermann, Boon Ping Gan, and Rassul Ayani. Preventive what-if analysis in symbiotic simulation. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 750–758. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[62] Catriona Kennedy and Georgios Theodoropoulos. Adaptive intelligent modelling for the social sciences: Towards a software architecture. Technical Report CSR-06-11, University of Birmingham, 2006.

[63] N. Koyuncu, Seungho Lee, K.K. Vasudevan, Young-Jun Son, and P. Sarfare. DDDAS-based multi-fidelity simulation for online preventive maintenance scheduling in semiconductor supply chain. In *Simulation Conference, 2007 Winter*, pages 1915–1923, 2007.

[64] Dylan Evans, Annerieke Heuvelink, and Daniel Nettle. The evolution of optimism: A multi-agent based model of adaptive bias in human judgement. Scientific Methods for the Analysis of Agent-Environment Interaction Symposium, 2003.

[65] Seungho Lee and Young-Jun Son. Integrated human decision making model under belief-desire-intention framework for crowd simulation. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 886–894. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[66] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *SIGGRAPH'93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 135–142, 1993.

[67] Alok Chaturvedi, Angela Mellema, Sergei Filatyev, and Jay Gore. DDDAS for fire and agent evacuation modeling of the Rhode Island nightclub fire. In *International Conference on Computational Science (3)*, Lecture Notes in Computer Science, pages 433–439. Springer, 2006. ISBN 3-540-34383-0.

[68] Lin Padgham, Dave Scerri, Gaya Jayatilleke, and Sarah Hickmott. Integrating BDI reasoning into agent based modelling and simulation. In *WSC '11: Proceedings of the 43rd Conference on Winter Simulation*. Winter Simulation Conference, 2011.

[69] Maria Hybinette and Richard Fujimoto. Dynamic virtual logical processes. *SIGSIM Simul. Dig.*, 28(1):100–107, 1998. ISSN 0163-6103.

[70] Maria Hybinette and Richard M. Fujimoto. Scalability of parallel simulation cloning. In *SS '02: Proceedings of the 35th Annual Simulation Symposium*, page 275, Washington, DC, USA, 2002. IEEE Computer Society.

[71] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. Agent-based simulation platforms: review and development recommendations. *Simulation: Transactions of The Society for Modeling and Simulation International*, 89(9):609–623, 2006.

[72] Levent Yilmaz, Guangyu Zou, and Osman Balci. A robust evolutionary strategy for generative validation of agent-based models using adaptive simulation ensembles. In *WSC '11: Proceedings of the 43rd Conference on Winter Simulation*. Winter Simulation Conference, 2011.

[73] A. Louloudi and F. Klugl. Immersive face validation: A new validation technique for agent-based simulation. In *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pages 1255–1260, 2012.

[74] Cristina Montaola-Sales, Stephan Onggo, and Josep Casanovas-Garcia. *Agent-based simulation validation: A case study in demographic simulation.* IARIA, 2011.

[75] Tibor Bosse, Charlotte Gerritsen, Michel CA Klein, and Frank M Weerman. Development and validation of an agent-based simulation model of juvenile delinquency. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 200–207. IEEE, 2009.

[76] A.G. Barnes and S. Skans. Fifty years of success and failure in flight simulation. In *50 Years of Flight Simulation.* Royal Aeronautical Society, Flight Simulation Group, 1979.

[77] Morris Ackerman. An integrated space-flight simulator. IRE Transactions on Military Electronics, 1959.

[78] Cary J. Peaden. Using simulation for launch team training and evaluation. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 1281–1290. Winter Simulation Conference, 2005. ISBN 0-7803-9519-0.

[79] Patrick M. Downes, Michael J. Kwinn, Jr., and Donald E. Brown. Using agent-based modeling and human-in-the-loop simulation to analyze army acquisition programs. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 994–1000. Winter Simulation Conference, 2004. ISBN 0-7803-8786-4.

[80] W. Hopkinson and J. Sepulveda. Real time validation of man-in-the-loop simulations. Proceedings of the 1995 Winter Simulation Conference, 1995.

[81] John J. Fruin. *Pedestrian Planning and Design.* Metropolitan Association of Urban Designers and Environmental Planners, 1971.

[82] R. L. Francis, T. M. Kisko, and C.R. Nobel. EvacNET4 user's guide. University of Florida, 1998.

[83] K. Togawa. Study on fire escapes basing on the observation of multitude currents building research institute. Ministry of Construction, Tokyo, 1955.

[84] G. Keith Still. *Crowd Dynamics.* PhD thesis, Warwick University, 2000.

[85] T. Aoki. The simulation system of passenger flow. *Quarterly Review of Rail Technical Research Institute Vol 35 No 2*, 1994.

[86] Media Department for Culture and Sport. *The Green Guide: Guide to Safety at Sports Grounds*. HMSO London, 4th edition, 1997. ISBN 0113000952.

[87] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons Limited, 1st edition, 2007.

[88] Michael J. North, Nicholson T. Collier, and Jerry R. Vos. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006. ISSN 1049-3301.

[89] Dirk Helbing and Peter Molnar. *Social Force Model for Pedestrian Dynamics*. American Physical Society, 1998.

[90] D. J. Kaup, Thomas L. Clarke, Rex Oleson, Linda Malone, and Florian G. Jentsch. Introducing age-based parameters into simulations of crowd dymanics. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 895–902. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[91] Charles S. Han, Xiaoshan Pan, and Kincho H. Law. A multi-agent based simulation framework for the study of human and social behavior in egress analysis. Center for Integrated Facility Engineering, Standford University, 2004.

[92] Sharon Adams Boxill and Lei Yu. An evaluation of traffic simulation models for supporting ITS development. *Security*, 7(2), 2000.

[93] Rutger Claes and Tom Holvoet. Multi-model traffic microsimulations. pages 1113–1123. WSC, 2009.

[94] Debashish Chowdhury, Ludger Santen, and Andreas Schadschneider. Statistical physics of vehicular traffic and some related systems. *Physics Reports-review Section of Physics Letters*, 329:199–329.

[95] Q. Yang. A Microscopic Traffic Simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3): 113–129, 1996. ISSN 0968090X.

[96] Transportation@MIT. A global collaboration to chart the future of urban mobility. MIT News, 2009. URL http://web.mit.edu/newsoffice/2009/transportation-collaboration.html.

[97] Cheng Tao and Shengguo Huang. An extensible multi-agent based traffic simulation system. In *Proceedings of the 2009 International Conference on Measuring Technology and Mechatronics Automation - Volume 03*, pages 713–716, Washington, DC, USA, 2009. IEEE Computer Society.

[98] S. Krauß, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597–5602, 1997.

[99] S. Krauß. *Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics*. PhD thesis, University of Cologne, Koln, Germany, 1998.

[100] Christian Gawron, Stefan Krauß, and Peter Wagner. Dynamic user equilibria in traffic simulation models, 1997.

[101] Simulation of urban mobility. http://sumo.sourceforge.net, 2011.

[102] Greg Tito. Peter Jackson confirms he's shooting The Hobbit at 48 fps. The Escapist Magazine, 2011. URL http://www.escapistmagazine.com/news/view/109210-Peter-Jackson-Confirms-Hes-Shooting-The-Hobbit-at-48-FPS.

[103] Robert Reinhardt and Snow Dowd. *Macromedia Flash8 Bible*. Wiley, 1st edition, 2006. ISBN 978-0471746768.

[104] Arthur Valadares, Thomas Debeauvais, and Cristina V Lopes. Evolution of scalability with synchronized state in virtual environments. In *Haptic Audio Visual Environments and Games (HAVE), 2012 IEEE International Workshop on*, pages 142–147. IEEE, 2012.

[105] C. Bouras, E. Giannaka, and T. Tsiatsos. Exploiting virtual objects' attributes and avatar's behavior in DVEs partitioning. In *Artificial Reality and Telexistence, 17th International Conference on*, pages 157–163, 2007.

[106] M. Sonnenfroh, T. Baeuerle, P. Schulthess, and M. Schottner. Sharing in-memory game states. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*, pages 58–65, 2010.

[107] Dewan Tanvir Ahmed and Shervin Shirmohammadi. Improving online gaming experience using location awareness and interaction details. *Multimedia Tools and Applications*, 61(1):163–180, 2012. ISSN 1380-7501.

[108] V.Y. Kharitonov. A consistency model for distributed virtual reality systems. In *Dependability of Computer Systems, 2009. DepCos-RELCOMEX '09. Fourth International Conference on*, pages 271–278, 2009.

[109] Michael Kwok and Johnny W Wong. Scalability analysis of the hierarchical architecture for distributed virtual environments. *Parallel and Distributed Systems, IEEE Transactions on*, 19(3):408–417, 2008.

[110] S. Kulkarni. Badumna network suite: A decentralized network engine for massively multiplayer online applications. In *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, pages 178–183, 2009.

[111] Cristina V Lopes, Thomas Debeauvais, and Arthur Valadares. Restful massively multi-user virtual environments: A feasibility study. In *Games Innovation Conference (IGIC), 2012 IEEE International*, pages 1–4. IEEE, 2012.

[112] Rogelio Adobbati, Andrew N. Marshall, Andrew Scholer, and Sheila Tejada. Gamebots: A 3D virtual world test-bed for multi-agent research. In *In Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[113] Jakub Gemrot, Rudolf Kadlec, Michal Bída, Ondřej Burkert, Radek Píbil, Jan Havlíček, Lukáš Zemčák, Juraj Šimlovič, Radim Vansa, Michal Štolba, Tomáš Plch, and Cyril Brom. Pogamut 3 can assist developers in building AI (not only) for their videogame agents. In Frank Dignum, Jeff Bradshaw, Barry Silverman, and Willem Doesburg, editors, *Agents for Games and Simulations*, pages 1–15. Springer-Verlag, 2009. ISBN 978-3-642-11197-6.

[114] Joost Oijen, Los Vanhe, and Frank Dignum. CIGA: A middleware for intelligent agents in virtual environments. In Martin Beer, Cyril Brom, Frank Dignum, and Von-Wun Soo, editors, *Agents for Educational Games and Simulations*, volume 7471 of *Lecture Notes in Computer Science*, pages 22–37. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32325-6.

[115] Joost van Oijen, Han La Poutré, and Frank Dignum. Agent perception within CIGA: Performance optimizations and analysis. In *Proceedings of the International Workshop on Agent-Oriented Software Engineering, AAMAS'12*, 2012.

[116] Craig C. Douglas, Jonnathan D. Beezley, Janice Coen, Deng Li, Wei Li, Alan K. Mandel, Jan Mandel, Guan Qin, and Anthony Vodacek. Demonstrating the validity of a wildfire DDDAS. Lecture Notes in Computer Science, 2006.

[117] Robert V. Kenyon. The CAVE automatic virtual environment: Characteristics and applications. In Ahmed Noor, editor, *Human-Computer Interaction and Virtual Environments*, pages 149–168. NASA Conference Publication, 1995.

[118] University of Illinois at Chicago Electronic Visualization Laboratory. CAVE user's guide: CAVE library version 2.6. http://www.evl.uic.edu/pape/CAVE/prog/CAVEGuide.html, 1997.

[119] University of Illinois at Chicago Electronic Visualization Laboratory. Information on using Linux for CAVE development. http://www.evl.uic.edu/pape/CAVE/linux, 2001.

[120] Eric Schumacher and Jim Redner. New virtual reality gaming headset from Oculus VR gets kickstarted. http://www.oculusvr.com/press/

new-virtual-reality-gaming-headset-from-oculus-gets-kickstarted, 2012.

[121] Mark Ryan, Doug Hill, and Dennis Mcgrath. Simulation interoperability with a commercial game engine. In *In European Sim. Interop. Workshop*, pages 27–30, 2005.

[122] Meena Jha and L. O'Brien. A comparison of software reuse in software development communities. In *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, pages 313–318, 2011.

[123] Ralph E Johnson. Frameworks=(components+ patterns). *Communications of the ACM*, 40(10):39–42, 1997.

[124] S. Vinoski. CORBA: integrating diverse applications within distributed heterogeneous environments. *Communications Magazine, IEEE*, 35(2):46–55, 1997. ISSN 0163-6804.

[125] David N Gray, John Hotchkiss, Seth LaForge, Andrew Shalit, and Toby Weinberg. Modern languages and Microsoft's component object model. *Communications of the ACM*, 41(5):55–65, 1998.

[126] Foundation for Intelligent Physical Agents. FIPA agent message transport service specification. http://www.fipa.org/specs/fipa00067/index.html, 2002.

[127] Foundation for Intelligent Physical Agents. FIPA agent management specification. http://www.fipa.org/specs/fipa00023/index.html, 2002.

[128] Simulation Interoperability Standards Organization (SISO). Base object model (BOM) template specification. http://www.boms.info/spec/BOM_Specification_v2006_FINAL.pdf, 2006.

[129] Bernard P. Zeigler. *Theory of Modeling and Simulation*. John Wiley, 1976.

[130] Claudia Szabo. *Composable Simulation Models and Their Formal Validation*. PhD thesis, National University of Singapore, 2010.

[131] Robert G. Sargent. Verification and validation of simulation models. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 124–137, Piscataway, NJ, USA, 2007. IEEE Press. ISBN 1-4244-1306-0.

[132] Seth N. Hetu and Gary Tan. The big picture of symbiotic decision support: Designing a "What-If" simulation framework for crisis management. In *FISAT: Second International Conference on Advanced Computing and Communications Technologies for High Performance Applications*, 2010. Keynote address.

[133] John Sokolowski and Catherine Banks. *Principles of Modeling and Simulation: A Multidisciplinary Approach*. Wiley, 1st edition, 2009.

[134] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[135] Seth N. Hetu and Gary Tan. MMOHILS: a simpler approach to valid agents in human simulation studies. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 909–913. Winter Simulation Conference, 2008. ISBN 978-1-4244-2708-6.

[136] Nancy Pennington and Reid Hastie. The story model for juror decision making. In Reid Hastie, editor, *Inside the juror: The psychology of juror decision making*. Cambridge University Press, 1993.

[137] Leonard Adelman, Martin A. Tolcott, and Terry A. Bresnick. Examining the effect of information order on expert judgment. *Organizational Behavior and Human Decision Processes*, 56(3):348–369, december 1993.

[138] J. Shanteau and G.A. Dino. Environmental stressor effects on creativity and decision making. *Time pressure and stress in human judgment and decision making*, pages 293–308, 1993.

[139] Karl E. Weick. The collapse of sensemaking in organizations: The Mann Gulch disaster. Administrative Science Quarterly, 1993.

[140] Matt Weisfeld. *The Object-Oriented Thought Process*. Addison-Wesley Professional, 3rd edition, 2008. ISBN 978-0672330162.

[141] GTK+ Team. The gtk+ project, 2012. URL http://www.gtk.org.

[142] SWT Developers. SWT: The standard widget toolkit, 2012. URL http://www.eclipse.org/swt.

[143] Nokia. Qt, 2012. URL http://qt.nokia.com/products.

[144] Phoenix Developers. phoenix::about, 2012. URL http://byuu.org/phoenix.

[145] The computer language benchmarks game. http://shootout.alioth.debian.org, 2012.

[146] Ohloh: Open source statistics. http://www.ohloh.net, 2012.

[147] Tom Pender. *UML Bible*. Wiley, 1st edition, 2003. ISBN 978-0764526046.

[148] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, 2nd edition, 2004.

[149] Dai Bibo. Fuzzy synthetic evaluation of public crisis management capability. In *Computing, Control and Industrial Engineering (CCIE), 2010 International Conference on*, volume 2, pages 348–350, june 2010.

[150] Stefan Bleuler, Marco Laumanns, Lothar Thiele, and Eckart Zitzler. PISA — a platform and programming language independent interface for search algorithms. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science, pages 494–508, Berlin, 2003. Springer.

[151] Hisashi Tamaki, Hajime Kita, and Shigenobu Kobayashi. Multi-objective optimization by genetic algorithms: A review. In *International Conference on Evolutionary Computation '96*, pages 517–522, 1996.

[152] Brian Tierney, William E. Johnston, Ling Tony Chen, and Doron Rotem. Using high speed networks to enable distributed parallel image server systems, 1994.

[153] Seth N. Hetu and Gary Tan. Proper handling of real players in serious gaming studies. In *Learn to Game, Game to Learn; the 40th Conference ISAGA 2009*. Society of Simulation and Gaming of Singapore, 2009. ISBN 978-981-08-3769-3.

[154] Seth N. Hetu and Gary Tan. Potential benefits of symbiotic simulation to pedestrian evacuation. In *Asia Simulation Conference 2009*. Japan Society for Simulation Technology, 2009.

[155] W. Daamen and S.P. Hoogendoorn. Qualitative results from pedestrian laboratory experiments. *Pedestrian and evacuation dynamics 2003 (pp. 121-132)*, 2003.

[156] Leon Festinger. A theory of social comparison processes. *Human Relations*, 7: 117–140, 1954.

[157] Stanley Milgram, Leonard Bickman, and Lawrence Berkowitz. Note on the drawing power of crowds of different size. *Journal of Personality and Social Psychology*, 13 (2):79–82, 1969.

[158] Seth N. Hetu and Gary Tan. Real-time simulation in Java: A feasibility study. In *System Simulation and Scientific Computing, 2008. ICSC 2008*, pages 396–399. Asia Simulation Conference, 2008. ISBN 978-1-4244-1786-5.

[159] Proguard. http://proguard.sourceforge.net, 2002.

[160] MineCraft Servers List. Minecraft servers for online multiplayer gaming. http://minecraft-server-list.com, 2012.

[161] Far Lands or Bust Charity. About: Far lands or bust. http://farlandsorbust.com/about.html, 2012.

[162] Mohcine Chraibi, Armin Seyfried, and Andreas Schadschneider. Generalized centrifugal-force model for pedestrian dynamics. *Phys. Rev. E*, 82:046111, october 2010.

[163] Seth N. Hetu and Gary Tan. Perennial simulation of a legacy traffic model: Implementation, considerations, and ramifications. In *WSC '11: Proceedings of the 43rd Conference on Winter Simulation*. Winter Simulation Conference, 2011.

[164] OpenStreetMap.org. Open street map: The free wiki world map. http://www.openstreetmap.org, 2010.

[165] E. Atkociunas, R. Blake, A. Juozapavicius, and M. Kazimianec. Image processing in road traffic analysis. *Nonlinear Analysis: Modelling and Control*, 10(4):315–332, 2005.

[166] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[167] T. N. Tan, G. D. Sullivan, and K. D. Baker. Efficient image gradient-based object localisation and recognition. In *Computer Vision and Pattern Recognition*, 1996.

[168] S. Lin, K. Chao, and C. Lo. Service-oriented dynamic data driven application systems to traffic signal control. In *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, pages 3463–3470. Kudret Press and Digital Printing, 2010.

[169] K. Hewage and J. Ruwanpura. Optimization of traffic signal light timing using simulation. In R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *WSC '04: Proceedings of the 2004 Conference on Winter Simulation*, pages 367–372. Winter Simulation Conference, 2004.

[170] O. Barnich and M. Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6): 1709–1724, 2010.

[171] S. Perreault and P. Hebert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9):2389–2394, 2007.

[172] Sylvia K. Plevritis. Decision analysis and simulation modeling for evaluating diagnostic tests on the basis of patient outcomes. *American Journal of Roentgenology*, 185(3):581–590, 2005.

[173] G. C. Critchfield and K. E. Willard. Probabilistic analysis of decision trees using Monte Carlo simulation. *Medical Decision Making*, 6(2):85–92, 1986.

[174] Seth N. Hetu and Gary Tan. Application of symbiotic decision support to managed evacuation studies using a perennial framework. In *Asia Simulation Conference 2011*. Korea Society for Simulation (KSS), 2011.

[175] Diane Gross. How thorough is your evacuation program? Disaster Resource Guide, 2012. URL http://www.disaster-resource.com/index.php?option=com_content&view=article&id=1721%3Ahow-thorough-is-your-evacuation-program.

[176] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L. Ananda. Indriya: A low-cost, 3D wireless sensor network testbed. In Thanasis Korakis, Hongbin Li, Phuoc Tran-Gia, and Hong-Shik Park, editors, *Testbeds and Research Infrastructure. Development of Networks and Communities - 7th International ICST Conference,TridentCom 2011, Shanghai, China, April 17-19, 2011, Revised Selected Papers*, pages 302–316. Springer, 2011.

[177] Timothy C. Germann, Kai Kadau, Jr. Ira M. Longini, and Catherine A. Macken. Mitigation strategies for pandemic influenza in the United States. Proceedings of the National Academy of Sciences, 2006.

[178] Joseph T Wu, Steven Riley, Christophe Fraser, and Gabriel M Leung. Reducing the impact of the next influenza pandemic using household-based public health interventions. PLoS Medicine, 2006.

[179] EasySen. SBT80 multi-modality sensor board for TelosB wireless motes, 2007. URL http://www.easysen.com/support/SBT80v2/DatasheetSBT80v2.pdf.

[180] Thiago Teixeira, Gershon Dublon, and Andreas Savvides. A survey of human-sensing: Methods for detecting presence, count, location, track, and identity. 2010.

[181] Singapore Civil Defence Force. Fire code 2007 master version (chapter 1), 2007. URL http://www.scdf.gov.sg/content/scdf_internet/en/building-professionals/publications_and_circulars/fire_code_2007master_version.html.

[182] IEEE Std. 1516-2010. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)– framework and rules, 2010.

Appendix A

Library EvacNET Specification

A.1 Generic Model Template

```
! Library model, no hazards
!
! Change maximum number of time periods allowed for egress
SYS
1
120
END
!
! Change how long a time period is, in seconds
SYS
3
2
END
!
! Change how model's name appears
SYS
5
LIBRARY MODEL
END
!
! Suppress option summaries
SYS
6
0
END
!
! Name of model file
SYS
7
LIBREG.MOD
END
!
! Name of results file
SYS
```

```
8
LIBREG.RES
END
!
!
! Starting node, second floor
EN
RM15.2,274,35
END
!
! Remaining second floor nodes
EN
RM1.2,55
RM2.2,8
RM3.2,8
RM4.2,8
RM5.2,27
RM6.2,55
RM7.2,47
RM8.2,17
RM9.2,23
RM10.2,48
RM11.2,8
RM12.2,59
RM13.2,17
RM14.2,11
SW1.2,8
SW2.2,2
SW3.2,5
SW4.2,5
END
!
!First floor rooms, including destinations
EN
RM1.1,98
RM2.1,54
RM3.1,73
RM4.1,55
RM5.1,54
RM6.1,45
RM7.1,117
RM8.1,22
RM9.1,22
RM10.1,40
SW1.1,8
SW2.1,2
SW3.1,5
SW4.1,5
DS1.1
DS2.1
END
!
!
! Arc definitions, second floor
EA
```

```
RM1.2-RM15.2,2,9
RM15.2-RM1.2,2,9
RM5.2-RM15.2,5,7
RM15.2-RM5.2,5,7
RM6.2-RM15.2,2,9
RM15.2-RM6.2,2,9
RM10.2-RM15.2,10,7
RM15.2-RM10.2,10,7
RM7.2-RM15.2,5,4
RM15.2-RM7.2,5,4
RM9.2-RM15.2,1,7
RM15.2-RM9.2,1,7
RM12.2-RM15.2,6,4
RM15.2-RM12.2,6,4
RM13.2-RM15.2,1,6
RM15.2-RM13.2,1,6
RM1.2-RM14.2,2,2
RM14.2-RM1.2,2,2
RM13.2-RM14.2,2,1
RM14.2-RM13.2,2,1
RM13.2-RM12.2,2,5
RM12.2-RM13.2,2,5
SW1.2-RM3.2,3,1
RM3.2-SW1.2,3,1
RM6.2-RM7.2,2,5
RM7.2-RM6.2,2,5
RM8.2-RM7.2,2,4
RM7.2-RM8.2,2,4
RM8.2-RM9.2,2,2
RM9.2-RM8.2,2,2
SW2.2-RM11.2,1,1
RM11.2-SW2.2,1,1
RM1.2-RM2.2,3,2
RM2.2-RM1.2,3,2
RM1.2-RM5.2,3,3
RM5.2-RM1.2,3,3
RM2.2-RM3.2,3,1
RM3.2-RM2.2,3,1
RM4.2-RM3.2,3,1
RM3.2-RM4.2,3,1
RM4.2-RM6.2,3,2
RM6.2-RM4.2,3,2
RM5.2-RM6.2,3,3
RM6.2-RM5.2,3,3
SW3.2-RM13.2,3,1
RM13.2-SW3.2,3,1
SW4.2-RM8.2,3,1
RM8.2-SW4.2,3,1
RM12.2-RM11.2,3,5
RM11.2-RM12.2,3,5
RM10.2-RM11.2,3,3
RM11.2-RM10.2,3,3
RM10.2-RM9.2,3,4
RM9.2-RM10.2,3,4
END
```

```
!
!
! Arc Definitions, stairs
EA
SW1.1-SW1.2,2,4
SW1.2-SW1.1,2,4
SW2.1-SW2.2,1,4
SW2.2-SW2.1,1,4
SW3.1-SW3.2,1,3
SW3.2-SW3.1,1,3
SW4.1-SW4.2,1,3
SW4.2-SW4.1,1,3
END
!
!
! Arc definitions, first floor
EA
RM7.1-RM5.1,11,2
RM5.1-RM7.1,11,2
RM5.1-RM1.1,8,3
RM1.1-RM5.1,8,3
SW3.1-RM3.1,3,3
RM3.1-SW3.1,3,3
SW4.1-RM2.1,3,3
RM2.1-SW4.1,3,3
RM1.1-DS1.1,6,2
RM3.1-RM7.1,9,4
RM7.1-RM3.1,9,4
RM3.1-RM5.1,6,3
RM5.1-RM3.1,6,3
RM9.1-RM3.1,3,3
RM3.1-RM9.1,3,3
RM4.1-DS2.1,1,2
SW1.1-RM8.1,3,1
RM8.1-SW1.1,3,1
SW2.1-RM6.1,1,2
RM6.1-SW2.1,1,2
END
!
!
!
%%%% TEMPLATE{HAZARD NODES} %%%%
!
!
!
!
! Run, output
!
RUN
EXAM
1
3
S
SW1.2-SW1.1
3
```

```
S
SW2.2-SW2.1
3
S
SW3.2-SW3.1
3
S
SW4.2-SW4.1
3
S
RM1.1-DS1.1
3
S
RM4.1-DS2.1
END
QQ
```

A.2 Hazard Template: Control Set

```
!
!
!Arc definitions that will be modified by hazard 1
EA
RM7.1-RM6.1,3,6
RM6.1-RM7.1,3,6
RM7.1-RM2.1,3,5
RM2.1-RM7.1,3,5
END
!
!
!Arc definitions that will be modified by hazard 2
EA
RM9.1-RM8.1,3,2
RM8.1-RM9.1,3,2
RM4.1-RM3.1,1,6
RM3.1-RM4.1,1,6
RM8.1-RM10.1,1,6
RM10.1-RM8.1,1,6
RM10.1-RM1.1,1,7
RM1.1-RM10.1,1,7
END
```

A.3 Hazard Template: Hazard Set 1

```
!
!
!Arc definitions that will be modified by hazard 1
EA
RM7.1-RM6.1,1,14
```

```
RM6.1-RM7.1,1,14
RM7.1-RM2.1,1,13
RM2.1-RM7.1,1,13
END
!
!
!Arc definitions that will be modified by hazard 2
EA
RM9.1-RM8.1,3,2
RM8.1-RM9.1,3,2
RM4.1-RM3.1,1,6
RM3.1-RM4.1,1,6
RM8.1-RM10.1,1,6
RM10.1-RM8.1,1,6
RM10.1-RM1.1,1,7
RM1.1-RM10.1,1,7
END
```

A.4 Hazard Template: Hazard Set 2

```
!
!
!Arc definitions that will be modified by hazard 1
EA
RM7.1-RM6.1,3,6
RM6.1-RM7.1,3,6
RM7.1-RM2.1,3,5
RM2.1-RM7.1,3,5
END
!
!
!Arc definitions that will be modified by hazard 2
EA
RM8.1-RM10.1,1,6
RM10.1-RM8.1,1,6
RM10.1-RM1.1,1,12
RM1.1-RM10.1,1,12
END
```

Appendix B

Complete UML Diagram

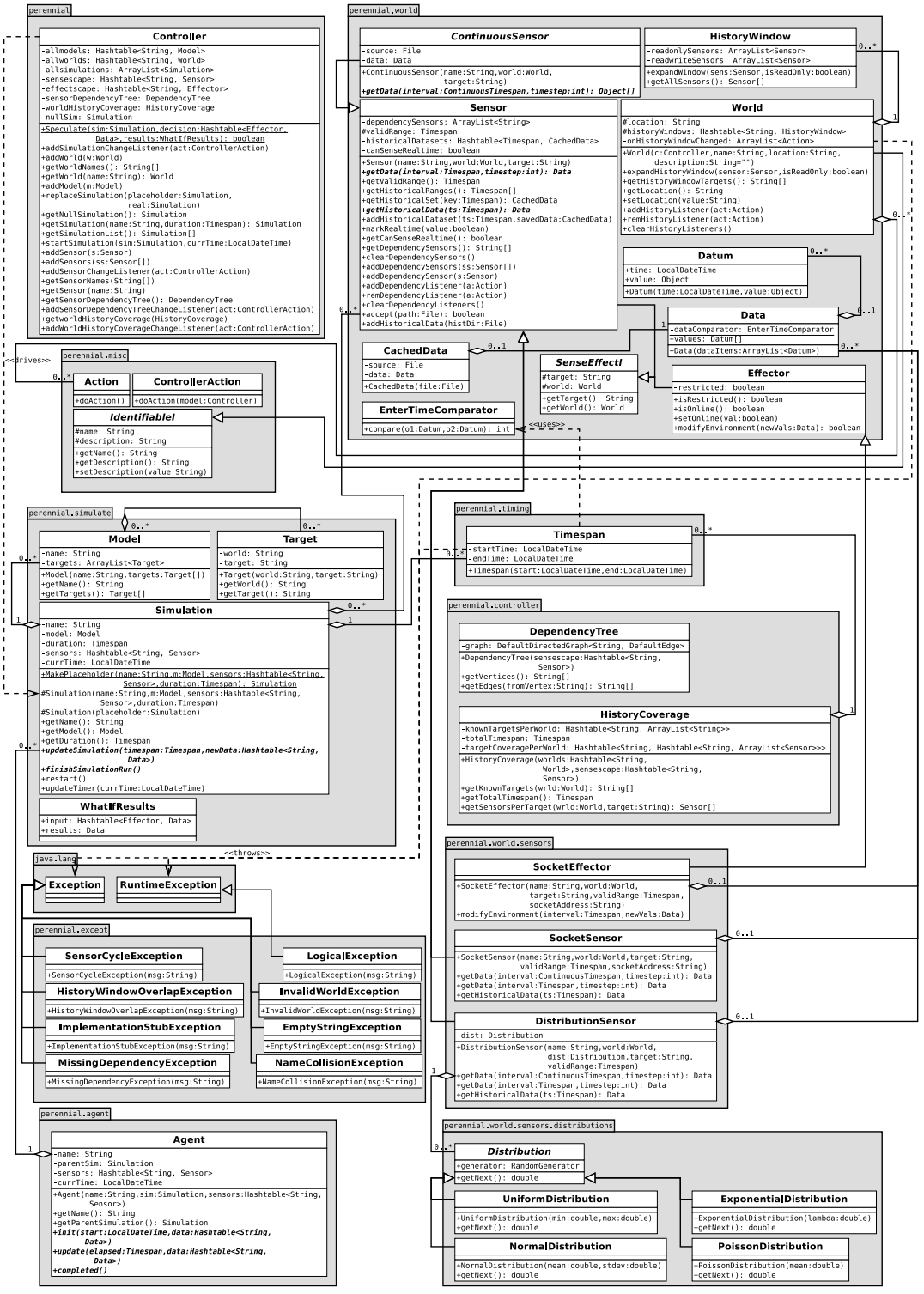

FIGURE B.1: The full UML diagram for the perennial implementation. To increase clarity, controller *aggregation* links and several other obvious links are not shown.

Appendix C

COM1 EvacNET Specification

```
! COM1, first floor model, empty rooms
!
! Change maximum number of time periods allowed for egress
SYS
1
120
END
!
! Change how long a time period is, in seconds
SYS
3
2
END
!
! Change how model's name appears
SYS
5
COM1 MODEL
END
!
! Suppress option summaries
SYS
6
0
END
!
! Name of model file
SYS
7
COM1.MOD
END
!
! Name of results file
SYS
8
COM1.RES
END
!
```

```
! Non-replicated rooms.
EN
RM1.1,135
RM2.1,85
RM3.1,184
RM4.1,178
RM5.1,52
RM6.1,35
RM7.1,130
DS1.1
DS2.1
DS3.1
DS4.1
END
!
!Replicated rooms
EN
RM11.1,15
RM12,15
RM13.1,15
RM14.1,15
RM15.1,15
RM16.1,15
RM21.1,48
RM22.1,48
RM23.1,48
RM24.1,48
RM25.1,48
RM26.1,48
RM27.1,48
RM31.1,81
RM32.1,81
RM41.1,48
RM42.1,48
RM43.1,48
RM44.1,48
RM45.1,48
RM46.1,48
RM47.1,48
RM48.1,48
END
!
!
! Arc definitions, non-replicated rooms
EA
RM2.1-DS2.1,2,6
DS2.1-RM2.1,2,6
RM2.1-RM3.1,2,2
RM3.1-RM2.1,2,2
RM3.1-DS1.1,1,2
DS1.1-RM3.1,1,2
RM3.1-RM4.1,1,4
RM4.1-RM3.1,1,4
RM3.1-RM5.1,2,5
RM5.1-RM3.1,2,5
```

```
RM3.1-RM6.1,2,4
RM6.1-RM3.1,2,4
RM6.1-RM7.1,2,7
RM7.1-RM6.1,2,7
RM6.1-DS4.1,1,4
DS4.1-RM6.1,1,4
RM7.1-RM5.1,2,7
RM5.1-RM7.1,2,7
RM5.1-RM1.1,2,4
RM1.1-RM5.1,2,4
RM1.1-DS3.1,3,4
DS3.1-RM1.1,3,4
END
!
!
! Arc Definitions, replicated rooms
EA
RM11.1-RM2.1,2,6
RM12.1-RM2.1,2,6
RM13.1-RM2.1,2,6
RM14.1-RM2.1,2,6
RM15.1-RM2.1,2,6
RM16.1-RM2.1,2,6
RM2.1-RM11.1,2,6
RM2.1-RM12.1,2,6
RM2.1-RM13.1,2,6
RM2.1-RM14.1,2,6
RM2.1-RM15.1,2,6
RM2.1-RM16.1,2,6
RM21.1-RM6.1,2,4
RM22.1-RM6.1,2,4
RM23.1-RM6.1,2,4
RM24.1-RM6.1,2,4
RM25.1-RM6.1,2,4
RM26.1-RM6.1,2,4
RM27.1-RM6.1,2,4
RM6.1-RM21.1,2,4
RM6.1-RM22.1,2,4
RM6.1-RM23.1,2,4
RM6.1-RM24.1,2,4
RM6.1-RM25.1,2,4
RM6.1-RM26.1,2,4
RM6.1-RM27.1,2,4
RM31.1-RM6.1,2,4
RM32.1-RM6.1,2,4
RM6.1-RM31.1,2,4
RM6.1-RM32.1,2,4
RM41.1-RM7.1,2,4
RM42.1-RM7.1,2,4
RM43.1-RM7.1,2,4
RM44.1-RM7.1,2,4
RM45.1-RM7.1,2,4
RM46.1-RM7.1,2,4
RM47.1-RM7.1,2,4
RM48.1-RM7.1,2,4
```

```
RM7.1-RM41.1,2,4
RM7.1-RM42.1,2,4
RM7.1-RM43.1,2,4
RM7.1-RM44.1,2,4
RM7.1-RM45.1,2,4
RM7.1-RM46.1,2,4
RM7.1-RM47.1,2,4
RM7.1-RM48.1,2,4
END
!
!
!
!
!
! Run, output
!
RUN
EXAM
! Exam depends on actual statistics required.
END
QQ
```

Appendix D

Simplified Building Monitor Prototype

```
//Start the GUI if you want live feedback:
MainFrame f = new MainFrame("config.txt");

//Create your sensor classes:
class VL_Sensor extends SocketSensor {
  VL_Sensor(String name, World world, Target target, Timespan validRng,
            String moteID)
  {
    //Save connection settings.
  }

  Data getData(Timespan interval, int tstep)
  {
    //e.g., access serialForwarder of a mote
    //via TCP; track value via MySQL db, etc.
    CurrVal = /* access Indriya */;
    ArrayList res = new ArrayList<Data.Datum>();
    res.add(new Data.Datum(interval.getStart(), new Double(currVal)));
    return new Data(res);
  }
}
class IR_Sensor extends SocketSensor {
  IR_Sensor(String name, World world, Target target, Timespan validRng,
            String moteID)
  {
    //Save connection settings.
  }

  Data getData(Timespan interval, int tstep)
  {
    //e.g., access serialForwarder of a mote
    //via TCP; track value via MySQL db, etc.
    CurrVal = /* access Indriya */;
    ArrayList res = new ArrayList<Data.Datum>();
```

```
    res.add(new Data.Datum(interval.getStart(), new Double(currVal)));
    return new Data(res);
  }
}
class MIC_Sensor extends SocketSensor {
  MIC_Sensor(String name, World world, Target target, Timespan validRng,
             String moteID)
  {
    //Save connection settings.
  }

  Data getData(Timespan interval, int tstep)
  {
    //e.g., access serialForwarder of a mote
    //via TCP; track value via MySQL db, etc.
    CurrVal = /* access Indriya */;
    ArrayList res = new ArrayList<Data.Datum>();
    res.add(new Data.Datum(interval.getStart(), new Double(currVal)));
    return new Data(res);
  }
}

//Create Higher-Order sensor classes:
class PresenceSensor extends SocketSensor {
  PresenceSensor(String name, World world, Target target, Timespan validRng,
                 Hashtable<String, Sensor> src)
  {
    //Save the "src" sensors (VL, IR, etc.), e.g.:
    this.ir = src.get("ir");
  }

  Data getData(Timespan interval, int tstep)
  {
    //Naive threshold sensor. A proper sensor will
    // buffer values and detect sharp changes.
    ArrayList res = new ArrayList<Data.Datum>();
    int flags =   (getData(vl)>3000?1:0)
                | (getData(ir)>3500?2:0)
                | (getData(mic)>2000?4:0);
    res.add(new Data.Datum(interval.getStart(), new Integer(flags)));
    return new Data(res);
  }
}

//Create Higher-Order sensor classes:
class CombinedPresenceSensor extends SocketSensor {
  CombinedPresenceSensor(String name, World world, Target target, Timespan validRng,
                         Vector<Sensor> src)
  {
    this.roomSensors = src; //At least 1.
  }

  Data getData(Timespan interval, int tstep)
  {
    //Sum all flags
```

```
    int flag = 0;
    for (Sensor src : roomSensors) {
      int curr = getData(src);
      flag |= curr;
    }
    double confidence = ((flag&1)>0?1:0) + ((flag&2)>0?1:0) + ((flag&4)>0?0.5:0);
    ArrayList res = new ArrayList<Data.Datum>();
    res.add(new Data.Datum(interval.getStart(), new Boolean(confidence>1.5)));
    return new Data(res);
  }
}

//Create Controller, Worlds, Targets
Controller ctl = new Controller();
World w = new World(ctl, "COM1", "COM1");
Target hall1 = new Target(w, "hall1");  //etc.

//Instantiate Sensor Objects
Hashtable<String,Sensor> make_sbt80(String s, Target t, World w, Timespan ti)
{
  Hashtable<String,Sensor> res = new Hashtable();
  String name = s+"-vl";
  int mID = /* get mote ID */;
  res.put(name, new VL_Sensor(name, w, t.sub(s).sub("v1"), ti, mID);
  return res;
}

//"sub" targets are new; they allow better space breakdown
Timespan ti = new Timespan("Now", "Inf");
Sensor sens38 = new PresenceSensor("sens38", w, hall1.sub("s38"),
                                   ti, make_sbt80("s38", hall1, w, ti));

//Instantiate Sensor Objects, ctd.
//Initiate remaining sensors
Sensor sens44 = /* same */;
Sensor sens49 = /* same */;
Sensor sens53 = /* same */;

//Note: Historical timespans work too (if data exist)
Timespan ti2 = new Timespan("1/1/2008,0:00", "1/1/2008,1:00");

//Combine them
Sensor sensHall1 = new CombinedPresenceSensor("sHall1", w,
                   hall1, ti, make_vect(sens38, sens44, sens49, sens53));

//Make more sensors
Sensor sensHall2 = /* same */;
Sensor sensBackHall = /* may use an "estimated room presence sensor"
                         for rooms with no actual sensors. */;

//Our "team" location sensors function similarly,
// reporting the position for an agent, then aggregating.

//Create an Agent class
class TeamNotifierAg extends Agent {
```

```
  TeamNotifierAg(String name, Simulation sim, Hashtable<String, Sensor> sensors)
  {} //easy

  void init(LocalDateTime st, Hashtable<String,Data> data)
  {
    this.goal = null;
    this.location = new Point(0,0); //e.g.
  }

  void update(Timespan elaps, Hashtable<String,Data> data)
  {
    if (this.goal == null) {
      this.goal = /* choose a new room to notify. */
    }

    //Naive movement model.
    dyn_vect path = new dyn_vect(location, goal);
    path.scale(this.speed*elaps).translate();
    this.location = path.getPos();
    if (this.location == this.goal) {
      get_room(this.goal).notify();
      this.goal = null;
    }
  }
}

//Creating a Model, Simulation class
class TeamMovementModel extends Model {
  TeamMovementModel(String name, Target[] targets)
  {
    //Our model class is primarily driven by the
    // Simulation and implemented at the Agent level.
  }
}

class TeamMovementSim extends Simulation {
  void init(AraryList<Agent[]> teams, FloorSpec rooms)
  {
    /* save Agent teams locally, init agents */
    /* save FloorSpec locally, partially initialize it */
  }

  void updateSimulation(Timespan ti, Hashtable<String,Data> newData)
  {
    for (Agent[] tm : teams) {
      for (Agent ag : tm) {
        ag.update(newData); //Simple update
      }
    }

    if (/*check if room notified*/) {
      updateRooms(rooms);
    }
  }
}
```

```
//Start the simulation
ctl.addWorld(w);
ctl.addModel(new TeamMovementModel());

//Using "MakePlaceholder()" is a minor design point.
Simulation temp = Simulation.MakePlaceholder("team_sim", makeSensorList(),
                                             null, new Timespan("Now"));

//Make and initialize our simulation.
TeamMovementSim tsim = new TeamMovementSim();
tsim.init(makeAgents(), makeFloorplan());
ctl.replaceSimulation(temp, tsim);

//We would presumably add more simulations, and then:
ctl.startSimulation(tsim, new LocalDateTime());
```