

# Virtual Fusion: The Integration and Analysis of Simulation and Real Processes for Manufacturing Process Deployment

by

William Simeon Harrison

A dissertation submitted in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
(Mechanical Engineering)  
in The University of Michigan  
2011

Doctoral Committee:

Professor Dawn M. Tilbury, Chair  
Professor Jeffrey Stein  
Associate Professor Vineet Kamat  
Associate Research Scientist James Moyne

This work is dedicated to my adviser, Professor Dawn M. Tilbury

## ACKNOWLEDGEMENTS

I would like to thank Rackham Graduate School, the NSF Engineering Research Center for Reconfigurable Manufacturing Systems, NSF grants EEC 95-92125 and CMS 05-28287 for their financial support during my time at the University of Michigan.

I am superbly grateful to Professor Dawn Tilbury for her continued support and unceasing patience. She allowed me to utilize my talents, while helping me to overcome my weaknesses.

I would also like to thank Dr. James Moyne who has been a positive influence on my research from the very beginning. A special thanks should go to Janani Viswanathan for continuing to carry the torch of virtual fusion even after I am gone.

Finally I would like to thank Dhananjay Anand, Serge Gregory, Juil Yum, Josh Langsfeld, Steve Vozar, Adam Brzezinski, Jake Valentic, Yang Zhang, Alex Sobolev, Jeff Fletcher, Ivana Mrazova, Lindsay Allen, Kyle Schroeder, Deepak Sharma and the rest of my family and friends.

\*\*\*\*\*

# TABLE OF CONTENTS

<b>DEDICATION</b> . . . . .	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b> . . . . .	<b>iii</b>
<b>LIST OF TABLES</b> . . . . .	<b>vii</b>
<b>LIST OF FIGURES</b> . . . . .	<b>viii</b>
<b>ABSTRACT</b> . . . . .	<b>xi</b>
<b>CHAPTER</b>	
<b>I. Introduction</b> . . . . .	<b>1</b>
1.1 Contributions . . . . .	3
1.2 Expected Impact . . . . .	4
1.3 Assumptions . . . . .	4
1.4 Dissertation Overview . . . . .	6
<b>II. Background and Related Work</b> . . . . .	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Identified Need for Hybrid Process Simulation . . . . .	8
2.3 Formalisms for Distributed Simulation . . . . .	9
2.4 Formalisms for HIL . . . . .	10
2.4.1 Formal Fidelity Approach to HIL . . . . .	11
2.4.2 Formal Interpretation Approach to HIL . . . . .	12
2.5 Virtual Reality as the Basis for a Conceptual Architecture . . . . .	13
2.5.1 HIL and VR Key Objectives . . . . .	14
2.5.2 Concept of Presence . . . . .	15
2.6 Analysis Methodologies . . . . .	15
2.6.1 Fidelity and Validity . . . . .	15
2.6.2 Validation Methodologies . . . . .	17
2.7 Summary . . . . .	19
<b>III. Hybrid Process Simulation</b> . . . . .	<b>20</b>
3.1 Introduction . . . . .	20
3.2 Simulated and Actual Components: Structuring a Foundation from HIL for HPS . . . . .	21
3.3 HPS Object Oriented Ontology . . . . .	23
3.3.1 Terms . . . . .	24
3.3.2 Existent Superclass . . . . .	27
3.3.3 Entity, Body, and Pathway Subclasses . . . . .	30

3.3.4	Regions, States and Transitions . . . . .	31
3.4	HPS Implementation Methodology . . . . .	35
3.5	Reconfigurable Factory Testbed . . . . .	38
3.5.1	HPS Workpiece Transitions and Sensing . . . . .	39
3.5.2	System Level Controller 1 . . . . .	42
3.5.3	Cell 1 Controller . . . . .	42
3.5.4	CNC and Robot . . . . .	43
3.6	RFT Application . . . . .	43
3.6.1	Simulated Supply Cell Addition . . . . .	45
3.6.2	Simulated Robot Integration . . . . .	48
3.6.3	Simulated Cell Integration . . . . .	49
3.6.4	The Applied Conceptual Architecture . . . . .	50
3.7	Technical Landscape and Limitations for an HPS . . . . .	52
3.7.1	Simulated Components . . . . .	55
3.7.2	Actual Components and Signal Emulation . . . . .	55
3.7.3	Limitations . . . . .	57
3.8	Summary . . . . .	58
<b>IV. Hybrid Process Simulation Equivalence Analysis for Deployment . . . . .</b>		<b>59</b>
4.1	Introduction . . . . .	59
4.2	Equivalence Analysis . . . . .	60
4.2.1	Assumptions . . . . .	62
4.2.2	Pre-Analysis . . . . .	62
4.2.3	Structural Equivalence Analysis . . . . .	66
4.2.4	Leaf Equivalence Analysis . . . . .	67
4.2.5	Communication Equivalence Analysis . . . . .	69
4.2.6	Equivalence Analysis Discussion . . . . .	70
4.3	RFT HPS Equivalence Application . . . . .	71
4.3.1	RFT Equivalence Analysis . . . . .	72
4.3.2	RFT Equivalence Analysis Summary . . . . .	73
4.4	Manufacturing System Emulation Environment Equivalence Application . . . . .	75
4.4.1	MSEE Equivalence Analysis . . . . .	78
4.4.2	MSEE Equivalence Analysis Summary . . . . .	81
4.5	Summary . . . . .	82
<b>V. Autonomous Mobile Entities in Hybrid Process Simulation . . . . .</b>		<b>83</b>
5.1	Introduction . . . . .	84
5.2	Background . . . . .	84
5.2.1	Simulation of Mobile Robotics . . . . .	85
5.2.2	Human Simulation in Manufacturing . . . . .	85
5.3	Autonomous Mobile Entities . . . . .	86
5.4	AME Domain . . . . .	89
5.5	AME Implementation Methodology . . . . .	90
5.6	AME Equivalence . . . . .	93
5.7	AME Example Application . . . . .	95
5.7.1	Applied Implementation Methodology . . . . .	96
5.7.2	Game Engine Implementation . . . . .	97
5.7.3	Digital AME Domain Mapping and Control . . . . .	98
5.7.4	Communication . . . . .	100
5.8	Summary . . . . .	100

<b>VI. Future Technical Landscape, Conclusions, and Future Work . . . . .</b>	<b>102</b>
6.1 Contributions . . . . .	102
6.2 Development of Technical Landscape . . . . .	103
6.3 Resulting Industrial Environment and Impact . . . . .	104
6.4 Future Work . . . . .	104
6.4.1 Unidirectional Mapping . . . . .	105
6.4.2 Parallel Component Monitoring . . . . .	106
6.4.3 Implementation Studies . . . . .	106
<b>Bibliography . . . . .</b>	<b>107</b>

## LIST OF TABLES

### Table

3.1	Possible Values of the Essence and Effect Attributes (non-relevant in italics) . . . . .	29
3.2	The Essences and Effects of the Simulated Supply Cell . . . . .	46
3.3	The Essences and Effects of the Simulated Robot . . . . .	48
3.4	The Essences and Effects of the Simulated Cell (Motoman) . . . . .	51
4.1	A sample comparison table created after structural equivalence analysis . . . . .	67
4.2	Comparison table of the RFT HPS . . . . .	73
4.3	The RFT's HPS terminal bodies and their calculated CNL . . . . .	74
4.4	The pathways within the HPS and their comparison numbers . . . . .	74
4.5	Comparison table of the MSEE . . . . .	78
4.6	The terminal bodies and their calculated comparison numbers . . . . .	80
4.7	The pathways within the MSEE and their comparison numbers . . . . .	81
5.1	Comparison Between Different Types of Components . . . . .	88
5.2	Actual Component . . . . .	94
5.3	Simulated Component . . . . .	94
5.4	The Essences and Effects of the Simulated Human Worker . . . . .	96

## LIST OF FIGURES

### Figure

3.1	A component $C$ of a test setup has several interfaces (three in this example); each is a set of inputs and outputs. . . . .	22
3.2	An HPS consists of multiple components. Some are simulated $\tilde{C}$ and others are a part of the final physical implementation $C$ . The large arrow shows the progression during a system's development from all component simulation, through HPS, to complete physical implementation. . . . .	23
3.3	A hierarchical break down of the classifications of the pieces of an HPS, with the attributes illustrated on the right. <i>Spatial essence</i> , <i>data essence</i> , <i>Newtonian effect</i> , and <i>procedural effect</i> are categories (not values) of the essence and effect attributes	23
3.4	An example of an HPS/HIL, where the real time controller is a simulated component ( $\tilde{C}$ ), and the thermometer and fan are part of the final physical implementation ( $C$ ). 27	27
3.5	The fan body can be written in its UML class diagram form. . . . .	29
3.6	When a simulated <i>Existent</i> is grouped together with other <i>Existents</i> in the same program it is aggregated. . . . .	30
3.7	When a simulated <i>Existent</i> has the same interface as the process or component it represents it is distinct. . . . .	30
3.8	A diagram showing the composition of an HPS. . . . .	31
3.9	The essence and effect of a workpiece are largely dependant on whether the region the workpiece occupies is simulated or actual. . . . .	32
3.10	Domains are comprised of essence and effect. The only direct connection between two different domains is the mapped effects. . . . .	33
3.11	Components of an HPS can exist in different states with different combinations of essence and effect. . . . .	33
3.12	A photo of the serial-parallel line. Cell 2 is on the left and Cell 1 is on the right . .	38
3.13	The control hierarchy of the RFT, where R, CNC, CONV, and C1C stand for robot, CNC machine, conveyor and Cell 1 controller respectively. The numbers in the corner of the blocks specify which cell the entity/body is in. . . . .	39



3.14	The inputs, outputs, and states (a) of the tested portion of the RFT (b), where the parentheses denote the other communicating entity. SLC1 is the version of the SLC specifically designed for the testing covered in this dissertation, and W stands for workpiece. . . . .	41
3.15	When simulated workpieces leave the simulation and enter the physical domain all Newtonian effects are emulated. . . . .	44
3.16	A screen shot of the visualization for the simulated supply cell (SSC) . . . . .	46
3.17	A schematic of the states, inputs, and outputs of the Simulated Supply Cell (SSC), and how it connects to the existing hierarchy. The SSC has no outputs because the SLC1 can see its states . . . . .	47
3.18	A screen shot of the Roboguide visualization for the simulated robot . . . . .	49
3.19	A screen shot of the simulated Motoman . . . . .	50
3.20	A schematic of the states, inputs, and outputs (a) of the Motoman (Moto), and how it connects to the existing hierarchy (b). . . . .	51
3.21	For a completely new process, deployment will start with all non-present components and end in all actual components. An inner box around a component C corresponds to an individual simulation environment. An inner box around multiple Cs corresponds to one simulation environment that includes multiple components. . . . .	53
3.22	An alternative deployment progression to Fig. 3.21. . . . .	53
4.1	Pre-analysis divides a hierarchical control and communication infrastructure (a) into levels and divisions, where E represents an entity, and B represents a body (b). . . . .	63
4.2	Graphs of the equivalences for the RFT component deployment example where $T_a$ represents the time when the simulated supply cell is introduced, prior to which the supply cell is non-present. $T_b$ marks the time when the actual component is installed. . . . .	72
4.3	A directed graph of the version of the RFT used in the HPS (3.17) . . . . .	72
4.4	The preliminary analysis of the HPS used in testing. . . . .	72
4.5	System Structure of the case study from [36]. . . . .	75
4.6	Graphs of the equivalences for the MSEE component deployment example, where $T_a$ signifies the point in time represented by the example. LE and CE are on the same graph to show how CE can lag behind LE. . . . .	76
4.7	The directed graph of the MSEE system. . . . .	78
4.8	The preliminary analysis of the MSEE . . . . .	79
5.1	A four room environment with four controllers that can all request cooling from the fan AME. . . . .	86
5.2	A component AME of a test setup may have one interfaces with inputs and outputs. . . . .	87

5.3	An AME such as an AGV can connect in different locations to the control hierarchy at different times. . . . .	92
5.4	Domain mapping permits the AME to move about its simulated three dimensional space, and have the same effects within the manufacturing process it would have had in the physical domain. . . . .	92
5.5	Examples of how multiple domains can be connected. . . . .	93
5.6	The states, inputs and outputs of the conveyor and the AME. . . . .	96
5.7	Pictures of the Panda3D environment and how it mimics the real environment . . .	98
5.8	The Panda3D domain is bidirectionally mapped to the digital logic domain. . . . .	101
6.1	A screen shot of the RFT in Second Life. . . . .	106

## ABSTRACT

The Hybrid Process Simulation (HPS) approach presented in this dissertation provides a testing and validation platform to support the complete deployment of a manufacturing system. A process planner can start from pure simulation and progressively add real components as they arrive until the entire process is non-simulated. The HPS approach should be able to reduce ramp-up time by allowing system wide troubleshooting to occur before all process components are installed. System-wide operation is enabled by the use of component simulations that can function while interacting with the real process.

The HPS approach is based on Hardware-in-the-loop (HIL) which is a widely used testing approach for embedded systems, where real components and/or controllers are tested in closed-loop with a simulation model. This dissertation generalizes the HIL concept into HPS. An HPS is a test setup that contains at least one simulated and one actual component, but may contain many of both. A conceptual architecture is developed that separates the effect of a component from its spatial essence (volume or mass).

The conceptual architecture is applied to a small manufacturing line at the University of Michigan. A formalized method is then devised for replacing simulations with real processes and vice versa. Application of this method demonstrates how an HPS can be used to test a manufacturing system setup with multiple regions of real and simulated components.

An analysis methodology for quantitatively describing a manufacturing process during deployment, called Equivalence, is developed. The equivalence analysis put forth here has been applied to two manufacturing processes containing both real and simulated components. These examples demonstrate how equivalence analysis can be used as a tool to track and describe deployment when the HPS approach is employed.

Finally the conceptual architecture is expanded to include Autonomous Mobile Entities (AMEs) such as AGVs and human workers. This expansion includes the incorporation of simulation environments capable of simulating the laws of physics. To demonstrate an AME implementation, a game engine was used to create a three dimensional environment in which a simulated AME could move about and interact. This implementation allowed a simulated AME to control a pallet stop release in the real manufacturing process.

## CHAPTER I

### Introduction

Today, manufacturing industries are facing intense global competition because of shorter product life cycles, higher quality standards, and cost constraints. To address these challenges, ongoing research is investigating technologies that facilitate product launch with shorter ramp-up times and lower costs. One focus of this investigation is into the deployment phase of the manufacturing system. Here, a manufacturing system refers to the set of components (conveyors, robots, CNCs, etc.), controllers, and communication connections that make up a process. The deployment phase includes the times when process simulation, component installation, programming, and process validation are taking place. Simulation is one part of deployment that has experienced rapid progress in recent years. Simulation enables better decision making in planning and more accurate performance analysis in the early development stages of deployment.

Simulation, though widely utilized, has historically been a disjointed step unconnected to the rest of the deployment phase. Additionally, validation and testing using simulation is very reliant on the accuracy of the simulation models, and a single processor based simulation may not be powerful enough to simulate all the components including the connections between the components accurately.

The integration of simulation with actual components can begin to address these shortcomings. Integrating simulation with actual components for control testing is more accurate than simulation alone, since it includes some of the same components that will be used in the final process. Testing with actual components makes model accuracy and model availability less of an issue, while also reducing process setup time and the overall cost of deployment.

The integration of simulation and actual components for the purpose of testing is termed Hardware-in-the-Loop (HIL). HIL makes it possible to leverage the strengths of simulation while also reducing the drawbacks that come from a lack of model accuracy. HIL has been applied to a wide variety of applications [49]; however, the number of applications could be expanded further if the method were formalized. The majority of HIL applications have an ad hoc approach to integrating real and simulated processes. Typical HIL applications are also limited to only one region of actual (the controller) and one region of simulation [29]. This two region setup is not ideal for systems that would benefit from a distributed approach, with multiple regions of simulated and actual components such as a manufacturing system.

Hybrid Process Simulation (HPS) is an approach to manufacturing system deployment that was developed to address the shortcomings of the typical HIL approach. HPS is a formalized approach that allows an arbitrary number of simulated and actual components. It can be applied during deployment allowing the manufacturing process to transition from complete simulation to a real process with all actual components. This transition takes place by systematically replacing individual component simulations with the actual components in a step by step fashion. The involved component simulations allow system wide troubleshooting to happen much earlier, before the actual components are installed.

## 1.1 Contributions

The HPS approach and the analysis methodologies associated with it are presented as contributions in this dissertation. The contributions assume a technical landscape where simulated components can be obtained from the vendors of the actual machines and controllers that will be present in the final system. The four contributions are outlined below.

The first contribution is an ontology for describing a manufacturing system during its deployment phase. The presented ontology provides a conceptual architecture for translating a manufacturing system into predefined conceptual components, from which it will be possible to deduce information about the system in its current state of deployment.

The second contribution is a formalized methodology for replacing or integrating simulations with actual processes and vice versa. The methodology outlines a step by step procedure for both integrating components into existing system and deploying a process from scratch.

The third contribution is an analysis methodology for a manufacturing system in its deployment phase. The result of this analysis is a quantitative description that provides a concise understanding of an HPS's percentage of simulation as it progresses to a production ready non-simulated system.

The fourth contribution is a framework for incorporating autonomous mobile entities, such as automatic guided vehicles or humans at manual stations, into manufacturing Hybrid Process Simulations. The framework fits within the ontology and analysis methodology of Chapter 3.

The contributions in ontology and analysis are applied in different scenarios on a

small manufacturing line at the University of Michigan's Engineering Research Center for Reconfigurable Manufacturing Systems. These scenarios include the following: replacing a robot, replacing a manufacturing cell (two CNCs and 1 robot), adding a manufacturing cell (overhead gantry), and integrating an autonomous mobile entity into the HPS. These scenarios illustrate the proposed technology and describe how it can directly benefit manufacturing system design and implementation.

## **1.2 Expected Impact**

The aforementioned contributions create an approach to manufacturing system deployment that can decrease ramp-up time and overall cost. If adopted, manufacturing process designers will know more about the real components they are going to purchase by working with the simulated versions of the components they are considering. Process planners will also be able to transfer training and code done with the simulated component directly to the real component, because they will be designed by the same vendor. Additionally, when the system is online the entire process can be run without workpieces, allowing another level of validation.

If the approach outlined in this dissertation is followed, future deployments will incorporate the use of many component simulations representing the diversity of equipment not just in the actual implementations, but in the simulation phase as well, making the transition from simulated components to real much more seamless.

## **1.3 Assumptions**

In order to develop the framework and analysis methodologies for HPS, a number of assumptions are made. First, this dissertation only considers discrete manufacturing systems that are composed of a set of distinct components and controllers that process individual workpieces; continuous processes (such as chemical processes) are



not considered. Here, the term components refers to the machines in the manufacturing process capable of transporting or processing workpieces. Examples include gantries, assembly robots, and CNCs. Components that do not process distinct parts such as paper production machines or other continuous flow machines are not addressed.

Components of the manufacturing system considered here do not change their overall location and have one physical connection (wired) to the rest of the manufacturing system. Higher level controllers coordinate groups of controllers and/or components (see Fig. 3.13). It is assumed that controllers and components can receive commands from only one other controller. This assumption means that the control hierarchy can be represented as a tree structure where all nodes have a maximum of one parent node.

The manufacturing systems addressed in this dissertation can be modularized by separating controllers and components at defined interfaces (Fig. 3.1). The manufacturing system's components and controllers can then be simulated by individual component and controller simulations that can be created or are already available.

There are some assumptions that specifically pertain to the simulated components themselves. The first is that the component simulations are of a high enough quality, meaning they do not prohibit the HPS from functioning correctly by having incorrect internal functionality or not acting within reasonable time frames. Secondly, the applied component simulations are assumed to have the same behavior as the real components they represent, where the behavior is defined by the signals sent and received at the interface of the component. Thirdly, the signals at the interface of the component are strictly those associated with event based control, and, therefore, the bandwidth required for the simulations communication is the same as that required

for the real components. Physical interfaces such as those that connect components using force and velocity are not considered. Lastly, it is assumed that component simulations are realtime, meaning that signal behavior is observed to occur in time intervals comparable to those of the real components.

#### **1.4 Dissertation Overview**

The organization of this dissertation is as follows: Chapter 2 gives a background of HIL and HIL formalisms. It also gives a literature review of analysis methodologies that apply to HIL type applications. Chapter 3 presents the HPS ontology for describing a manufacturing system during deployment. Chapter 4 covers an analysis methodology for a manufacturing system in its deployment phase. Chapter 5 is a relaxing of particular assumptions to create a framework for incorporating autonomous mobile entities into a manufacturing Hybrid Process Simulation. Finally, Chapter 6 addresses conclusions and future work.

## CHAPTER II

# Background and Related Work

Manufacturing process development typically requires a planning and simulation phase followed by a component installation phase. Hardware-in-the-Loop (HIL) is a testing and validation methodology that can merge these two phases where simulations are connected to real components. HIL approaches, however, are often times ad hoc and lack any type of formal approach or terminology. This chapter first presents the inconsistencies and ambiguities within the HIL literature due to its lack of formalism. It then surveys the literature for formalized approaches that could apply to HIL type applications. Lastly, it surveys the literature for analysis methodologies that could apply to a HIL test setup. The analysis methodology should quantify how the HIL test setup compares to the actual non-simulated system. Parts of this chapter have been published in [29].

### 2.1 Introduction

The lifespan of a manufacturing system can be divided into multiple phases, which include the following: 1) planning 2) system wide deployment and 3) production. The planning stage will typically include a simulation analysis. Manufacturers use simulation softwares like Arena, Demo3d, Tecnomatix, and Delmia [18] for operational and technical planning [64]. Though these simulations have proven to be very useful in

industry, they are all-in-one inclusive proprietary solutions that do not represent the diversity of component solutions present in a manufacturing processes. Additionally, they do not permit the distributed structure required by an HPS.

After the initial simulation analysis is completed, deployment (physical implementation) can begin, and will continue until the non-simulated manufacturing system is fully deployed. Here the term *non-simulated manufacturing system* is used to refer to the final production-ready system. Deployment is considered here to be the process by which all installation of the actual (real and physical) components is accomplished, and the production phase is when the process is functioning and creating product. The production phase of the manufacturing process can also be followed by a component deployment phase for reconfiguration and new component installation.

The deployment phase is the focus of this dissertation and is important because it is where a manufacturer faces a great deal of pressure. Shortening deployment times can result in higher profit margins and longer product life cycles, but conversely can also mean overspending for implementation and the deployment of an inefficient processes which will cut into profits [4, 50]. There exists, however, no formal method for analysis of a manufacturing system during the deployment phase. This dissertation puts forth an approach to formalizing the deployment process called HPS.

## **2.2 Identified Need for Hybrid Process Simulation**

The literature supports a general understanding of HIL, but within this understanding there are some variations which point to the need for a clear ontology. A large portion of the literature considers HIL to only be for controller testing. In this case HIL is considered to be when “control hardware is interfaced to the computer simulation and receives inputs from the simulation...” [66]. Another definition

says that “HIL simulation involves modeling the plant hardware being controlled, and interfacing the model with the intended controller” [60]. All of the above papers and many others are committed to the idea that HIL is generally for testing controllers [27, 34, 39, 45, 48, 57].

In addition to the control-testing oriented applications of HIL, there is a more general understanding that includes but is not limited to controllers. One such definition is that “HIL is a scheme that incorporates hardware components of primary concern in the numerical simulation environment” [34]. Another definition says that HIL is “a combination of real time process simulation connected to a subsystem or component hardware” [59]. These definitions are more general, and define HIL as a testing methodology that may consist of many different types of components, both real and simulated.

In addition to the differences in the definition of HIL, the literature also has confounding classifications of the terms *software* and *hardware*. One of HIL’s original uses was for electronic control units (ECUs) [49]. ECUs tested with HIL have well-defined software and hardware components, where software represents the simulation and hardware represents the physical controller. Software, however, is not always synonymous with simulation, and hardware is not always synonymous with the final physical implementation. For example, an engine connected to a dynamometer could be considered an HIL test setup by some of the above definitions; however, the dynamometer is hardware which is part of the simulation.

### **2.3 Formalisms for Distributed Simulation**

HIL test setups are distributed simulations in a manner of speaking. Just like the typical distributed simulations, a HIL simulation must interface externally and

be synchronized with the component it is connected to. For this reason distributed simulation is a good place to search for a formalism applicable to HIL.

Distributed simulation was adopted because it could result in reduced execution time, geographically distributed computation, and integrated simulations models from different vendors [24, 52]. The defense community has long since embraced the distributed approach and has come up with a standard called the high-level architecture (HLA) [15]. HLA is the official standard for the Department of Defence and an accepted IEEE standard. HLA, though widely utilized in defense, and intended for industry [17], has not been widely accepted by industry [46]. To address the absence of an HLA type standard in industry, research has been done to devise methodologies for integrating dissimilar simulations in a distributed environment [42, 53]. The research has not resulted in a widely accepted methodology however.

## 2.4 Formalisms for HIL

HIL is a mature field which has been around for 40 years, “ever since the aerospace industry became safety-critical” [7]. Though there are a large number of academic and industrial applications using HIL, it lacks a formal approach which would provide a conceptual foundation.

The existing literature approaches HIL formalism from two perspectives; they are described here as *fidelity* and *interpretation*. HIL fidelity addresses how well the test setup describes the system or process it represents, oftentimes on a component-by-component basis, and taking an input-output or transfer function view of the components. The fidelity approaches that exist in the literature focus primarily on a particular application. HIL interpretation considers the structure of the HIL set-up itself, often focusing on communication between different pieces of the system, and

usually with less specification on the internal behavior of the components. The main benefit of the interpretation approach is its potential for a broad class of applications, as opposed to the fidelity approaches which have a more focused set of applications. Although the HPS ontology presented in this dissertation is a formal interpretation, formal fidelity is also summarized, since it comprises the bulk of the work that has been published on formalization of HIL.

#### 2.4.1 Formal Fidelity Approach to HIL

Bacic suggested a method for assessing HIL test setup fidelity using the concept of transparency [7]. A perfectly transparent system is one in which real components do not feel a difference between the real environment and the HIL test setup. Bacic also focused on the interface between the simulation and the real components, and the “robustness of prediction”. However, the definitions were not well-developed enough to be applied to a broad class of problems.

In a follow-on work, MacDiarmid *et al.* provided a detailed approach on how to quantify the accuracy of an HIL test setup [48]. The approach was limited to simulations that could be implemented as linear time-invariant differential equations and used an input-output approach to bound the difference in outputs between the actual and simulated system.

Gawthrop *et al.* showed how to connect mechanical components to an HIL test setup in a way that compensates for dynamic effects of the actuators and sensors used to monitor mechanical devices [25]. The authors provided a sound method for including mechanical components in an HIL test setup, but again use a transfer function perspective.

Boer *et al.* discuss using a backbone architecture called First All Modes All Sizes (FAMAS) that was originally designed to connect different simulations in a

distributed architecture [16]. Real components are connected to this backbone, creating an HIL test setup. Though this work formalizes the data sharing for HIL, it is limited to FAMAS and does not provide any insight into the HIL setup itself.

The PABADIS‘PROMISE project used an ontology to develop a novel approach to flexible manufacturing system design and execution [3]. The project also integrated several actual and simulated components for testing purposes, but no general approach or methodology for HIL was presented [2].

In the area of pure distributed simulation for manufacturing the authors of [38, 42, 51–53, 62] address fidelity by specifically looking at the interoperability of different software applications. Their goal was to create a platform for simulation interoperability testing; they did not consider an approach for integrating actual and simulated components.

An HIL fidelity approach specific to mechatronic systems was used by [40, 58]. Fidelity approaches for testing actual and simulated components in manufacturing systems were considered by [19, 35, 36]. All of these approaches, however, require a custom interface for communication between simulated and actual components.

#### **2.4.2 Formal Interpretation Approach to HIL**

Several researchers have developed frameworks or approaches for defining the interconnections between real and virtual pieces in an HIL system; these are grouped under the formal interpretation approach to HIL. Many of these frameworks focus on the communication protocol, and several require a centralized system-wide communication infrastructure that connects all of the different pieces of the system.

Auinger *et al.* put forth a comprehensive approach to interpretation formalism [5]. They designed an architecture from the characteristics and requirements of what they call Reality in the Loop (RIL) and Soft Commissioning. They specifically outline



a set of requirements on the simulation model in an HIL test setup. For example, requirement 1 (R1) states that the simulation of the plant (machines, robots, etc) should be separate from the simulations of the control environment. Requirement 3 (R3) states that a visualization should be separate from the physical model. R3 goes on to say that “the visualization should be attachable to a specific behavioral model (and therefore to its specific level of abstraction)”. Their example, however, is a typical HIL application with a single simulated plant and a real controller (PLC).

The Manufacturing System Emulation Environment (MSEE) put forth by [36] is a manufacturing process with both simulated and actual components. [36] makes a distinction between simulation and emulation, where “manufacturing system simulation means to create certain conditions of manufacturing systems by means of models”, and “a manufacturing system emulator is a device or piece of software that enables a program or an item of equipment intended for one type of computer or equipment to be used with exactly the same results with another type of computer or equipment”. The MSEE approach also makes use of a middle layer that allows communication between all involved simulated and actual components.

Some of the existing work could be considered to address both formal interpretation and formal fidelity. For example, [7] could be interpreted as an early attempt at the interpretation approach, however, the focus was on measuring transparency and robustness of prediction. [16] also attempts an interpretation approach but requires the customized FAMAS backbone for communication.

## **2.5 Virtual Reality as the Basis for a Conceptual Architecture**

HIL as put forth thus far serves as a testing methodology for hardware equipment and control code. Here, the connection is described between the basic idea of HIL

and Virtual Reality (VR).

### 2.5.1 HIL and VR Key Objectives

Manufacturing simulation and VR are approaching a congruence in their application. Simulations are now capable of being portrayed in VR [41]. HIL also shares a congruence with VR's key underlying objective, which is to provide mediated stimuli to the user, such that he/she can respond as if receiving non-mediated stimuli [14]. The mediated stimuli are represented by signals that manifest themselves in an artificial environment, and mimic non-mediated stimuli which originate from the real environment. Similarly, the key underlying objective of HIL is to provide mediated inputs to a controller or component to see how it would respond in a non-mediated setting i.e., with the actual components installed and functioning.

HIL and VR methodologies have two important common features that come from the similarities in their key objectives. Within VR, it is said that “scientists want to better understand psychological and physiological processes as they occur in non-mediated settings” [47]. HIL is also used by engineers or process planners to better understand a process or system during its operation (non-mediated). The other parallel is the interface; the VR interface is most commonly represented by the connection between the user's eyes and ears and the device used to convey the environment. The interface in HIL is between the tested component and the device that creates the simulated scenario. Virtual reality has a well defined interface called the *communication* interface which is paramount in creating a believable experience [14]. The interface in HIL is equally important, it allows the *behavior* to remain consistent between simulated and actual counterparts. When a simulated component is swapped out for its actual counterpart, communication protocols and connections should remain unchanged, and the same signals should be sent between the components in

both directions.

### 2.5.2 Concept of Presence

Because HIL and VR share some common features, it makes sense to leverage underlying concepts from VR and apply them to HIL type applications. One important underlying concept in VR is the idea of presence. Lombard *et al.* says that presence is at the heart of virtual reality [47], it “refers not to one’s surroundings as they exist, but the perception of those surroundings” [14]. Perception in this case can be described as the user’s personal interpretation of the environment.

The concept of presence cannot directly be applied to HIL because it is based on personal interpretation. Machines and humans can receive information about their environment through sensors or senses, but human interpretation is not controllable and machine interpretation is. Therefore, presence must be expanded to a concept that includes, but is not limited to, human-type perception/interpretation.

## 2.6 Analysis Methodologies

During a manufacturing process’s deployment using the HPS approach, the HPS itself will take on different compositions of actual and simulated components. Developing a method of analysis in which the HPS’s composition can be quantitatively described is useful in understanding the current progress toward completion of manufacturing process deployment. Existing fidelity and validity analysis approaches serve as a good starting point for developing such a method.

### 2.6.1 Fidelity and Validity

Validation is a means to assess how good the simulation is at portraying the real system [1, 8]. The validation of pure simulation is chiefly concerned with the simulation’s function (its predictive quality), but not its form (how the simulation is

connected and organized). How the simulation works is not nearly as important as whether the output and performance of the simulation represent the real system to an acceptable (user-defined) degree. The HPS's form can take on many combinations of simulated and non-simulated components, affecting the accuracy and operation of the HPS as a whole, thus the form of an HPS holds as much relevance as its function. Since an HPS's form changes during deployment, a formal approach to describing an HPS's form is needed, but because the HPS approach is new to manufacturing deployment, the literature does not as yet contain work concerning a description of form. The most pertinent related body of work is fidelity and validity, which both concern function. If a formal approach to fidelity or validity exists, and is general enough to apply to different manufacturing systems, this approach could possibly be applied to devising a method of describing the form of an HPS.

Validity and fidelity have similar definitions, however, some do differentiate [56]:

Many confuse fidelity and validity. Fidelity is an absolute measure of simulation relationship to the reality represented (usually described in terms of resolution, accuracy, etc.). On the other hand, validity is a relative measure of whether the fidelity of a simulation is adequate to support its intended use.

Validity and fidelity, however, are not commonly used within the existing discussion of validation [28, 30]. In [13] the introduction mentions validity but only in passing. The vast usage differences between validation and validity points to the difficulty with quantifying the results of a validation study. Validity is the measure where validation is the action. Validation can be done in many ways but coming up with a quantifiable measurement of validity is difficult and thus often times not done at all. Much of the literature suggests that validity is very close to fidelity, so close

in fact that validity would almost seem to be a synonym. This is at least partially contradictory as can be seen by [56]. From this point forth the definitions supplied by [56] will be used for fidelity and validity.

### 2.6.2 Validation Methodologies

The following is a three step approach put forth by [13] that has been widely followed for validation.

1. Face Validity: "... construct a model that appears reasonable on its face to model users and others who are knowledgeable about the real system being simulated."
2. Validation of model assumptions: Assumptions can either be structural or data. "Structural assumptions involve questions of how the system operates and usually involve simplifications and abstraction of reality." "Data assumptions should be based on the collection of reliable data and correct statistical analysis of the data."
3. Validate Input-Output Transformations: Given input-output sets from the real system, compare that to the input-output set created by the simulation. One set should be used for calibration and a different set should be used to validate the simulation after it is calibrated.

Though there is quite a bit of discussion on validation, most approaches take some form of the above [43]. This existing body of work lacks a quantifiable means of assessing fidelity or validity that is general enough to be applied to a large variety of systems [55]. Devising a means of assessment that is general enough to apply to different scenarios is challenging. Methodologies that apply to one simulation are typically hard to apply to another.

Henderson talks about assessing the fidelity of a virtual reality environment [33]. In her description she discusses quantifying the fidelity of the simulated environment by comparing it to how the user is allowed to connect with it. One such example would be the field of view. Human field of view is over 180 degrees but head mounted displays are rarely that high. Henderson suggests comparing these numbers to give a quantifiable sense of simulation fidelity.

Henderson's approach is insightful considering how difficult the quantification of simulation fidelity is. The human user, which is the one commonality in all virtual reality applications, is perfect for basing a quantification of fidelity that is general enough to apply to all virtual reality applications. Quantification can be achieved by comparing two quantifiable characteristics of a particular virtual reality system to the average human user (field of view for example). Any approach for fidelity or validity assessment that can be applied to many different scenarios must leverage commonality on some level. This commonality could come from comparing one part of a system that all applicable systems have in common such as an interface [33]. Commonality could also be achieved by using a particular technique to create a model of the system to which analysis can be performed.

In practice simulation validity often has a more qualitative approach [28], where validity is assessed to be "high, medium, or low" for example. This is likely due to the variance in simulation solutions, and the lack of output data for calibration and validation. For this reason validation primarily relies "on review by experts and peers".

Typical simulation validation approaches are not easily applied to an HPS. Providing inputs to the system and monitoring the outputs, which is a popular form of validation, becomes a much more complex endeavor when the simulations are sepa-

rated by real processes (Fig. 3.2). This complexity is compounded by the evolving nature of an HPS as it develops from all simulated to all non-simulated processes during a manufacturing system's deployment.

## 2.7 Summary

While the literature supports many solutions to different HIL applications with various different drawbacks, we focus here on those that would address the challenges in integrating simulated and real components into an HPS. The focus on integration, as stated in Section 1.3, considers the quality and behavior of the simulated component to be adequate for the HPS application; and thus, the internal details of the simulated components are not considered. With these assumptions in mind, an HPS approach requires a formalized terminology, application approach, analysis methodology, and comparison metrics (relative to the final physical implementation) for manufacturing deployment. A non-ambiguous formalized terminology provides a language that research can build upon in an organized fashion. The application methodology clearly prescribes steps for implementing the approach. The analysis enables an easy to implement assessment of the current progress toward complete physical implementation using clear well defined metrics.

## CHAPTER III

# Hybrid Process Simulation

Hardware-in-the-Loop (HIL), as an existing methodology for testing and validation, lacks a formal approach in both methodology and terminology. HIL test setups often times consist of only one region of real components and one region of simulated components. The term Hybrid Process Simulation (HPS) is presented as a new approach to testing and validation that utilizes the HIL testing style. This chapter first introduces an ontology and conceptual architecture for a manufacturing process during its deployment phase. A methodology for implementation is then prescribed for different deployment situations.

The ontology, conceptual architecture, and implementation methodology are applied to three different HPS scenarios including: adding a manufacturing cell, replacing a robot, and replacing a manufacturing cell. Parts of this chapter have been published in [31], and a journal paper based on this chapter has been conditionally accepted [32].

### 3.1 Introduction

This dissertation presents HPS as a generalization of HIL. HPS has a new conceptual architecture with its own ontology and taxonomy. The ontology is not only generic enough to allow different systems to be translated into conceptual pieces, but



also specific enough to retain pertinent information about the system. The ontology is derived from a common understanding of HIL, but has a well defined terminology to support it. HPS is best suited to support a gradual progression during system deployment, and is applied on the component level of equipment installation. The progression could start with simulation alone, where real components are then gradually added to the simulation, until the system is completely implemented in its final functional form:

$$\boxed{\text{simulation} \rightarrow \text{HIL} \rightarrow \text{physical implementation}}$$

### 3.2 Simulated and Actual Components: Structuring a Foundation from HIL for HPS

Instead of hardware and software, in this dissertation HIL test setups will be defined using the terms *simulated* and *actual* (Def. 1 and 2).

**Definition 1. *Actual:*** *Component or region of a test setup that will be used in the final physical build.*

**Definition 2. *Simulated:*** *Component or region of a test setup that approximates the behavior of its actual counterpart.*

The mixed simulated/actual test setup consists of multiple components, where each simulated component has an actual counterpart in the physical implementation, and each actual component is the same as in the final physical implementation. A component is considered to be any part of the test setup that can be simulated, including controllers and machines. Components of the test setup are connected by

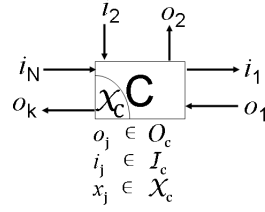


Figure 3.1: A component  $C$  of a test setup has several interfaces (three in this example); each is a set of inputs and outputs.

interfaces (Fig. 3.1) through which signals (physical or electrical) are transmitted. A component can have multiple interfaces, depending on how many other components are connected to it. An interface is a set of signals (inputs and outputs) where the inputs of one component are the outputs of another. The behavior of a component is defined by the signals that it sends and receives (Def. 3). The intention is that simulated portions of the test setup will eventually be replaced with their actual counterparts.

**Definition 3. Behavior:** *The behavior of a component  $C$  of a test setup is a function  $o_C = f_C(i_C, x_C)$ , where  $x_c \in \mathcal{X}_C$  are the states of the component,  $o_c \in \mathcal{O}_C$  are the outputs, and  $i_c \in \mathcal{I}_C$  are the inputs.*

As discussed in Section 2.2, software and hardware do not accurately describe the components of an HIL's progression from simulated components to actual components. An HIL setup often also only includes one simulated region and one actual region. For the preceding reasons the term Hybrid Process Simulation (HPS) is used to replace HIL (Def. 4).

**Definition 4. Hybrid Process Simulation:** *A test setup that contains at least one simulated and one actual component.*

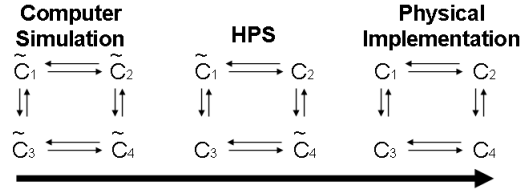


Figure 3.2: An HPS consists of multiple components. Some are simulated  $\tilde{C}$  and others are a part of the final physical implementation  $C$ . The large arrow shows the progression during a system’s development from all component simulation, through HPS, to complete physical implementation.

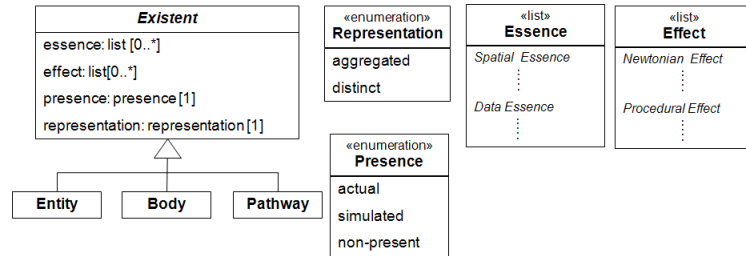


Figure 3.3: A hierarchical break down of the classifications of the pieces of an HPS, with the attributes illustrated on the right. *Spatial essence*, *data essence*, *Newtonian effect*, and *procedural effect* are categories (not values) of the essence and effect attributes

“Hybrid” refers to the mix of simulated and actual components. “Simulation” refers to test setups that have at least one simulated component.

HPS differs from conventional HIL because there can be multiple simulated components in a test setup. Conventional HIL (with typically one simulated component) can be considered a subset of HPS. Fig. 3.2 shows how an HPS exists between complete simulation and the actual system during a development cycle. Each simulated component  $\tilde{C}$  must have the same inputs and outputs (and approximate behavior) as its actual counterpart  $C$ .

### 3.3 HPS Object Oriented Ontology

The object oriented approach to programming can be leveraged as a tool to describe the ontology for HPS and the accompanying equivalence analysis methodologies. The object oriented approach to programming provides a standardized means

to describe structure (how something is organized) and behavior (what it does). This chapter will utilize the approach of [61] for its object oriented ontology. The ontology put forth consists of terminology, relations, and definitions used for describing a manufacturing system. Specifically the chapter will define the abstract components of an HPS.

The characterization of HPS using the Object Oriented (OO) approach will be described from a top-down perspective. The OO approach is characterized by objects and classes. Classes are described as a group of things with the same properties, behaviors, and relationships, where an instance of a class is an object. Fig. 3.3 is the UML diagram for the characterization terminology.

### 3.3.1 Terms

The remainder of the Chapter 3 terms will be presented here. A more detailed discussion of the terms is then presented in the succeeding sections.

**Definition 5. *Existent*:** *An abstract superclass of objects that includes all things in the final manufacturing system such as machines, controllers, and the means of communication between them.*

**Definition 6. *Essence Attributes*:** *A set of characteristics that make an existent object what it is, such as location, tracking number, mass, volume, etc.*

**Definition 7. *Effect Attributes*:** *A set of outcomes stemming from an act that changes or maintains essence, such as change in location, tracking number, mass, volume, etc.*

**Definition 8. *Physical Domain:*** *The complementary pair of Newtonian effect and spatial essence attributes.*

**Definition 9. *Digital Domain:*** *The complementary pair of procedural effect and data essence attributes.*

**Definition 10. *Dual Presence:*** *An entity with both spatial and data essence.*

**Definition 11. *Ghosting:*** *The act of losing spatial essence.*

**Definition 12. *Birthing:*** *The act of gaining Newtonian effect.*

**Definition 13. *Presence:*** *An enumerated attribute that describes how an existent object is implemented. Its possible values are:*

- *Simulated: attribute value which indicates that the existent object is a simulation and is represented in a form other than that which will be present when the entire system is non-simulated.*
- *Actual: attribute value which indicates that the existent object exists and is non-simulated final form.*
- *Non-present: attribute value which indicates that an existent object is not simulated or implemented in any way in the current HPS.*

**Definition 14. *Representation:*** *An enumerated attribute that describes how an existent object is separated from the rest of the system. Its possible values are:*

- *Aggregated*: attribute value which means that the existent with the attribute is located inside of a larger simulation as a function.
- *Distinct*: attribute value which indicates that the existent is distinctly separate and is located in its own environment, in which it communicates by the same interface as its non-simulated counterpart.

The representation attribute is relevant only if the presence attribute of the corresponding existent (Def. 13) is simulated.

**Definition 15. *Entity***: A subclass of objects that can perform actions and/or provide decisions. Two separate entity objects cannot share information without each entity having the recognition and permission of the other entity.

**Definition 16. *Body***: A subclass of objects that can store information and perform actions but cannot make decisions or provide control.

**Definition 17. *Pathway***: A subclass of objects that provide communication between any combination of Entities and Bodies.

**Definition 18. *Mapping***: A mapping is a function from an effect  $a$  in domain  $A$  to another effect  $b$  in domain  $B$  such that whenever  $a$  occurs,  $b$  also occurs. We say that effect  $a$  is mapped to effect  $b$ ; note that effect  $b$  can occur even if effect  $a$  does not occur. We also say that domain  $A$  is mapped to domain  $B$ .

**Definition 19. *Association***: Existent  $a$  in domain  $A$  and existent  $b$  in domain  $B$  are said to be associated if they are acknowledged as being different representations

of the same *existent*.

The terminology will be explained through an example illustrated in Fig. 3.4. In this example a fan receives signals from a controller based on readings from a thermometer. Both the thermometer and the fan have interfaces to the computer. The computer receives realtime temperature information from the thermometer where it can then provide control for the fan based on the information from the thermometer. The example includes five *Existent* objects: the computer, the thermometer, the fan, the connection between the computer and the fan, and the connection between the computer and the thermometer.

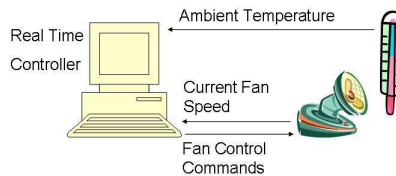


Figure 3.4: An example of an HPS/HIL, where the real time controller is a simulated component ( $\hat{C}$ ), and the thermometer and fan are part of the final physical implementation ( $C$ ).

### 3.3.2 Existent Superclass

The highest level class or superclass in HPS is called an *Existent* (Def. 5). Since an *Existent* is at the highest level, it exists as an abstract class, and therefore all machines, controllers and connections of a manufacturing system fall under the *Existent* superclass in a generalization hierarchy. The *Existent* superclass includes all parts of the manufacturing system, those represented in the present HPS as real components or simulations and those that are not present yet but will be implemented in the non-simulated manufacturing system. The *Existent* superclass has a set of *essence*

attributes (Def. 6) and a set of *effect* attributes (Def. 7).

Two categories of *effect* attributes are considered in this dissertation. The first category encompasses effects from the physical world, which is any effect stemming from physical interaction. Because the effect attributes of physical interaction are echoed in Newton's third law, this type of effect attribute is termed *Newtonian effect*. Newtonian effect attributes are categorized as being a part of the *physical domain* (Def. 8). One effect attribute in the cooling system example in Fig. 3.4 would be the electrical signal sent from the thermometer, or the new ambient temperature of a room cooled down by the fan.

The other category of *effect* attributes encompasses effects from the digital world. Because procedural programming languages focus more on functions and the change of data rather than the data itself, this category of effect attributes is termed *procedural effect*. Procedural effect attributes take place in the *digital domain* (Def. 9). Here the digital domain includes any computer stored information. One procedural effect attribute might be the changing of the temperature stored in the controller.

The existence of an effect suggests a cause. The cause of an effect from the physical point of view involves mass and volume. Mass or volume could then be considered the essence of physical existence. For this reason the *essence* attribute (Def. 6) is complementary to the effect attribute, and also defines the effect attribute. In the physical domain *spatial essence* (mass/volume) is complementary to *Newtonian effect*. In the digital domain *data essence* is complementary to *procedural effect*. The spatial essence attribute is considered to be the physical volume or mass (the actual physical thermometer) representing an existent object in the physical domain. The data essence attribute (the stored temperature information) is considered to be the electronic information representing an existent object in the digital domain. The



Table 3.1: Possible Values of the Essence and Effect Attributes (non-relevant in italics)

Physical Domain		Digital Domain	
Spatial Essence	Newtonian Effect	Data Essence	Procedural Effect
Mass Volume	Wind speed <i>Weight</i> Temperature change <i>Electrical signal</i>	Stored orientation data	Fan status change

fan is an existent object with values for the essence and effect attributes listed in Table 3.1. Table 3.1 has been divided into digital and physical domains for easier understanding but this division is not required for the approach. Fig. 3.5 is the UML class representation of the fan where the domain distinction is not made.

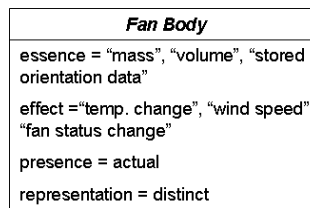


Figure 3.5: The fan body can be written in its UML class diagram form.

Enumerations are a data type with a finite set of values. The Existent superclass also has two enumerated attributes which are *presence* (Def. 13) and *representation* (Def. 14).

If the cooling system was tested with only a thermometer and a controller, the *presence* attribute of the fan would be non-present. This might happen if the fan's effect on the environment and its internal behavior are very well understood and simulating it is not necessary.

*Representation* is an enumerated attribute which applies only to simulated *Exis-*  
*tents*. The representation attribute can be either aggregated or distinct. If an object of the *Existent* superclass is currently located within a larger program with other simulated entities, that simulation's representation attribute is *aggregated* (Fig. 3.6).

If it is separate, then it is *distinct* (Fig. 3.7).

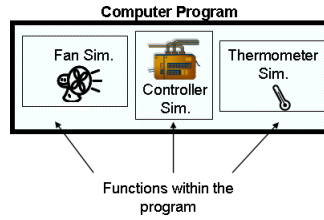


Figure 3.6: When a simulated *Existent* is grouped together with other *Existents* in the same program it is aggregated.

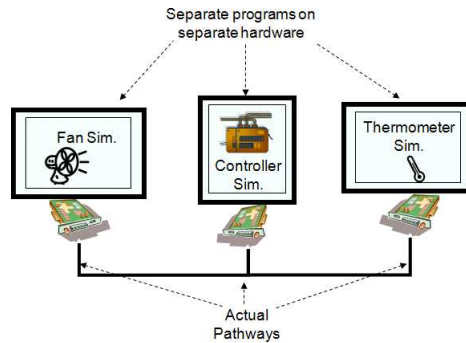


Figure 3.7: When a simulated *Existent* has the same interface as the process or component it represents it is distinct.

One key to the HPS approach is that a simulated component within an HPS can emulate the Newtonian effect attributes which means that the non-simulated components are not affected by the absence of the spatial essence of the simulated component. The term emulate is used here to mean the recreation of an effect. An example might be a simulation of the fan that can communicate and send the same signals that the real fan would send to the controller.

### 3.3.3 Entity, Body, and Pathway Subclasses

The *Existent* superclass has three sub classes which are *Entity* (Def. 15), *Body* (Def. 16), and *Pathway* (Def. 17). All three subclasses inherit the attributes of the *Existent* superclass.

Instances of the *Entity* class serve as the brains of the type of manufacturing sys-

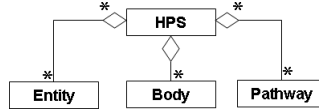


Figure 3.8: A diagram showing the composition of an HPS.

tems considered in this dissertation; they will make all decisions and account for the control. The controller in Fig. 3.4 is an instance of the *Entity* class because it makes the decisions for the system. The thermometer and the fan are not because they have no decision making capabilities, therefore, they are instances of the *Body* class. The connections between the controller, thermometer, and fan are the pathways. An HPS is composed of *Entities Bodies and Pathways*. Fig. 3.8 shows the aggregation of the three subclasses.

### 3.3.4 Regions, States and Transitions

An HPS potentially has many different simulated and actual regions. Actual regions may exist in both domains, and computer simulated regions only exist in the digital domain. In addition to the simulated and actual regions of an HPS, the workpiece itself exists as an actual or simulated entity. Regardless of whether a workpiece starts as being actual or simulated, it should be capable of moving through any region of an HPS (Fig. 3.9). A workpiece can also exist in one or more domains. The *state* of an entity is considered to be the domains (essence and effect) in which it exists. Additionally, the workpieces may transition between regions that occupy different domains.

Workpieces that transition between regions must change their state (essence and effect) to match the domain of their surroundings. An entity can transition in and out of the physical domain numerous times; however, after it transitions into the physical domain, it may not gain spatial essence but it can regain Newtonian effect.

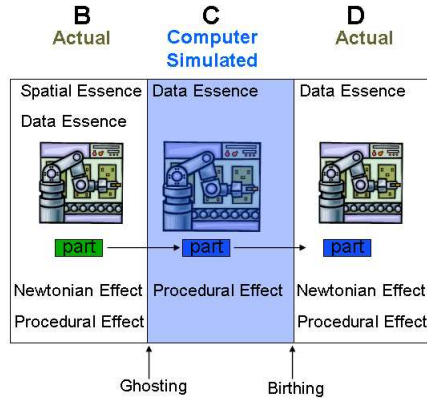


Figure 3.9: The essence and effect of a workpiece are largely dependant on whether the region the workpiece occupies is simulated or actual.

To illustrate the connection between a workpiece's state and its surroundings, consider a manufacturing HPS. Assume that the workpiece will interact with various different machines in the process as it moves down a single track serial line. The essence and effect of the workpiece depend on the region it currently occupies. Fig. 3.9 shows how a process could start with a region containing all actual physical machines, followed by a region with all simulated machines. These regions alternate as the workpiece moves down the line. When the workpiece is in the physical world it may exist in both the physical and the digital domain, and have both Newtonian and procedural effect as well as data and spatial essence, which is termed *dual presence* (Def. 10). Spatial essence and data essence are connected through association only (Fig. 3.10). A tracking number (data essence) may electronically represent a part (spatial essence), but its connection is not intrinsic. It is accepted that the number is connected; however, that connection is strictly an agreed upon association (Def 19). Effects between domains, however, can be directly connected which is called mapping (Def. 18). If the effects in domain A are mapped to effects in domain B, then domain B serves as a passive portrayal of what is happening in domain A. If effects in domain A are mapped to effects in domain B, and effects in domain B are mapped to effects

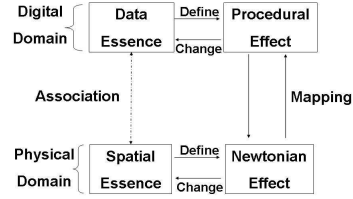


Figure 3.10: Domains are comprised of essence and effect. The only direct connection between two different domains is the mapped effects.

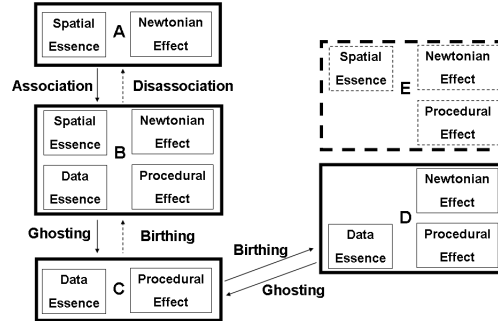


Figure 3.11: Components of an HPS can exist in different states with different combinations of essence and effect.

in domain A then the two domains can mutually affect one another.

When the workpiece is in a computer simulation, however, it will only exist in the digital domain because it cannot have physical essence in a computer simulation. The transition into the simulated region with a loss of spatial essence is termed *ghosting* (Def. 11). When the workpiece reaches the next region in the test setup where it interacts with components that have physical essence, it still will not have spatial essence but it should regain Newtonian effect. The transition from the simulated region back into the actual region with Newtonian effect is called *birthing* (Def. 12).

When a workpiece leaves a computer simulation and goes through birthing into the actual system's physical domain (Fig. 3.9 region C to D), the workpiece must be simulated and must still interact with the physical domain. The simulated workpiece can interact with the physical domain because it will have Newtonian effect (Fig. 3.9 and Fig. 3.11, region D). Because the workpiece maintains Newtonian effect,

the other components of the system are unaware of the workpiece's lack of spatial essence. This type of Newtonian effect is accomplished through signal emulation, allowing all entities that interact with the physical workpiece to have the same exact interactions with the simulated workpiece. An example of such an interaction might be a workpiece presence sensor that must sense both the physical workpiece and the simulated workpiece. The simulated workpiece has Newtonian effect in the physical domain, because the simulated workpiece activates the same sensor signal effect a real workpiece would activate.

As a workpiece moves through an HPS it may have many different states. Fig. 3.11 shows the possible states of a workpiece in an HPS. The workpiece may gain or lose essence and/or effect as it transitions from one region (simulated or actual) to another. Dashed lines represent uncommon states or transitions, and the solid arrows represent the most common transitions in an HPS. The solid arrow from region A to B in Fig. 3.11 represents an entity that gains essence and effect in the digital domain, which might happen by giving a product in a delivery system a bar code or tracking number; the bar code is then connected to the physical workpiece through association. The solid arrow in Fig. 3.11 from region B to C represents ghosting, which is a transition where an entity leaves the physical domain. This transition happens when an entity goes from an actual physical region of an HPS to a completely computer simulated region. The dashed arrow in Fig. 3.11 from region C to B represents a transition where an entity in the digital domain also enters the physical domain. A physical test workpiece would need to be added at the boundary between the two regions and would likely come from something that is not a regular part of the system (e.g. a person in charge of validation). The dashed arrow from region B to A in Fig. 3.11 represents the disassociation of an entity's digital and

physical domain. An example of this disassociation might be after a workpiece is complete, and no longer needs individual tracking. The workpiece is still physically present and will be delivered, but its individual information holds no value. Region E in Fig. 3.11 represents an entity that can effect the digital domain through mapping but has no representation in the digital domain. Region E is the only dashed lined state because it is not a useful state for validation purposes, and thus no transitions to or from it are shown.

### 3.4 HPS Implementation Methodology

Two possible scenarios in which the HPS approach can be applied are discussed here. In scenario 1, which starts with complete simulation, actual components/entities replace their simulated counterparts iteratively until the system itself becomes actual. Because HPS can be applied to test setups including machines, controllers, and many other types of components, the term entity will be used generically. In scenario 2, a simulated component either replaces an actual entity in a test setup, or is added to an existing actual test setup. After the HPS has been validated, the simulated component can be replaced with its actual counterpart as in scenario 1.

Scenario 1 starts from a test setup that is either all simulation or a mix of simulation and actual components. The simulations must have the capability of being connected to actual components. The HPS approach cannot be applied to a simulation that is completely stand-alone with no outside connections.

To replace a simulated entity with an actual one, first several steps must be completed:

1. Identify the entities involved in the process
2. Define the essence and effect of each entity

3. Define the boundaries that outline the regions of the test setup that will be replaced; regions can contain one or more entities
4. Define the behavior of each entity

After the above have been defined, the following steps are repeated until the process becomes all actual.

1. Choose the simulated entity that will be replaced with the actual entity.
2. Acquire the actual entity (hardware, software, and/or components).
3. Remove or deactivate the simulated entity where the actual entity will be added.
4. Connect the actual entity to the test setup.
5. Modify the setup as needed to accommodate both the actual and simulated entities so that the HPS runs properly. For example, the actual entities may need sensor signal emulation so that they function correctly with workpieces that have no spatial essence.
6. Run the test setup for validation.
7. If more simulated entities remain, go to step 2, or, if validation fails, return to step 5.

Scenario 2 starts from a test setup that has some or all actual entities. A simulated entity will replace part of or be added to the test setup. The simulated entity represents a potential change to the existing test setup; after validation with the new simulated entity, it will be replaced with its actual counterpart. The simulated entity must have the capability of being connected to the actual test setup in the same manner as its actual counterpart.

To replace an actual entity by adding a simulated entity, the following steps are used.



1. Identify the entity to be replaced.
2. Define the essence and effect of the simulation that will be added.
3. Define the behavior of the entity.
4. Define the interface (logical and physical).
5. Acquire or build the simulation.
6. Remove or deactivate the actual entity where the simulation will be placed.
7. Connect the simulation to the test setup.
8. Modify the setup as needed to accommodate both the actual and simulated entities so that the HPS runs properly. For example, the actual entities may need sensor signal emulation so that they function correctly with workpieces that have no spatial essence.
9. Run the test setup for validation.

After the validation is complete, when the actual change is ready to be made, go to scenario 1 step 2. If the validation process cannot be completed successfully, the chosen simulated component may not be a viable option for replacement. Another component simulation with different characteristics can be tested in place of the previous until a viable option is reached. In addition, if the component simulation lacks some fidelity or functionality that the actual component will contain, the validation with the HPS may not be successful.

One variation of scenario 2 is to add a simulated entity to the existing test setup instead of replacing one. For this variation the first step becomes:

1. Identify the entity to be added.

All other steps remain the same with the exception of step 6, which can be skipped.

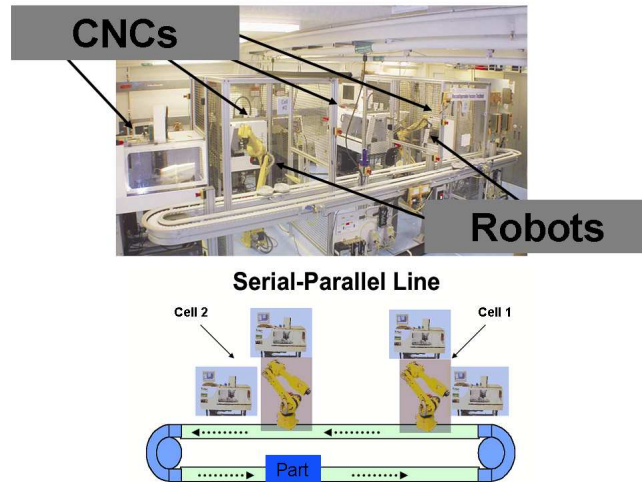


Figure 3.12: A photo of the serial-parallel line. Cell 2 is on the left and Cell 1 is on the right

### 3.5 Reconfigurable Factory Testbed

The concepts and methodologies presented in this dissertation are demonstrated on the Reconfigurable Factory Testbed (RFT). The RFT is located in the Engineering Research Center for Reconfigurable Manufacturing Systems at The University of Michigan. The RFT is designed to make small trains out of two pieces of wax, with the letter M carved on the front. The RFT’s physical components include two robots, four CNC machines, and one conveyor, which make up the serial-parallel line (Fig. 3.12). The robots and CNCs are equally divided into two cells (Cell 1 and Cell 2) (Fig. 3.13), which are coordinated by the System Level Controller (SLC).

The RFT represents a process that is initially a completely *actual* manufacturing system. The RFT becomes an HPS in the testing scenarios presented below. In these scenarios, *simulated* cells and components are integrated into the existing process. The entities (blocks in Fig. 3.13) in the RFT will be abstracted and modeled as discrete event systems.

The term “workpieces” will refer to the pieces of wax, and the regions and com-

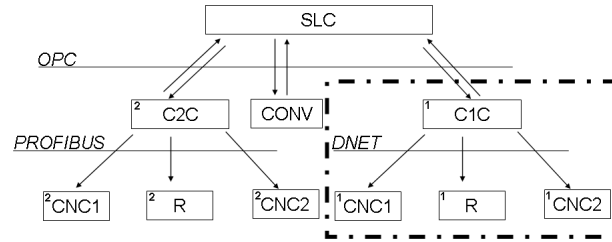


Figure 3.13: The control hierarchy of the RFT, where R, CNC, CONV, and C1C stand for robot, CNC machine, conveyor and Cell 1 controller respectively. The numbers in the corner of the blocks specify which cell the entity/body is in.

ponents in the RFT will be identified by name (such as Cell 1 or CNC1). Because Cell 1 and Cell 2 are very similar, the focus of the implementation is on Cell 1. Entities in Cell 1 include the following: wax workpiece (W), Cell 1 controller (C1C), CNC machine 1 (CNC1), CNC machine 2 (CNC2), robot (R).

It is assumed in the RFT example that the controllers can see the internal states of the CNCs and robots because the C1C as originally written [21] tracks their internal states. An explicit communication interface with inputs and outputs could also be used. The following subsections will define the behavior, interface, essence, and effect of the existing entities in Cell 1.

### 3.5.1 HPS Workpiece Transitions and Sensing

One challenging aspect of an HPS is dealing with entities that must travel from simulated regions to actual regions (Fig. 3.15). Workpieces must be processed by different regions of the manufacturing system whether they are actual regions in the physical domain or computer simulated regions in the digital domain. Fig. 3.15 shows sensors at the boundaries between a computer simulated region and an actual region. These sensors activate workpiece emulation (in the physical domain) when a workpiece crosses the boundary from a computer simulated region to an actual region. The sensors must be dual present (existing both in the simulated and actual

region) because simulated workpieces must trip the sensor when they go from a simulated region to an actual region, and actual workpieces must trip the sensor when they go from an actual region to a simulated region.

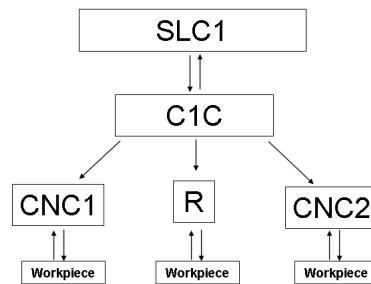
Dual presence sensors work by triggering the appropriate procedural effect when they are activated, regardless of whether the workpiece is simulated or physically present. On the RFT when a physical sensor is activated (Newtonian effect of a pallet stop sensor) by a physical pallet, the robot, for example, is alerted (procedural effect) and begins its pick and place operation. When the robot and the sensor are simulated, the physical (spatial essence) robot is no longer present but the physical sensor remains. When the physical sensor is activated (Newtonian effect) so too is the simulated sensor (procedural effect) alerting the simulated robot to begin its pick and place operation. The link between the physical and simulated sensor is a mapped effect (Def. 18). When the simulated workpiece is placed near the simulated sensor by the simulated robot after processing is complete, the same procedural effect as that caused by the physical sensor will occur. The location of the dual presence sensor is also where ghosting and birthing occur.

Wax trains are the end product of the RFT. Initially each wax train starts as two blocks (workpiece 1 and workpiece 2). Each wax block set is processed in series in each cell. If the workpiece has no spatial essence just before it reaches Cell 1 the Newtonian effects must be emulated.

The Newtonian effect of a workpiece in Cell 1 is sensed by an inductive presence sensor located on the robot gripper. The emulated signals come from a positive 24V applied to the robot presence sensor inputs.

C1C	SLC1	R	CNC
<b>States (<math>x_{C1C}</math>)</b>	<b>States (<math>x_{SLC1}</math>)</b>	<b>States (<math>x_R</math>)</b>	<b>States (<math>x_{CNC}</math>)</b>
<ul style="list-style-type: none"> <li>• Empty</li> <li>• Part 1 Cutting</li> <li>• Part 2 Cutting</li> <li>• Part 1 Loading</li> </ul>	<ul style="list-style-type: none"> <li>• Part 2 Loading</li> <li>• Part 1 Unloading</li> <li>• Part 2 Unloading</li> </ul>	<ul style="list-style-type: none"> <li>• Part 1 in Cell 1</li> <li>• Part 2 in Cell 1</li> <li>• Both Parts in Cell 1</li> </ul>	<ul style="list-style-type: none"> <li>• Robot Idle</li> <li>• Loading Part 1</li> <li>• Loading Part 2</li> <li>• Unloading Part 1</li> <li>• Unloading Part 2</li> </ul>
<b>Outputs (<math>o_{C1C}</math>)</b>	<b>Outputs (<math>o_{SLC1}</math>)</b>	<b>Outputs (<math>o_R</math>)</b>	<b>Outputs (<math>o_{CNC}</math>)</b>
<ul style="list-style-type: none"> <li>• Cut Part (CNC1 and CNC2)</li> <li>• Load Part 1 (R)</li> <li>• Load Part 2 (R)</li> <li>• Part 2 Done (SLC1)</li> </ul>	<ul style="list-style-type: none"> <li>• Unload Part 1 (R)</li> <li>• Unload Part 2 (R)</li> <li>• Part 1 Done (SLC1)</li> </ul>	<ul style="list-style-type: none"> <li>• Process Part 1 (C1C)</li> <li>• Process Part 2 (C1C)</li> </ul>	<ul style="list-style-type: none"> <li>• Part 1 in Process</li> <li>• CNC Idle</li> </ul>
<b>Inputs (<math>i_{C1C}</math>)</b>	<b>Inputs (<math>i_{SLC1}</math>)</b>	<b>Inputs (<math>i_R</math>)</b>	<b>Inputs (<math>i_{CNC}</math>)</b>
<ul style="list-style-type: none"> <li>• Process Part 1 (SLC1)</li> <li>• Process Part 2 (SLC1)</li> <li>• Part Done (CNC1 and CNC2)</li> </ul>	<ul style="list-style-type: none"> <li>• Part 1 Done (C1C)</li> <li>• Part 2 Done (C1C)</li> </ul>	<ul style="list-style-type: none"> <li>• Load Part 1 (C1C)</li> <li>• Load Part 2 (C1C)</li> <li>• Unload Part 1 (C1C)</li> <li>• Unload Part 2 (C1C)</li> </ul>	<ul style="list-style-type: none"> <li>• Cut Part (C1C)</li> </ul>

(a)



(b)

Figure 3.14: The inputs, outputs, and states (a) of the tested portion of the RFT (b), where the parentheses denote the other communicating entity. SLC1 is the version of the SLC specifically designed for the testing covered in this dissertation, and W stands for workpiece.

### 3.5.2 System Level Controller 1

The SLC1 is the coordinator for the entire process. The behavior of SLC1 is a function  $o_{SLC1} = f_{SLC1}(i_{SLC1}, x_{C1C}, x_{SLC1})$  where  $o$ ,  $x$ , and  $i$  are discrete states and events as described in Fig. 3.14. The SLC1 (used for HPS testing) only has one interface. The interface input and output signals are communicated through OPC. The SLC1 exists in the *digital domain* (Fig. 3.11; its *data essence* is the program code and its *procedural effects* are the output commands given in Fig. 3.14).

### 3.5.3 Cell 1 Controller

The C1C, which acts as the coordinator for the robot and the CNC machines, will remain in its *actual* form throughout testing. The behavior of the C1C is a function  $o_{C1C} = f_{C1C}(i_{C1C}, x_{CNC1}, x_R, x_{CNC2}, x_{C1C})$  where  $o$ ,  $x$ , and  $i$  are given in Fig. 3.14. The C1C does not have inputs from the CNCs and robot because it can see their internal state. The highest level interface is to the SLC, where the communication is done through OPC. The lower level communications to the CNCs and the robot are over DeviceNet.

The C1C exists in the *digital domain*; however, some of its *procedural effects* are mapped to *Newtonian effects* (Fig. 3.11, region D). This mapping is for the communication between the C1C and the CNCs. Mapping is needed because the CNCs receive 24V signals for control. These signals are different from the normal C1C's DeviceNet communication signals. The voltage signals require a conversion from the digital bit data of the DeviceNet signal to a voltage, which is handled by a DeviceNet I/O block. Similar to the SLC, the C1C's *data essence* is the program code and its procedural effects are the output commands (given in Fig. 3.14).

### 3.5.4 CNC and Robot

The CNCs are responsible for cutting the wax blocks, and the robot is responsible for transporting the wax blocks between the conveyor and the CNCs. The CNCs will remain in their *actual* form throughout testing, but the *actual* robot will be replaced by a *simulated* robot. The behaviors of the CNC and the robot are functions  $o_{CNC} = f_{CNC}(i_{CNC}, x_{CNC}, x_{workpiece})$  and  $o_R = f_R(i_R, x_R, x_{workpiece})$  respectively where  $o$ ,  $x$ , and  $i$  are given in Fig. 3.14. The state of the workpiece  $x_{workpiece}$  is conveyed through the workpiece’s Newtonian effects.

The CNCs and robot exist in the *physical domain* completely at first. Similar to the C1C, their *Newtonian effects* are mapped to *procedural effects*. When the CNC is finished cutting and the gripper opens, the “process done” command is then sent to the C1C. The open gripper is a *Newtonian effect*, but the command being sent is a *procedural effect*. The CNCs and robot differ from both the C1C and the SLC because they have *spatial essence*. The spatial essences are the actual physical machines, and the effects/outputs are covered in Fig. 3.14.

## 3.6 RFT Application

In this section the HPS ontology and the formalized application method put forth in this chapter are applied to the RFT [54]. The HPS test setups addressed in this chapter represent three variations of scenario 2 (starting with a mostly actual system and replacing parts of it with simulated entities). Scenario 2 is chosen because it shows the highest contrast between the HPS approach and common practices within manufacturing. Customarily, if modifications of an existing process line were required, process planners would not implement realtime “plug and play” simulations. They would instead buy the equipment and/or software/hardware directly. Buying

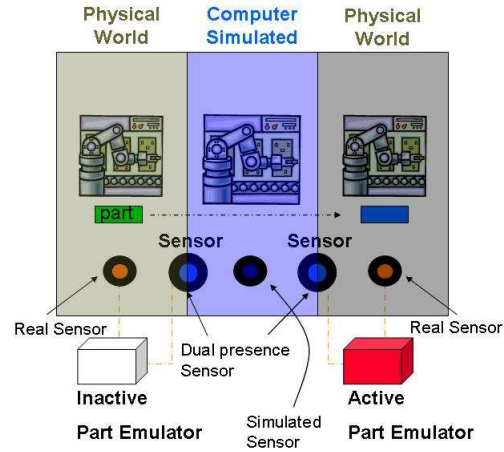


Figure 3.15: When simulated workpieces leave the simulation and enter the physical domain all Newtonian effects are emulated.

the actual components directly eliminates the ability to evaluate different solutions using different simulations, and also could result in damage to personnel and equipment due to unexpected errors. Scenario 2 is also chosen because, once finished, it then becomes scenario 1 (replacing a simulated entity with an actual entity), step 6, which is the step the process planner would already have to accomplish.

To apply HPS concepts, the following three testing scenarios were applied to the RFT: the actual Cell 1 robot was replaced with a simulated robot, the actual Cell 1 was replaced entirely with a simulated cell, and a simulated automated supply cell was added to the hierarchy. These three case studies (replacing a component, replacing a cell, and adding a new cell) demonstrate many of the capabilities of an HPS.

The simulated supply cell and the simulated cell that replaced cell 1 use behavioral models instead of mathematical or stochastic simulation models. The behavioral model provides the correct signal response given a specific set of signal inputs; it is not a detailed model based on equations of motion. The behavioral models have geometric and kinematic descriptions that exist in a game engine environment. The



screen shots in the following sections are examples of what the model’s visualizations look like in their functioning environment. The simulated robot, however, is a simulation software package obtained from the vendor (Fanuc Robotics) of the component it represents. For this reason it is a much higher fidelity simulation with program code that is directly portable to the actual Fanuc robot. Understandably, detailed simulation models give a more accurate prediction of system performance for various scenarios. From the standpoint of a proof of concept, however, individual component simulation fidelity, though important, is not the focus of this dissertation.

### **3.6.1 Simulated Supply Cell Addition**

The simulated cell addition will be covered in detail because it includes all of the important principles of an HPS. Adding a simulated cell requires signal emulation and “plug and play” testing. The simulated cell addition also includes birthing an entity which is the most challenging of the transitions (Definition 12).

Adding the simulated cell represents a scenario in which a process planner may want to automate a loading process that is currently manual. The process planner can then test with simulation different supply cells manufactured by different companies. Testing allows the process planner to make a more informed decision while also decreasing the total amount of time for integration.

This situation corresponds with scenario 2, where the system starts in an actual form with physical processes in place and follows the approach described in Section 3.4.

1. **Identify the entity to be added.** The simulated supply cell (SSC) was added to create an automated loading station for the workpiece. (Fig. 3.16 shows the visualization of the supply cell).



Figure 3.16: A screen shot of the visualization for the simulated supply cell (SSC)

Table 3.2: The Essences and Effects of the Simulated Supply Cell

Simulated Supply Cell		
Physical Domain	Spatial Essence	<i>None</i>
	Newtonian Effect	<i>None</i>
Digital Domain	Data Essence	Java simulation
	Procedural Effects	Outputs and states listed in Fig. 3.17

2. **Define the essence and effect.** The SSC is an overhead gantry that exists completely in the *digital domain*, which means that it has data essence which is the program itself, and its procedural effects are its changes in state. (Table 3.2).
3. **Define the behavior of the entity.** The SSC receives requests from the SLC. The SSC starts the process by transporting a workpiece that only exists in the digital domain (Fig. 3.11, region C) (scenario 2, step 3). The behavior of the SSC is a function  $o_{SSC} = f_{SSC}(i_{SSC}, x_{SSC})$  where  $o$ ,  $x$ , and  $i$  are given in Fig. 4.3.
4. **Define the interface.** The SSC has an OPC interface.
5. **Build the simulation.** The SSC was built using both an open source game engine (Blender3d) and a signal emulator written in Java.
6. **Deactivate the actual entity.** Because this is an addition there is no need for deactivation.
7. **Connect the simulation to the test setup.** The SSC was connected using the same OPC interface as the existing cells.

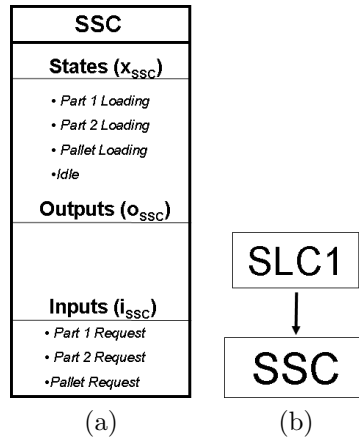


Figure 3.17: A schematic of the states, inputs, and outputs of the Simulated Supply Cell (SSC), and how it connects to the existing hierarchy. The SSC has no outputs because the SLC1 can see its states

8. **Modify the system as needed, such that it can function correctly within the larger system and handle the transition from simulated workpiece to actual workpiece.** The workpiece leaves the SSC (Fig. 3.9, region C to D) where it is transported by the conveyor to Cell 1 (Fig. 3.9, region C). The transition out of the simulation into the physical domain means that birthing has occurred. The workpiece never had *spatial essence*, but it must now have *Newtonian effect*. Newtonian effect is accomplished through workpiece emulation.
9. **Run the test setup for validation.** The addition of the SSC did require some modifications to be made to the SLC ( scenario 2, step 9). These are the same changes that will be needed when the actual supply cell is added. The integration of the actual supply cell should then be faster because the control logic is already validated to work with the SSC.

Table 3.3: The Essences and Effects of the Simulated Robot

<b>Simulated Robot</b>		
Physical Domain	Spatial Essence	<i>None</i>
	Newtonian Effect	<i>None</i>
Digital Domain	Data Essence	Roboguide simulation
	Procedural Effects	Outputs and states listed in Fig. 3.14a

### 3.6.2 Simulated Robot Integration

Replacing the real robot with a simulation represents scenario 2, where a manufacturing process planner detects a malfunctioning robot. The planner then has the option to purchase a new robot with similar functionality. With the HPS approach in mind, the planner could acquire multiple simulated robots (from different vendors) and compare and contrast them before making a final selection. The steps for the approach are the same as those for the simulated supply cell in Section 3.6.1.

The simulated robot has no *spatial essence*; it exists solely in the *digital domain* (Fig. 3.18 shows the visualization of the simulated robot) and has the *procedural effects* that match the inputs and outputs of its actual counterpart (Table 3.3). The Newtonian effects (gripping the workpiece) are replaced by procedural effects that allow the simulated robot to interact with an emulated workpiece.

During the integration of the simulated robot, no changes were made to the hierarchy or the control logic. The system functionally behaved as if the robot was real. This testing setup means that any scenario run on the system should be a good indicator of how the real system (with emulated workpieces) will behave.

The simulated robot differs from the other simulated components in that the simulation is from the vendor of the actual component (Fanuc Robotics). Fanuc has a simulation software called Roboguide that simulates the motion as well as



Figure 3.18: A screen shot of the Roboguide visualization for the simulated robot

the control structure and interface. Programs written in Roboguide can be ported directly from the simulation to a real Fanuc robot.

The Roboguide simulation software is proprietary and thus its source code is closed. Proprietary restrictions are typical of most simulation software, in that component vendors do not want competing vendors to have access to detailed information about their product. Fanuc does however supply a PC development kit (PCDK) that makes it possible to communicate with Roboguide. [65] describes how PCDK was used to write an interface between the DeviceNet card and Roboguide, allowing the simulation to be connected over DeviceNet in exactly the same fashion as the actual Fanuc robot.

### 3.6.3 Simulated Cell Integration

Replacing the real cell with a simulated cell also represents scenario 2, where a process planner may want to try a very different manufacturing solution. The solution in this case is to replace the two CNCs and one robot with a single robot. The new approach may be more expensive, but it could improve the overall system performance (throughput, quality, etc.). The steps to accomplish the integration are the same as those outlined in Section 3.6.1 for the SSC. Using the simulated cell

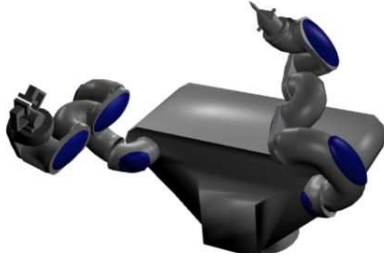


Figure 3.19: A screen shot of the simulated Motoman

allows testing of the control logic as well as the communication protocol. Testing will reduce both risk and the time for integration.

The emulated Motoman DA20 was developed to replace Cell 1 (Fig. 3.19 shows the visualization of the Motoman). The Motoman is a 13 degree-of-freedom two-armed robot that could potentially replace both CNCs and the existing robot by holding the workpiece with one arm and processing it with the other. The Motoman's inputs and outputs are identical to the C1C, therefore, the SLC1 remains unchanged and unaware of the switch. The Motoman was added to the RFT by disconnecting the Cell 1 controller from the SLC1. Disconnection was done by removing the Ethernet cable that enables OPC communication. The Motoman was then connected over Ethernet, using the same OPC outputs and inputs as the C1C.

Like the simulated robot, the Motoman exists purely in the *digital domain* and does not have Newtonian effect. The behavior of the Motoman is a function  $o_{moto} = f_{moto}(i_{moto}, x_{moto})$  where  $o$ ,  $x$ , and  $i$  are given in Fig. 3.20.

Integration of the simulated cell required no change to the SLC or the existing OPC communication.

### 3.6.4 The Applied Conceptual Architecture

The simulated supply cell, simulated robot, and simulated cell are computer emulated entities that maintained data essence and procedural effect (Fig. 3.9). In the

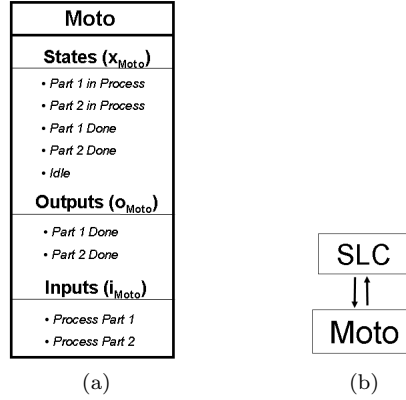


Figure 3.20: A schematic of the states, inputs, and outputs (a) of the Motoman (Moto), and how it connects to the existing hierarchy (b).

Table 3.4: The Essences and Effects of the Simulated Cell (Motoman)

Simulated Cell		
Physical Domain	Spatial Essence	<i>None</i>
	Newtonian Effect	<i>None</i>
Digital Domain	Data Essence	Blender and Java simulation
	Procedural Effects	Outputs and states Listed in Fig. 3.20a

case of the simulated robot and the simulated cell, workpieces entered the simulation through *ghosting*, and after being processed, the workpieces left the simulation through *birthing*. The simulated robot differs from the other two simulations in that it came from the vendor. The simulated robot demonstrates that the modular simulations needed for an HPS can be created without requiring the vendor to open up their proprietary simulation software. The fact that the proprietary simulation software was used shows that the tasks can be done by a third party as well. A third party could then create these modularized simulations from multiple vendors ensuring that their implementations are uniform.

### 3.7 Technical Landscape and Limitations for an HPS

The HPS approach can be applied during both the system wide deployment phase and the component deployment phase. System wide deployment starts with no components in place, then uses pure simulation, and finally ends with a production ready system (Fig.3.21, a-e). The component deployment phase starts with a full manufacturing system, at which point components are added or replaced, also resulting in a production ready system (Fig.3.21, e,d). When the HPS approach is employed in both system wide and component deployment, component simulations are used initially and replaced iteratively by the real components the simulations represent.

**Definition 20. *Pure Simulation:*** *A simulation that includes the components of the manufacturing system in a single simulation environment with no outside connections or external parts.*

**Definition 21. *Component Simulation:*** *A simulation of some specific part of the manufacturing process with the capability of interfacing with real or other simulated components.*

With the HPS approach described in this dissertation, the individual component simulations will use the same interface (communication protocol) as the real components they represent, thus, each simulation is a distinct separate piece of software and hardware. Real components can then be added on a component by component basis (Fig. 3.21d), until the entire process is in its non-simulated actual form (Fig. 3.21e).

Fig. 3.21 represents one progression for deployment. Other progressions for de-



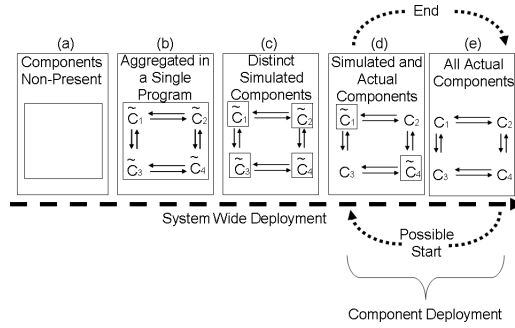


Figure 3.21: For a completely new process, deployment will start with all non-present components and end in all actual components. An inner box around a component  $C$  corresponds to an individual simulation environment. An inner box around multiple  $C$ s corresponds to one simulation environment that includes multiple components.

deployment are also possible as illustrated in Fig. 3.22. Step a is the same but b, c, and e are different. Two limitations of the HPS approach to deployment are that the simulations must be capable of communicating with real components and real components must be capable of interfacing with simulations [32]. The exact order of the introduction of component simulations and real components will at least partially be based on the availability of resources and expertise.

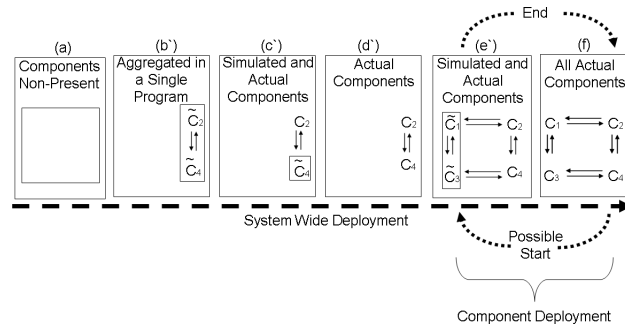


Figure 3.22: An alternative deployment progression to Fig. 3.21.

The HPS accommodates many different simulated components interfaced with many actual components. This type of implementation begs the questions, from where do these simulations originate, what form do they take, and what information do they need. If the HPS approach is adopted in the future the simulations

will be developed and supplied by the vendors of the actual components they represent. The manufacturers are motivated to provide accurate, reasonably priced simulations because the simulation models will be used to support purchasing decisions. A manufacturing process planner could then bring in multiple simulation models from different vendors and compare them in a “try before you buy” scenario. This approach has several benefits to the manufacturing process planner, one being that it allows the process planner to try many different solutions before financially committing to a particular one. Another benefit is that code used to program the simulation can directly be ported to the actual component. For example, Fanuc’s Roboguide simulation software allows the code from the simulation to directly be uploaded to the actual robot.

There are two drawbacks to the “try before you buy” scenario. First, each different vendor component simulation that serves as a possible component replacement or addition may be very different, requiring some learning curve for its implementation. There are both advantages and disadvantages to finding out about implementation difficulties. It will undoubtedly add to the time needed for the entire implementation; however, information about ease of implementation is useful too. Because the overall investment has not been made yet, a process planner can decide on a different solution if it looks like the difficulty of implementation outweighs the gains from production performance. Second, if there is a large variety of components in the process, it may be too time consuming to try multiple different vendor solutions for each component. The HPS approach can still be employed by acquiring the component simulations, but trying multiple vendor options for the same component should be used sparingly in accordance with time and work capacity constraints.

### 3.7.1 Simulated Components

One inevitable implementation barrier is the proprietary nature and form of the simulations from the vendors. The term simulation is used here broadly to include both the software that executes the simulation, and the hardware platform that the simulation sits on. For example, a “soft PLC” (programmable logic controller), which is a viable PLC option in its own right, can be considered as a hard PLC simulation with identical function and connectivity. A process planner could use soft PLCs to test and compare competing solutions. All of the comparisons could be done using the same desktop and interface cards, while the program being run is switched from one solution to another.

This type of simulation is not restricted to PLCs. Section 3.4 discussed how this type of simulation can be done with a robot. The important point is that migration from simulation to final implementation needs no internal alteration. It facilitates system level validation (how the component integrates into the larger system) instead of isolated validation (the component alone).

### 3.7.2 Actual Components and Signal Emulation

Future actual components should be designed with the capability of handling workpieces without spatial essence (signal emulation mode). After a workpiece has gone through ghosting for the first time, all signal emulation capabilities downstream of the ghosting should be active. Vendors could be motivated to give such capabilities because it should make their components more flexible for system development and changes.

The method by which sensor signal emulation is activated can vary. The easiest method is to pass a token/variable with the workpiece that lets the actual components

know that there is a workpiece without spatial essence present.

The behavior of the actual component in signal emulation mode will be up to the vendor. In the example covered in Section 3.6.1 the entire robot program is run in its final state. The part presence signal in the robot program is emulated in the robot when a simulation token is activated, causing the appropriate sensor response at the appropriate time.

The above approach works well when not all sensors are accessible from one location. If all the sensors reside on a single network, signal emulation can be handled by an external signal emulator program that emulates the procedural effects at the appropriate times.

Other ways of dealing with simulated workpieces have been explored. [23] use a conveyor buffer system to allow rough workpieces to appear when a virtual workpiece goes from a simulated component to an actual component, and then disappear when the opposite happens. This solution is effective, but requires that workpiece emulation be handled on a higher system level rather than on a local component level.

For simple part presence sensors, emulation mode may require few code changes. For more advanced sensing such as robot vision, the component may require an emulation mode runtime environment. This runtime environment may require that complex sensing capabilities that cannot be emulated be bypassed completely.

Signal emulation mode for actual components does not allow for the validation of very specific sensor functionality within the system; however, it does allow for the testing of the system as a whole, which is arguably the more challenging endeavor. Validation of an isolated machine (component sensor validation) can be done effectively without the presence of the system, and will likely be as effective whether done

alone or as part of the larger infrastructure.

An HPS allows for large scale system level validation as well as some validation on the local level. Though an HPS as proposed cannot test sensor performance or precise physical functionality, it does allow for validation on the system control level. This capability also gives the process planner the opportunity to integrate components into a system that could be in a mostly simulated phase, reducing the ramp-up or changeover process.

### **3.7.3 Limitations**

An HPS can test system connectivity as well as validate control on different levels of the system's hierarchy. Extending HPS to include robustness and repeatability analysis on a component basis is possible but would require the simulation to have stochastic characteristics or be based on historic statistical data. Internal errors could be introduced in the simulation models by the designers to emulate breakdowns or problems with the actual equipment. How the simulation handles such internal errors would be up to the designer of the simulation model. While simulations with these capabilities could be constructed, this type of information will most likely have to come from data supplied by the vendor of the equipment. Though testing system functionality for process errors such as robot gripper faults and other equipment specific errors are important, the focus of this dissertation is HPS for integration and connectivity.

Errors that arise from the real components or the interaction between components in the HPS can still occur. The error should be observable using the same real time diagnostic software used to monitor the actual system, or by observing that the HPS ceases to run correctly or at all. If the error is not visible for any of the aforementioned reasons, the HPS approach will not be useful in detecting the error. An example of

such an error would be a cutting process that is done incorrectly resulting in a misshapen workpiece. If ghosting happens right after the cutting process, an error may not be detected.

It is acknowledged that simulations which connect and function the same as the processes or components they are meant to simulate do not yet exist from all vendors. It is possible, however, for process planners to create real system interfaces for vendor simulations as demonstrated by the simulated robot from Section 3.6.2

### **3.8 Summary**

The HPS approach focuses on the important aspects of testing for creating a new system or making changes to an existing one. The approach leverages the strengths of HIL while developing a methodology and ontology, creating a formal means of application and understanding. The developed methodology and ontology were applied successfully to the RFT at The University of Michigan, demonstrating how the process could be accomplished using three different scenarios.

## CHAPTER IV

# Hybrid Process Simulation Equivalence Analysis for Deployment

Hybrid Process Simulation (HPS) provides an approach for the implementation of a manufacturing process during its deployment phase. As the HPS progresses towards a completely real system without simulation, a quantitative means of describing the HPS as compared to the final system becomes useful. This chapter presents a quantitative analysis for an HPS as it progresses toward a completely non-simulated system. Structural equivalence, leaf equivalence, and communication equivalence are presented as quantities used to describe where the HPS is in its progression to its non-simulated state.

The developed equivalence analysis is applied to both the Reconfigurable Factory Testbed and the Manufacturing System Emulation Environment. Both examples are HPSs with regions of real and simulated components.

### 4.1 Introduction

The transition from pure simulation to a production ready process using simulated and non-simulated components serves as the keystone of the HPS approach. As the system takes on different combinations of simulated and non-simulated components, understanding how close the HPS is to the final system (how much of the HPS is non-

simulated) as a whole becomes important. Certain questions about the current state of the HPS should be answered from both a control and connectivity perspective, such as: how are these processes simulated (e.g., which essences and effects are included in the simulation), how much of the communication infrastructure is in place, and how do the different components depend on one another. The equivalences and the analysis that goes along with them attempt to answer these questions in a quantifiable manner.

Fidelity and validity are aspects of simulation analysis that began to broach this topic, but they did not address simulation from the perspective of HPS. The equivalence analysis approach presented in this chapter is developed to concisely and quantitatively describe the progression from a completely simulated manufacturing process to a completely actual one.

## 4.2 Equivalence Analysis

Equivalence analysis provides a measure of an HPS's level of simulation, physical presence, and level of dependance. The three equivalence metrics defined here are structural equivalence, leaf equivalence, and communication equivalence. They stand separately from existing simulation metrics. These metrics were chosen because they represent aspects of an HPS that are not present in a pure simulation. With the HPS approach the amount of simulation could transition from 0% (all non-present) to 100% (all simulated) to 0% (all actual) with many intermediate percentages, but this measure is not as simple as adding up the number of simulations and dividing it by the number of components.

For the analysis, focus will be on manufacturing systems with control hierarchies that can be translated into tree structured directed graphs (exactly one root node,



with all nodes in the process having exactly one parent). This structure includes many of the systems in use, leaving the extension of the approach to more general structures for future work.

Fig. 3.21 can provide context for a discussion of what the equivalences are. Development can start with planning and nothing simulated, where all components are non-present. The next step is to develop a pure simulation where all components are simulated in the same program (aggregated). At this step structural equivalence becomes a useful quantitative measurement.

Structural equivalence is a measure of the amount of the HPS that is represented in at least simulated form. After pure simulation, deployment can proceed in two different directions. Simulated components capable of functioning in an HPS can be connected with actual components depending on what is available, or the communication infrastructure can be put in place first with all simulated existent objects. The latter case is called a hardwired simulation because all of the wires for communication are in place. Moving from the pure simulated phase to the hardwired simulation phase means that both structural equivalence and communication equivalence become non-zero.

Communication equivalence is a measure of how much of the communication infrastructure is in place and in what form. If the step after pure simulation is to begin to add actual existent objects and simulated existent objects, then structural, communication, and leaf equivalence become useful quantitative measurements.

Leaf equivalence is an indirect measure of how much of the physically involved (directly or indirectly physically interacts with the workpiece) part of the manufacturing system is present. The physically involved system includes components that physically interact, such as robots, CNCs, and actuators. Even with the hardwired

simulation step employed, deployment will eventually reach a point where some part of the manufacturing system will have actual components making all three equivalences descriptive quantities.

#### 4.2.1 Assumptions

Equivalence analysis as presented here does not consider an individual component's functionality or how accurately an individual component simulation represents its actual counterpart. Instead, the equivalence analysis presented here only considers the essences and effects of the individual components and their respective simulations. In addition, each component is weighted equally when computing the overall equivalence of the system. Future work may want to consider different weightings of the components depending on either the difficulty or the time required for their implementation.

#### 4.2.2 Pre-Analysis

Equivalence analysis is a comparison between the system in its final form and in its current state as an HPS. The quantities involved in calculating the equivalence metrics require a pre-analysis to be performed on the hierarchy of final system. Pre-analysis translates the hierarchy's directed graph into the objects and attributes (entity, body, essence, effect, etc.) introduced in Section 3.3.1. Pre-analysis provides a structural description of the system that is used to map out the system's control structure and connectivity. Fig. 4.1a shows the control hierarchy and connectivity of the Reconfigurable Factory Testbed (RFT) at the University of Michigan [54] in the form of a directed graph. Fig. 4.1b shows the pre-analysis of the final system.

**Definition 22. System Graph:** *A directed graph of a manufacturing system where the edges of the graph represent control communication from the parent node (the*

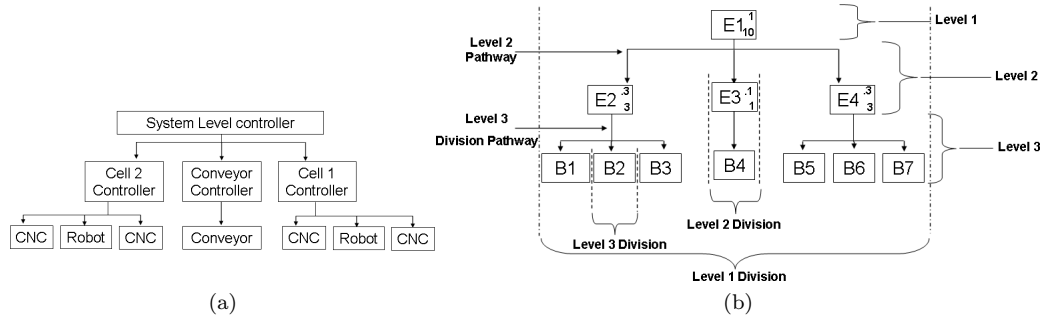


Figure 4.1: Pre-analysis divides a hierarchical control and communication infrastructure (a) into levels and divisions, where E represents an entity, and B represents a body (b).

*controller) to the child node (the controlled). Nodes represent entities/bodies and the edges of the graph represent the pathways.*

Fig. 4.1b shows Entities (E) and Bodies (B) that correspond to the blocks in Fig. 4.1a. The number after E or B represents an instance of that class. E1 corresponds to the System Level Controller (SLC) which is an instance of the *Entity* subclass, E2 corresponds to the Cell 2 Controller (C2C) which is also an instance of the *Entity* subclass, B1 corresponds to the CNC under the Cell 2 Controller which is an instance of the *Body* subclass, and so forth. The numbering convention is chosen for convenience, but E1 could also be referred to as the SLC *Entity*. From this point forward instances of the *Entity* or *Body* subclass will be referred to as entity and body in non-italicized form.

**Definition 23. Level:** *A subset of the graph containing nodes and edges in an HPS control hierarchy denoted by the distance of the nodes from the root node. Levels are numbered starting from the root node (node with no parent). The root node resides in level 1, and children of the root node reside in level 2, children of level 2 nodes reside in level 3 and so on. Edges/pathways between level  $i$  and  $i + 1$  belong to level  $i + 1$ ; therefore, the pathway between E1 and E2 is in level 2.*

Level 1 in the RFT example is the SLC (E1). Level 2 in the RFT contains the cell 2 controller (E2), the conveyor controller (E3), the cell 1 controller (E4), as well as the edges between E1 and the level 2 entities. All of the bodies in the RFT example are in level 3.

**Definition 24. *Division:*** *A subtree of the graph that corresponds to all the children, both indirect and direct, of a particular node.*

**Definition 25. *Terminal Node:*** *A node (leaf) with no children.*

Physical equipment will often times be a terminal node because the equipment itself does not usually supply control for another part of the manufacturing process.

For an HPS with  $K$  total levels let  $I = \{\text{all nodes}\}$ ,  $DC_i = \{\text{all direct children of node } i\}$ ,  $IC_i = \{\text{all indirect children of node } i\}$ , and  $|I|$  represents the total number of nodes. An individual node can be referenced by noting that a node  $i$  is on level  $k$  of an HPS. Node  $i$  has a number of direct children  $|DC_i|$  and indirect children  $|IC_i|$ .

As shown in Fig. 4.1b, the level 1 division has all pathways and bodies within it. There are three level 2 divisions because there are three level 2 entities. The level 2 divisions can be differentiated by using the name of the corresponding entity/object; therefore, the E2 division includes E2, B1, B2, B3, and the pathways that connect them.

Another part of pre-analysis involves determining the importance of the entities/bodies as well as their influence. Importance is determined by an entity's/body's degree  $D$ .

**Definition 26. Degree:** *The total number of direct children for node  $i$  ( $|DC_i|$ ) plus the number of indirect children of node  $i$  ( $|IC_i|$ ).*

$$D_i = |IC_i| + |DC_i| \quad (4.1)$$

$D_i$  can also be calculated based on the degree of its direct children.

$$D_i = \begin{cases} \sum_{j \in DC_i} (D_j + 1) & \text{if } |DC_i| > 0 \\ 0 & \text{if } |DC_i| = 0 \end{cases}$$

In Fig. 4.1b the degree of each entity/node is shown in the lower right corner of its block. Terminal nodes have zero degree and so are left blank. The entity E2 in Fig. 4.1b (in level 2) has three direct children and no indirect children, therefore, it has a degree of 3.

The other quantitative measure associated with entities/bodies is the influence; the influence is noted in the top right corner of an entity's or body's box.

**Definition 27. Influence:** *The effect an entity/body has on the system as a whole.*

*The influence  $IFL_i$  of node  $i$  is calculated by*

$$IFL_i = \frac{D_i}{D_{root}} \quad (4.2)$$

where  $D_{root}$  is the degree of the root node which is equal to  $|I| - 1$  (the SLC/E1 in the RFT example) and the  $IFL_{root} = 1$ .

### 4.2.3 Structural Equivalence Analysis

Structural Equivalence (STE) is defined as a measure of how much of the system is represented in some form, either simulated or non-simulated. With structural equivalence, a process planner can better understand how much of the system is accounted for.

**Definition 28. *Structural Equivalence:*** *The measure of how many of the entities and bodies in the HPS are represented in simulated or actual form.*

$$STE = \frac{1}{K} \sum_{k=1}^K \frac{N_k}{N_k^F} \quad (4.3)$$

where  $N_k$  is the number of simulated or actual nodes at level  $k$  of the HPS, and  $N_k^F$ , is the number of nodes in the non-simulated manufacturing system at level  $k$  for levels 1 to  $K$ . Equation 4.3 can also be written as:

$$STE = \frac{N}{|I|} \quad (4.4)$$

where  $N$  is the current number of nodes. STE is a measure between 0 and 1, where an STE near 1 indicates that much of the system is actual or simulated. The closer the STE is to 1 the more useful any testing and validation is to the final non-simulated system. An STE of 1 means that the system has all entities/bodies that will exist in the final system represented in the current HPS. If the STE is low, a process planner can use this understanding to allocate more resources to increasing the STE, rather than to validation, in the beginning of process deployment.

The STE can provide an overall look at the current state of deployment for the HPS; however, a more detailed look at the factors that contribute to structural equiv-

Table 4.1: A sample comparison table created after structural equivalence analysis

$k$	$N_k^F$	$N_k$
1	1	0
2	3	1
3	7	5

alence may be more useful. A comparison table can be used to give a more in depth view of the current HPS and its relation to the completely actual process. The columns represent the level  $k$ , the number of nodes in the non-simulated system at level  $k$  ( $N_k^F$ ), and the number of nodes in the current HPS at level  $k$  ( $N_k$ ). Table 4.1 is a comparison table for the RFT with an example HPS in the third column. The comparison table presents a bird's eye view of the HPS from a simulation and deployment perspective. A process planner can then have more information to make good decisions about how to proceed with implementation and troubleshooting. The process planner may have information about what equipment has arrived and even what equipment is installed; however, a comparison table is a mix of the information about the installed equipment, the implemented simulations, and where simulated and actual components are in the control hierarchy.

#### 4.2.4 Leaf Equivalence Analysis

Leaf Equivalence (LE) describes what is physically deployed at the moment based on terminal bodies and entities. Terminal bodies and entities (Def. 25) are used because machines with a physical footprint in a manufacturing process will in most cases be located on the bottom of the control hierarchy; therefore, an analysis on terminal entities and bodies is closely correlated to the physical presence of the system. LE differs from structural equivalence because a completely simulated system could have an STE of 1, but a much lower LE of 0.5.

Fig. 4.1 illustrates an example where level 3 contains the terminal nodes of the

HPS, and thus all bodies on level 3 are terminal bodies. Once the terminal bodies and entities are identified, a Comparison Number for Leaf Equivalence (CNL) can be calculated for all simulated components. The comparison number comes from comparing the essences of the simulated body or entity in the HPS to its corresponding non-simulated form. Effects are not included because their comparisons are captured in the analysis of the pathways between entities and bodies in communication equivalence (see Def. 30). Equation 4.5 shows how the comparison number is calculated.

**Definition 29. Leaf Equivalence:** *The measure of how many of the essences of the terminal entities and bodies (nodes) in the HPS are represented in simulated and actual form.*

$$CNL_i = \begin{cases} 1 & \text{if Presence} = \text{actual}, \\ \frac{ESS_i}{ESA_i} & \text{if Presence} = \text{simulated}, \\ -1 & \text{if Presence} = \text{non-present}. \end{cases} \quad (4.5)$$

Let  $TN = \{\text{nodes with } D_i = 0\}$ .

$$LE = \frac{1}{|TN|} \sum_{j \in TN} CNL_j \quad (4.6)$$

where for node  $i$   $ESA_i$  and  $ESS_i$  are the number of essences in the actual entity/body, and the number of essences in the simulated entity/body. In this dissertation, essences and effects within a domain are treated as all present or all absent; thus,  $ESA$  is almost always 2.  $ESA$  could be 1 for cases when a component only has a digital existence, such as a controller, or only has spatial essence such as a mechanical component that performs a physical task and has no need for data essence.  $CNL$  will be 0.5 in the vast majority of simulated components; however,  $CNL$  can



equal 1 if both the physical and digital domain essences are simulated.

The LE is always between -1 and 1, and an LE of less than 0 usually means that the majority of the components are non-present. An LE of 0 means that there are an equal number of non-present and simulated or actual essences. An LE greater than 0 means that the majority of the essences are represented in some way.

#### 4.2.5 Communication Equivalence Analysis

Communication Equivalence (CE) focuses analysis on the pathways of the HPS. Fig. 4.1 shows different pathways at different levels. Pathways are signified by the child node of a parent-child pair; therefore, in Fig. 4.1 there is a B1 pathway, a B2 pathway, an E2 pathway and so on. With the exception of the root node, every connected node has a pathway. The system illustrated in Fig. 4.1 has ten ( $=|I| - 1$ ) pathways. The number of pathways is also equal to the degree of the root node ( $D_{root}$ ). Multiple pathways can be implemented on a single network, such as DeviceNet or Profibus.

Calculation of CE is similar to LE, in that CE requires that each pathway in the HPS be compared to the pathway in the non-simulated manufacturing system to attain a Comparison Number for Pathway Equivalence (CNP). Unlike CNL, however, CNP includes both essence and effect to capture both the physical structure (wires and connections) as well as the information used in communication.

Let  $I_{-root} = \{\text{all nodes except the root node}\}$

**Definition 30. *Communication Equivalence:*** *The measure of how much of the communication infrastructure is represented and whether it is physically present.*

$$CNP_i = \begin{cases} 1 & \text{if } Presence = \textit{actual}, \\ 0.5 \left( \frac{ESS_i}{ESA_i} + \frac{EFS_i}{EFA_i} \right) & \text{if } Presence = \textit{simulated}, \\ -1 & \text{if } Presence = \textit{non-present}. \end{cases} \quad (4.7)$$

$$CE = \frac{1}{|I| - 1} \sum_{j \in I - \textit{root}} (CNP_j) \quad (4.8)$$

where  $CNP_i$ , is the comparison number for the pathway to node  $i$  from its parent,  $EFA_i$ , and  $EFS_i$  are the number of effects in the actual entity/body for node  $i$ , and the number of effects in the simulated entity/body for node  $i$  respectively. The denominator is  $|I| - 1$  because the root node ( $i = 1$ ) has no parent. The CE is between -1 and 1, and corresponds to the same conclusions as LE but only pertains to the communication infrastructure.

#### 4.2.6 Equivalence Analysis Discussion

Pre-analysis and the equivalences not only provide information about the HPS in reference to how close it is to the non-simulated system, but also provide information about the system as a whole. Pre-analysis done on the non-simulated manufacturing system can help determine how the setup should be brought online, and where resources should be allocated. For example, if pre-analysis identifies an entity with both a high degree and influence, the process planner can allocate resources early to this entity and its corresponding division. Equivalence analysis can also help outline an overall plan for deployment. Communication equivalence will complement both structural equivalence information as well as leaf equivalence information by quantitatively giving the status of the connectivity of the system as a whole.

During deployment the equivalences can be used to quantitatively track progress. The tracking can be recorded across different deployments and used to optimize a

strategy for deployment. An example of this comparison would be the deployments outlined in Fig. 3.21 and 3.22. Even though the equivalence metrics are the same for a and f, they are different for b through e. If these cases represented two similar systems then some conclusions could be made depending on which deployment was faster and cheaper. For large OEMs, equivalence analysis can also be used to compare multiple different deployment sites, aiding in the company wide strategic allocation of resources. This enables quick educated decisions to be made from a non-local perspective based on an objective analysis.

It is important to note that the information gained from equivalence analysis does not provide the complete picture for deployment status. There is an almost infinite number of factors that could also affect either a process planner's or a large OEM's decisions about deployment. The equivalence analysis does, however, provide a simple objective view of progress during deployment using the HPS approach.

### 4.3 RFT HPS Equivalence Application

The RFT test scenario represents a process that is initially a completely *actual* manufacturing system in need of modification due to some malfunctioning or out-dated component. In this scenario, a *simulated* cell/component is integrated into the existing process causing it to become an HPS. This scenario is a snapshot of the component deployment process described in Section 3.4. Fig. 4.2 contains two graphs of what the equivalences would look like during this deployment.

The supply cell addition from Section 3.6.1 is a scenario that includes the addition of a simulated overhead gantry (Fig. 3.16). The overhead gantry works as a supply cell to the rest of the RFT (Fig. 3.17). Testing the system with the simulated supply cell did not require the use of both cell 1 and cell 2, therefore, only cell 1 was used

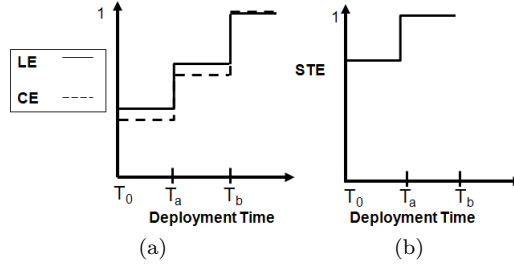


Figure 4.2: Graphs of the equivalences for the RFT component deployment example where  $T_a$  represents the time when the simulated supply cell is introduced, prior to which the supply cell is non-present.  $T_b$  marks the time when the actual component is installed.

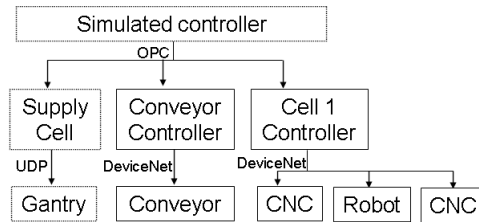


Figure 4.3: A directed graph of the version of the RFT used in the HPS (3.17)

(Fig. 4.3).

#### 4.3.1 RFT Equivalence Analysis

The first step in pre-analysis is to translate Fig. 4.3 into the same form as Fig. 4.1b, where the levels and divisions are specified, and the degrees and influences are calculated. Fig. 4.4 shows the results of the preliminary analysis. Table 4.2 shows the resulting comparison table.

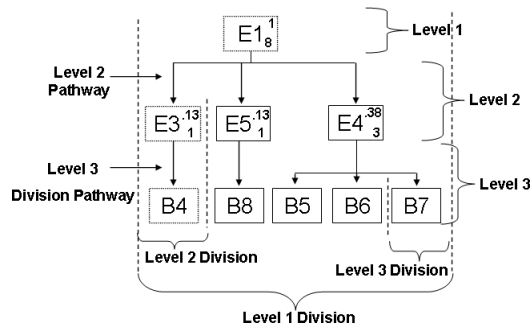


Figure 4.4: The preliminary analysis of the HPS used in testing.

Table 4.2: Comparison table of the RFT HPS

$k$	$N_k^F$	$N_k$
1	1	1
2	4	4
3	8	8

The structural equivalence can be calculated from Equation 4.4, where  $N = 9$ , and  $|I| = 9$ . The values for the divisions are listed in Table 4.2.

The STE changes during the deployment process. When the plan for adding the cell is first begun the supply cell is completely non-present making the STE 0.78. After the simulation of the supply cell is installed the STE then becomes 1. Time  $T_a$  on Fig. 4.2 corresponds to the time when the simulation of the supply cell is installed.

Leaf equivalence is determined by first calculating the comparison number for all terminal/leaf nodes (CNL). All of the bodies are terminal nodes in the RFT, and Table 4.3 lists all of the terminal bodies along with their CNL. If the component does not have any of the essences or effects it is listed as *not simulated*.

The leaf equivalence is 0.9, which indicates that the majority of the process is in place with both simulated and actual components.

The CNP for pathways are calculated in Table 4.4. The communication equivalence as defined in Equation 4.8 is determined to be 0.88. Pathways from the System Level controller (E1) to the supply cell controller (E3), and from the supply cell controller to the gantry (B4), are simulated.

#### 4.3.2 RFT Equivalence Analysis Summary

A structural equivalence of 1 means that no component in the system has a non-present attribute. If this information is coupled with the leaf equivalence of 0.9, it can be concluded that there are both simulated and actual components in the HPS.

The communication equivalence, which is 0.88, means that the communication

Table 4.3: The RFT's HPS terminal bodies and their calculated CNL

<b>Supply Cell Gantry (CNL<sub>B4</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>not simulated</i> <i>not simulated</i>
Digital Domain	Data Essence Procedural Effect	Gantry simulation program Workpiece data generation
<b>Conveyor (CNL<sub>B8</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	The physical conveyor Workpiece transport
Digital Domain	Data Essence Procedural Effect	Conveyor control program Workpiece location data change
<b>Cell 1 CNCs (CNL<sub>B5</sub> and CNL<sub>B7</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	The physical CNC Cutting of the workpiece
Digital Domain	Data Essence Procedural Effect	CNC control program Workpiece process status change
<b>Cell 1 Robot (CNL<sub>B6</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	The physical robot Loading and unloading of the workpiece into the CNCs
Digital Domain	Data Essence Procedural Effect	Robot control program Workpiece location data changed

Table 4.4: The pathways within the HPS and their comparison numbers

<b>UDP Pathway (CNP<sub>E3</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Part request commands Transported commands
<b>OPC Pathways (CNP<sub>E5</sub> and CNP<sub>E4</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	Ethernet cable Ethernet OPC signal
Digital Domain	Data Essence Procedural Effect	Conveyor and cell control commands transported commands
<b>UDP Pathway (CNP<sub>B4</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Part transport request Transported commands
<b>DeviceNet Pathway (CNP<sub>B8</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	DeviceNet wiring DeviceNet Signals
Digital Domain	Data Essence Procedural Effect	Stop control commands Transported commands
<b>DeviceNet Pathway (CNP<sub>B5</sub>, CNP<sub>B6</sub> and CNP<sub>B7</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	DeviceNet wiring DeviceNet Signals
Digital Domain	Data Essence Procedural Effect	Cell control commands Transported commands

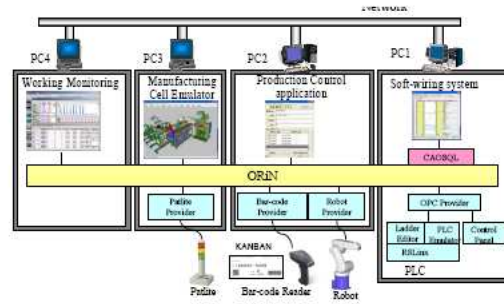


Figure 4.5: System Structure of the case study from [36].

infrastructure is generally at the same point of installation as the other components. A manufacturing process planner can deduce from the equivalences that there is very little installation remaining, and a good deal of the actual components are in place and connected. The process planner can now shift some resources from installation (if they haven't already) and into validation. A structural equivalence of 1 and a communication equivalence greater than 0.5 means that system wide validation is possible.

#### 4.4 Manufacturing System Emulation Environment Equivalence Application

In this chapter, the MSEE example from [36] will represent a snapshot of a system wide deployment process. Fig. 4.5 shows the layout of the MSEE system. Fig. 4.6 illustrates how the complete deployment process might progress. The installation procedure in this case dictates that the CE lags behind the LE for the entire deployment. The graph also illustrates how the STE becomes 1 (at time  $T_a$ ) when all components are simulated or actual. After time  $T_a$ , CE and LE progress toward 1 until the process has all actual components and connections.

Simulations in an HPS are meant to connect to the system in the same way as the components they represent, where emulators in an MSEE communicate through

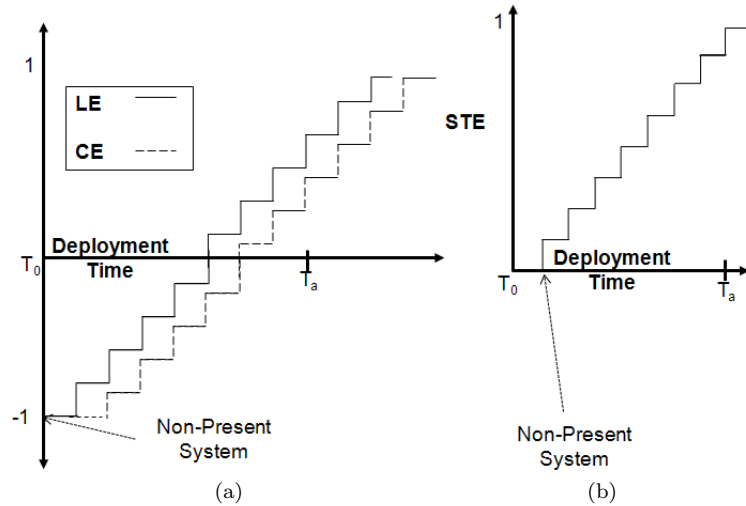


Figure 4.6: Graphs of the equivalences for the MSEE component deployment example, where  $T_a$  signifies the point in time represented by the example. LE and CE are on the same graph to show how CE can lag behind LE.

a software middle layer called ORiN (Open Resource interface for the Network / Open Robot interface for the Network). All communication takes place through ORiN, even between actual components. Their case study setup consists of a robot, a tester, a palettizer, and a conveyor as emulators, and a robot controller, a bar code reader, and a patlite (warning sign indicator) as actual equipment. There are also four systems present which include a production control system, a manufacturing cell emulator, a soft-wiring system, and a working monitoring system.

The soft-wiring system runs on ORiN and takes the place of the hard-wiring that would normally exist between controllers and equipment. The soft-wiring system enables communication between real and emulated components. The manufacturing cell emulator enables all of the actual and emulated components to function as if there is material flow. Because parts of the system are emulated, there can be no physical material flow; therefore, the synchronization of all processes requires material flow emulation.

The following is quoted directly from [36] to describe the function of the MSEE:



1. The barcode reader receives production order information by a Kanban.
2. The production control system receives the production order information via ORiN.
3. The production control system indicates the need to investigate the quantity of a product along with the production order information toward the PLC via the soft-wiring system.
4. The assigned product is picked up and moved to the conveyor by the palettizer in the emulator. These behaviors are controlled by the PLC via the soft-wiring system.
5. The assigned product is translated to a position in front of the robot by the conveyor in the emulator. This behavior is controlled by the PLC via the soft-wiring system.
6. The assigned product is picked up and moved to the tester by the robot. These behaviors are controlled by the PLC and the robot controller via ORiN.
7. The assigned product is investigated by the tester in the emulator.
8. Results of this investigation in the emulator are sent to the PLC via the soft-wiring system. Then the control panel displays the results via the PLC. The patlite shows warning signs along the results via the PLC.
9. The assigned product is picked up and moved from the tester to the conveyor by the robot. These behaviors are controlled by the PLC and the robot controller via ORiN.
10. The assigned product is translated to a position to the end of conveyor by the conveyor in the emulator. This behavior is controlled by the PLC via the soft-wiring system.

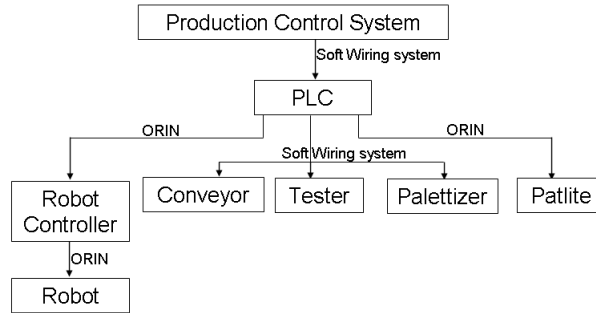


Figure 4.7: The directed graph of the MSEE system.

Table 4.5: Comparison table of the MSEE

$k$	$N_k^F$	$N_k$
1	1	1
2	1	1
3	5	5
4	1	1

11. The assigned product after evaluation is eliminated in the emulator.

For explanation purposes assume both the patlite and the robot controller receive commands from the PLC via ORiN, and the conveyor, tester, and palettizer receive commands from the PLC via the soft-wiring system on ORiN. Fig. 4.7 shows the directed graph of the system. The bar code reader is left out of the hierarchy because it only provides communication between Kanban and the Production Control System (PCS).

#### 4.4.1 MSEE Equivalence Analysis

The first step in preliminary analysis is to translate Fig. 4.7 into the same form as Fig. 4.1b, where the levels and divisions are specified, and the degrees and influences are calculated. Fig. 4.8 shows the results of the preliminary analysis, and Table 4.5 shows the resulting comparison table.

Structural equivalence can be calculated using Equation (4.4), where  $K = 4$  is the number of levels and the values for the divisions are listed in Table 4.5. The

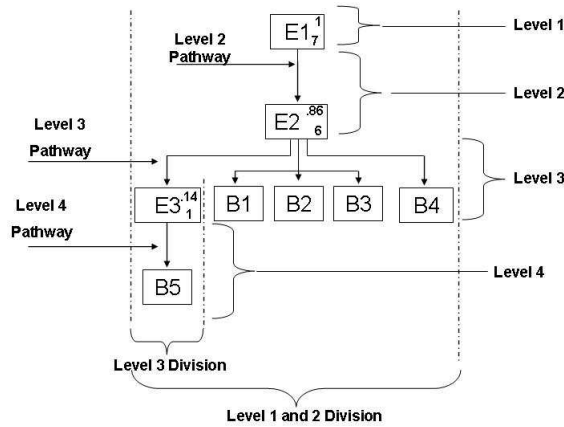


Figure 4.8: The preliminary analysis of the MSEE

structural equivalence is 1 because all of the bodies and entities in the non-simulated system are also present in the current setup.

Leaf equivalence is determined by first calculating the CNL for all terminal bodies and entities. In this example, all of the bodies are terminal nodes in the MSEE. Table 4.6 lists all of the terminal bodies along with their comparison numbers (Equation 4.7). If the emulated component of the body does not have an essence or effect that the actual component has, it is listed as *not simulated*. If neither the emulated nor the actual component have essence or effect, it is listed as *none*.

The patlite and the robot both have  $CNL = 1$  because they are in their actual form. The other three components are all emulations and so only exist in the digital domain, where their actual counterparts would exist in both domains. Using Equation 4.6, the leaf equivalence is determined to be 0.7.

The CNP for pathway equivalence is shown in Table 4.7. The communication equivalence as defined in Equation 4.8 is determined to be 0.5. Both ORiN and the soft wiring system transport the same commands as the hardwired system would; however, the actual hard wiring is not present nor is the same electrical signal that relays the message. The CE does capture that the hardwired infrastructure is not

Table 4.6: The terminal bodies and their calculated comparison numbers

<b>Conveyor (CNL<sub>B1</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Production control program Emulated product transportation
<b>Tester (CNL<sub>B2</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Tester emulation in the cell emulator Product investigation results
<b>Palettizer (CNL<sub>B3</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>non-present</i> <i>not simulated</i>
Digital Domain	Data Essence Procedural Effect	Tester emulation in the cell emulator Emulated product placement on the emulated conveyor
<b>Patlite (CNL<sub>B4</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	The physical patlite Warning light signals
Digital Domain	Data Essence Procedural Effect	<i>None</i> <i>None</i>
<b>Robot (CNL<sub>B5</sub> = 1)</b>		
Physical Domain	Spatial Essence Newtonian Effect	The physical robot Movement of the product
Digital Domain	Data Essence Procedural Effect	<i>None</i> <i>None</i>

Table 4.7: The pathways within the MSEE and their comparison numbers

<b>Soft wiring system Pathway (CNP<sub>E2</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	order commands and inventory inquiries Transported commands
<b>ORiN Pathway (CNP<sub>E3</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Robot task commands Transported commands
<b>ORiN Pathway (CNP<sub>B5</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Robot motion commands Transported commands
<b>Soft wiring system Pathway (CNP<sub>B1</sub>, CNP<sub>B2</sub>, and CNP<sub>B3</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Cell control commands Transported commands
<b>ORiN Pathway (CNP<sub>B4</sub> = 0.5)</b>		
Physical Domain	Spatial Essence Newtonian Effect	<i>Not simulated</i> <i>Not simulated</i>
Digital Domain	Data Essence Procedural Effect	Light commands Transported commands

present but it does not capture that the soft wiring system is perhaps a better simulation of the communication than the ORiN system alone.

#### 4.4.2 MSEE Equivalence Analysis Summary

From the perspective of the process planner, a structural equivalence of 1 means that the system is completely accounted for in some way, be it actual or emulated. The leaf equivalence of 0.7 means that, of the components that make up the physical presence of the system, 70% of their essences are in place. If the leaf equivalence is compared to the structural equivalence, it then becomes apparent that the system contains simulated components because there are no non-present entities (STE = 1). The communication equivalence means that 50% of the communication infrastructure's essences and effects are in place. A communication equivalence of 50% by

itself means that all of the pathways could be represented in some way. If the STE is then factored in, it reasonable to say that the entire system has communication capability, and system wide validation is possible. Additionally, some of the physical equipment that makes up the system has begun to arrive, is installed, and of the equipment that is not physically installed the rest is simulated. The communication equivalence also says issues that arise from component interfacing may arise later because the communication infrastructure has a large amount of simulation.

#### **4.5 Summary**

In this chapter a equivalence analysis methodology for a manufacturing system in its deployment phase using the HPS approach has been put forth. The analysis results in a quantitative description of an HPS's current composition as compared to its final non-simulated form. A process planner can use the results of an equivalence analysis for forming a concise understanding of an HPS's deployment progress.

Because the HPS approach to deployment requires a different perspective than the ones popularly employed for pure simulation, equivalence analysis is developed to create a framework for quantitatively describing form. The process planner can use this information about form to make decisions about resource allocation during process deployment.

## CHAPTER V

### Autonomous Mobile Entities in Hybrid Process Simulation

This chapter expands the existing ontology and conceptual architecture of Hybrid Process Simulation (HPS) to include Autonomous Mobile Entities (AMEs). The HPS approach as previously presented was applied to components that did not change their overall location. Components of a manufacturing system that change their overall location such as AGVs and humans were not addressed. Additionally, simulation environments that may contain AMEs (possibly simulating physical domain essences and effects) were also not addressed.

This chapter defines AMEs and a new digital domain, the digital AME domain. A digital AME domain is then implemented on the Reconfigurable Factory Testbed (RFT) using a game engine environment called Panda3D. The implementation on the RFT accomplishes the following:

- Mapping from the digital logic domain to the digital AME domain
- Mapping from the digital AME domain to the digital logic domain
- Creation, tracking, and control of an AME

## 5.1 Introduction

The HPS approach of Chapter 3 proposed a means for integrating simulated and actual existents (machines and controllers) that were assumed to be stationary, had a single parent node, and had a physical connection to the manufacturing system infrastructure. Examples of such Physically Connected Entities (PCE) include robots, CNCs, and gantries. This chapter will relax these assumptions, and thus serves as an example of how the HPS approach can be extended in future work. AMEs can switch their parent node as well as interact with sister nodes to negotiate tasks.

Though AMEs do account for a large fraction of the manufacturing process, only including PCEs would neglect entities that are not physically connected (autonomous mobile) but potentially very important to the process. Humans and AGVs are two types of Autonomous Mobile Entities (AMEs) that are commonly present in a manufacturing process. This chapter first gives a background of what already exists in the literature for AME simulation. It then expands the existing conceptual architecture to include AMEs, and presents the implementation methodology and equivalence analysis for AMEs. Lastly, an example case is put forth that demonstrates the implementation of an AME.

## 5.2 Background

AME simulation within manufacturing consists of two major topics: (1) mobile robot simulation and (2) human simulation in manufacturing. Mobile robot simulation addresses AMEs such as AGVs, while human simulation in manufacturing can address manual tasks and human controlled operations.



### 5.2.1 Simulation of Mobile Robotics

Within the scope of simulation, one of the closest fields of applicable knowledge for AME incorporation is that of mobile robotics. Mobile robots can be autonomous, and exist and interact in a three dimensional space. In order to incorporate AMEs in a manufacturing HPS one should first look at how mobile robots are addressed in simulation. There are various platforms that address mobile robot simulation such as Unified Systems for Automation and Robot Simulation (USARSim) [6, 11, 63], Mobility Open Architecture Simulation and Tools Framework (MOAST) [12], Player/Stage [26], and Microsoft's Robotics Studio [10, 37]. Most of these solutions entail an environment that allows testing of robot controllers and functionality within a game engine type environment. These platforms allow the robots to act autonomously but do not typically include native connections to outside physical systems which are required for the HPS approach.

### 5.2.2 Human Simulation in Manufacturing

A discussion of the literature for the incorporation of AMEs in HPS would not be complete without considering how humans are included in manufacturing simulation. In the past manufacturing simulations implemented a very simplified version of humans. They were often times modeled like other machines in the process, they only appeared human. New research has begun to address humans in manufacturing simulation from two vantage points: one from the behavioral perspective and the other from the integrative perspective. The behavioral perspective works to try and incorporate how humans make decisions [67] as well as incorporating human behavioral models and social networking models [20]. The integrative approach on the other hand tries to incorporate human models in existing manufacturing simulations

tools like Automod and Arena [9, 22, 44]. These solutions however are all-in-one and do not easily support the outside communication needed for HPS.

### 5.3 Autonomous Mobile Entities

In this chapter the components of the system (other than workpieces) that are not physically connected to the infrastructure are addressed. Non-connected entities that are capable of moving about are termed here as AMEs.

**Definition 31. *Autonomous Mobile Entity (AME)*:** Any entity that performs a physical task (causes Newtonian effect) that is also capable of moving about its environment while not physically being connected to any part of it.

We will revisit the example from Chapter 3 (Fig. 3.4), only now consider that there are four rooms each with wireless controllers that communicate with the fan (Fig. 5.1). Each controller can request that the fan come to its respective room and begin cooling the interior of the room. The fan in this example is an AME that must now make routing and navigation decisions based on its current location and its destination.

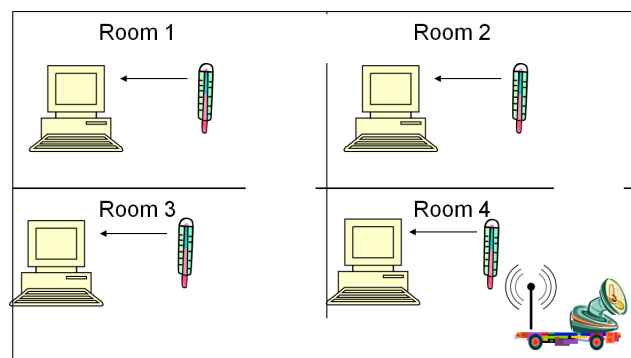


Figure 5.1: A four room environment with four controllers that can all request cooling from the fan AME.

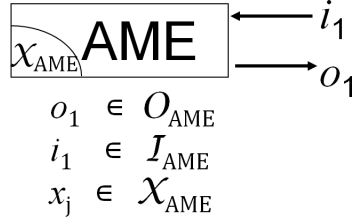


Figure 5.2: A component AME of a test setup may have one interfaces with inputs and outputs.

As the definition implies, AMEs are in fact entities, and thus Def. 15 applies in the same way as to PCEs. Similarly, AMEs are also components of the process with behaviors and interfaces. Fig. 5.2 shows an AME with one interface. Though the interface is singular, an AME may receive inputs from multiple components at different times. The fan AME can receive requests for cooling from all of the controllers through its interface. Depending on the programming of the AME, these requests can be prioritized and then sorted to calculate the route best suited to accomplish its tasks. The behavior of an AME in a test setup is a function  $o_{AME} = f_{AME}(i_{AME}, x_{AME})$ , where  $x_{AME} \in \mathcal{X}_{AME}$  are the states of the AME,  $o_{AME} \in \mathcal{O}_{AME}$  are the outputs, and  $i_{AME} \in \mathcal{I}_{AME}$  are the inputs.

Simulated AMEs in an HPS present a challenge because of their differences from simulated and actual PCEs. Some of the key differences regard spatial essence and the qualities by which a PCE's or AME's simulation environment is governed. Table 5.1 lists these key differences. PCEs never change their location, and thus, their position does not need to be tracked. If the simulated PCE is a simple signal emulator than its simulated location is undefined, because it is not relevant to the signal emulation.

AME position, on the other hand, is an important part of its function. If malfunctions occur with respect to guidance and/or navigation, damage to equipment or injury may occur; therefore, it is important that the simulation environment for

Table 5.1: Comparison Between Different Types of Components

	Actual PCE	Simul. PCE	Actual AME AGV	Simul. AME AGV	Actual AME human	Simul. AME human
<b>Spatial Essence (location) and PCP</b>	constant location, constant PCP	location not defined, constant PCP	location can change, no PCP	location not defined, no PCP	location can change, PCP can change	location not defined, PCP can change
<b>Environment Governed Qualities</b>	Newton's laws & process control	process control	Newton's laws and process control	Newton's laws and process control	Newton's laws	Newton's laws

an AME simulate the current location. The simulated fan AME from Fig. 5.1 needs to address path planning and simulate its changing location to avoid collisions with obstacles like the wall. If the fan AME were a simple signal emulator, motion would not be simulated, it would just communicate to the controller that it has arrived and whether or not it is currently blowing.

The need for the AME simulation environment to include location also requires that the simulation environments be governed by different qualities. Both the actual AME and the actual PCE are subject to Newton's laws, however, a signal emulating PCE simulation need not simulate Newton's laws. An AME simulation's capability to simulate collisions and other physics dependant qualities is important. Without this capability the usefulness of the simulation is diminished.

AMEs also do not have a physical connection point (PCP), and therefore, swapping a simulated AME out with an actual AME becomes a challenge. In the fan AME example, how does the simulated AME communicate with the infrastructure?

Despite the differences between AMEs and PCEs, an HPS approach that includes simulated AMEs must:

- Fit within the existing HPS framework and equivalence analysis methodologies

presented in preceding chapters.

- Maintain its distributed structure such that other components can remain fully functional regardless of whether they themselves are actual or simulated.
- Include simulated components that are interchangeable with their actual counterparts.

#### **5.4 AME Domain**

In order that simulated AMEs may fit within the existing HPS framework, the nature of their domain, essences, and effects must be determined. Simulated AMEs exist within a computer environment; therefore, their essence and effect attributes are a part of the digital domain. A simulated AME's essence attributes, however, differ greatly from those of the manufacturing control infrastructure, despite the fact that both reside in computers.

The digital domain of the control infrastructure requires essences that enable workpiece tracking and routing, such as tracking number, current processing station, and general OPC data. The control infrastructure of the manufacturing process is governed by control algorithms, and the essence and effect attributes of the control infrastructure (Section 3.5) are completely composed of information within the PLCs and machine controllers of the manufacturing process.

The simulated AME environment requires a simulated three dimensional space with digital essence attributes such as mass and velocity. The simulated AME's environment is governed by simulated laws of physics, and the essence and effect attributes are in a physics simulated environment completely outside of the PLCs and machines of the manufacturing process; furthermore, the typical PLCs of a manufacturing process are not capable of producing procedural effects from principles

based on the laws of physics, and thus a different platform is required. One possible approach to incorporating simulated AMEs in an HPS while also addressing the differences in environments is to introduce a new type of digital domain.

The differences in essence and effect attributes between the control infrastructure of the manufacturing process and the simulated AME environment require a differentiation to be made between two types of digital domains. The type of digital domain which encompasses the control infrastructure of the manufacturing process will be termed the Digital Logic Domain, and the essence and effect attributes therein will be termed Logic Data Essence Attributes and the Logic Procedural Effect Attributes. The AME version of the terms will then be the Digital AME Domain, AME Data Essence Attributes, and AME Procedural Effect Attributes.

## 5.5 AME Implementation Methodology

The HPS implementation methodology for an AME must retain the system's distributed structure and simulation/actual component interchangeability. Section 3.4 outlined steps for the implementation of entities and bodies in two scenarios (from pure simulation and from an existing system). These steps implied the use of PCEs; however, the incorporation of simulated AMEs into an HPS follows the same steps.

Though the basic steps are unchanged some key characteristics specific to AME implementation should be explained. The outlined region described in step 3 is the digital AME domain. One difference between an AME and a PCE is that an AME may have a single interface to different components at different times.

Consider an example where an AME is to be added to an existing process. The final setup after implementation will look like Fig. 5.3 where the illustrated control hierarchy shows that E1 controls E2, E3, and E4, E2 controls E5 and E7, and so

forth. Assume each of the terminal nodes has a fixed location on the manufacturing floor. Now consider that the AME is meant to travel from its current location to one of two pickup locations (dashed squares). Upon the AME's arrival to the location between E5 and E7, controlling entity E2 will regard the AME as one of the entities under its control, and upon arrival to the location between E9 and E11, controlling entity E4 will do the same. Both E2 and E4 are alerted to the AME's arrival by corresponding light curtain sensors. Which location the AME goes to is established by the priority of the requests sent by E2 and E4, and the current location of the AME.

Once the AME has arrived to the E2 controlled location for example, E2 will begin transmitting control commands to it. Using the fan example, control commands could include blowing duration and fan speed. An AGV in this case could be instructed to remain stationary until cargo is unloaded. Once complete, E2 can release control of the AME, at which point the AME should determine its next destination.

Now suppose that the AME is called by E4 for a pickup but instead goes to E2's control location. When the AME crosses the light curtain sensor, the system will go into a fault state due to the light curtain sensor effect being mapped to digital logic domain effects. This same fault will occur if the AME goes to the E4 controlled location erroneously.

This example demonstrates how an AME, unlike a PCE, may have multiple connections to different parts of the manufacturing process at different points in time.

Simulated workpieces interact with simulated and actual AMEs in much the same way they do with PCEs. Section 3.7.2 addressed some possible approaches for the implementation of actual components into an HPS, and is applicable to AMEs. The illustration of Fig. 3.15 is a good example of how a simulated region with a simulated

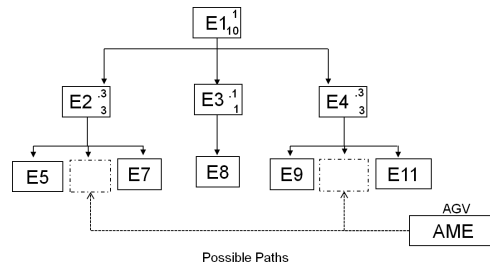


Figure 5.3: An AME such as an AGV can connect in different locations to the control hierarchy at different times.

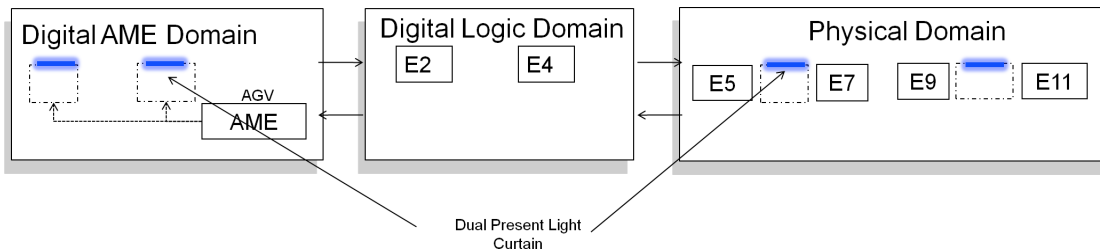


Figure 5.4: Domain mapping permits the AME to move about its simulated three dimensional space, and have the same effects within the manufacturing process it would have had in the physical domain.

AME could be implemented. Fig. 3.15 also shows how the HPS would be implemented if there were an actual AME adjacent to a simulated component. Fig 5.4 shows a digital AME domain with dual present light curtain sensors mapped to the same digital logic domain effects as the physical domain light curtain sensors. In the digital AME domain the AME still has the option of going to either location, and upon arrival, the digital AME domain light curtain sensors will activate and result in the same mapping as it would if the AME were actual.

For a digital AME domain to be a part of the HPS, effects must be mapped between the involved domains. Which domains are mapped is a function of the needs of the HPS itself. AME domain mapping can be done in different ways. Fig. 5.5 illustrates some of the possible mapping scenarios. In A, the left pointing arrow means that the Physical Domain (PD) is mapped to the Digital Logic Domain (DLD). The right pointing arrow means that the DLD is mapped to the PD. A is an



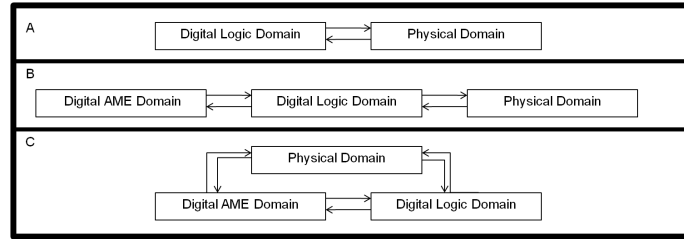


Figure 5.5: Examples of how multiple domains can be connected.

example of bidirectional mapping where effects in both domains cause effects in the other. B and C are examples of how three domains could be mapped. The example corresponding to Fig 5.4 has the mapping structure of B, where the digital AME domain is mapped to the digital logic domain.

Implementing multiple simulated AMEs within a singular AME domain is possible within the HPS approach. There can be multiple AMEs in one simulation environment, or there can be multiple simulation environments holding different AMEs. A scenario including multiple digital AME domains is permissible within the HPS approach but is not covered within the scope of this dissertation.

## 5.6 AME Equivalence

The equivalence analysis of Chapter 4 applies to AMEs in much the same way as to PCEs. The digital AME domain does not affect pre-analysis, because pre-analysis is completely based on the control hierarchy of the manufacturing process. Structural and communication equivalence are also unaffected by the use of digital AME domains. Structural equivalence is unchanged because it does not take into account the specific essences and effects of the entity. Communication equivalence is unchanged because the communication pathways are handled the same regardless of the type of component or environment they connect to.

The method for calculating leaf equivalence is unchanged; however, the result of

Table 5.2: Actual Component

<b>AGV Actual (ESA = 6)</b>	
<b>Physical Domain</b>	<b>Digital Domain</b>
Location	System Location
Volume	Status (Busy/Idle)
Mass	Current Process

Table 5.3: Simulated Component

<b>AGV Simulated (ESS = 6)</b>	
<b>Physical Domain</b>	<b>Digital Domain</b>
	System Location
	Status (busy/idle)
	Current Process
	<i>Simulated Location</i>
	<i>Simulated Volume</i>
	<i>Simulated Mass</i>

such an analysis on an AME does require some discussion. For this explanation consider the AGV example of Fig 5.3. Tables 5.2 and 5.3 show an actual and simulated AGV. The simulated AGV does not have the effects of the physical domain that are present in the actual AGV. These effects, however, are still present as simulated effects in the digital AME domain. The effects present in the digital AME domain are in italics in Table 5.3, and make the ESS equal to 6, which is also the value for the ESA. Equation 4.5 then gives the CNL to be 1, implying that the simulated AGV is equivalent to its actual counterpart from a leaf node perspective.

A leaf equivalence of 1 in a simulated entity means that all relevant essences are at least simulated. It does not mean that all actual leaf nodes are present in the system. A simulated component having a leaf equivalence of 1 is a result of the simulation's characteristics. A pure signal emulator that does not take into account location, volume, or mass would have a CNL of 0.5. The PCE simulations of Section 3.5 are examples of signal emulators with CNLs of 0.5.

When calculating a leaf equivalence that includes a simulation environment that

simulates the relevant essences from the physical domain, the meaning of the leaf equivalence is different from an HPS that only has signal emulating simulations. In the case of a simulation that simulates physical domain essences, a leaf equivalence of 1 is possible even if all of the components are simulated. If the component simulations are all signal only emulators (they do not simulate physical domain essences and effects), a leaf equivalence of 1 means that all leaf nodes are non-simulated and in place. The other equivalence quantities can also be used to clarify the current state of the HPS.

One advantage of leaf equivalence is that it can be used to compare various different simulation programs to see which has the potential to provide the most leaf equivalent simulated components. Simulation programs that have the potential for a leaf equivalence closer to 1 maybe more desirable in some cases. Additionally, the quantity can be calculated with a minimal amount of effort prior to simulated component purchase and implementation.

## **5.7 AME Example Application**

The implementation steps outlined in the remainder of this section serve as an example of how digital AME domains could be applied. In addition, a discussion of the challenges associated with their implementation is also included. The example covered in this section will be that of a human worker at a manual station. The worker's task will be to release pallets at the stop 1 location when they arrive. The implemented manual station was actually hybrid because it included an actual person with virtual controls; however, the implementation will be described from the perspective of implementing the digital AME domain alone. The source of the human worker's actions could be from a simulated entity inside the simulation program, or,

Table 5.4: The Essences and Effects of the Simulated Human Worker

Simulated Human Worker		
Physical Domain	Spatial Essence	<i>none</i>
	Newtonian Effect	<i>none</i>
Digital AME Domain	AME Data Essence	location, vel, mass
	AME Procedural Effects	change in location, vel, and mass
Digital Logic Domain	Logic Data Essence	none
	AME Procedural Effects	outputs and states listed in Fig. 5.6

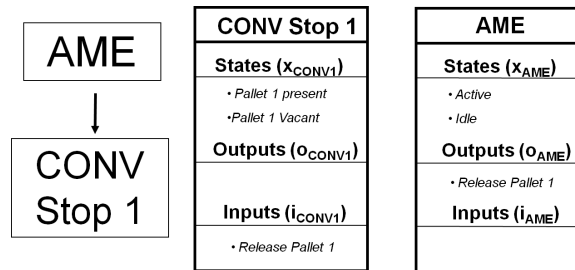


Figure 5.6: The states, inputs and outputs of the conveyor and the AME.

such as in this case, an actual human outside of it. The human implementation is specific to the simulation program and does not effect how the digital AME domain is connected and mapped to the existing system. Section 5.7.3 will, however, address the specifics of how the actual human was implemented.

### 5.7.1 Applied Implementation Methodology

Implementing the simulated manual station corresponds with scenario 2 in Section 3.4, where the system starts in actual form with the physical processes in place. The approach of Section 3.4 will be followed here to describe the implementation of the digital AME domain.

1. **Identify the entity to be added.** A virtual workstation with a human worker is added to control the stop 1 pallet stop.
2. **Define the essence and effect.** Table 5.4 lists essences of the human worker.

3. **Define the behavior of the entity.** The human worker visually sees when a pallet arrives at stop 1 and then releases it. The behavior of the human worker is a function  $o_{AME} = f_{AME}(x_{AME}, x_{CONVStop1})$  where  $o$  and  $x$  are given in Fig. 5.6. As in Chapter 3 it is assumed that the AME can see the internal states of CONV Stop 1.
4. **Define the interface.** The manual station in the digital AME domain has an OPC interface to CONV Stop 1.
5. **Build the simulation.** The AME simulation environment was built using an open source game engine (Panda3D).
6. **Deactivate the actual entity.** Because this is an addition there is no need for deactivation.
7. **Connect the simulation to the test setup.** The AME simulation environment was connected through OPC.
8. **Modify the system as needed, such that it can function correctly within the larger system and handle the transition from simulated workpieces to actual workpieces.** This particular implementation did not require workpiece transitions into the digital AME domain, only control of a part of the process from the digital AME domain.
9. **Run the test setup for validation.** The manual station was validated and observed to work as intended.

### 5.7.2 Game Engine Implementation

Because gaming technology has become such a ubiquitous phenomenon, using game engines serves as a viable option for the implementation of a digital AME domain. For the example described here Panda3D was the game engine used to create

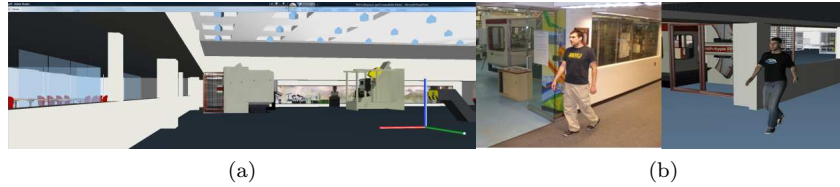


Figure 5.7: Pictures of the Panda3D environment and how it mimics the real environment

the three dimensional environment in the digital AME domain. Fig. 5.7 shows screen shots of the Panda3D simulation environment. Fig. 5.7a shows a wide shot while Fig. 5.7b shows how the Panda3D environment compares to the real environment.

Game engine environments are ideal because they allow real-time processing, physics based interactions, external communication, and are natively equipped with the functionalities needed to simulate sensors.

### 5.7.3 Digital AME Domain Mapping and Control

The Panda3D implementation demonstrates four specific characteristics:

- Mapping from the digital logic domain to the digital AME domain
- Mapping from the digital AME domain to the digital logic domain
- Macro scale (system wide location) externally controlled AME location
- Micro scale externally controlled AME movement

#### 5.7.3.1 Mapping

To demonstrate bidirectional mapping with a digital AME domain, the digital logic domain was mapped to the digital AME Domain and vice versa. Pallet stop control information was sent from the digital AME domain to the digital logic domain. Additionally, robot and CNC process information was sent from the digital logic domain to the digital AME domain such that machine processing was visible in the digital AME domain. The result of this bidirectional mapping was that a

pallet stop button activated in the digital AME domain resulted in an actual pallet being released, and when robots and CNCs begin their process in the physical domain the processing was also visible in the digital AME domain. The mapping of machine movement to the digital AME domain was not necessary for manual station functionality, but was implemented to demonstrate bidirectional mapping.

#### **5.7.3.2 Macro and Micro Scale AME Control**

AME location control was addressed in two ways, one on a macro level and one on a micro level. The macro approach was chosen to demonstrate how a manufacturing HPS could implement simulated AME position on a plant wide scale when receiving the AME location information from an external source. The micro approach was used to demonstrate how the manufacturing system could implement position information on a local scale (tens of inches) when receiving the AME location information from an external source.

The macro level AME implementation was done via user login. Each user logs in to tell the digital AME domain their current location. After the digital AME domain receives this information, an AME with their likeness is placed in the corresponding location of the digital AME domain.

To demonstrate the micro level AME implementation, three dimensional tracking was added to the Panda3D environment. A computer running ARToolKit was equipped with a web cam and used to physically track a person wearing a specific character. This location information was then sent to the Panda3D environment and resulted in the camera moving in simulated coincident three dimensional space. This enabled the person to look around the digital AME domain's simulated three dimensional space by moving their body and changing their perspective, while also having control over CONV Stop 1.

The movement mapping to the camera in the Panda3D environment also demonstrates how a real AME could be portrayed in the simulated environment. If the HPS approach is truly capable of implementing AMEs, a real AME's effects must be mapped to the digital AME domain as well as to the digital logic domain. Additionally, for simulation purposes the movement information sent from ARToolkit could have just as easily been from an AME simulation. The Panda3D environment is unaware of whether the external AME is simulated or actual, allowing the HPS to maintain its distributed structure.

#### **5.7.4 Communication**

The mapping described in this section provides the communication capabilities between domains (Fig. 5.8). The machines and robots (physical domain) communicate with the system level controller (digital logic domain) over DeviceNet. Communication between the digital logic domain and the Panda3D digital AME domain is done by a Java program that monitors manufacturing system status using OPC, and then strobes that information to the Panda3D environment via UDP messages over ethernet. The Java program also waits for any status change to be sent from the Panda3D environment, at which point it updates the logic domain.

The result of the mapping and tracking is that a human worker could interact with the process by controlling a pallet stop through the Panda3D environment. The human worker can decide when to release a pallet based on what he or she observes of the process.

### **5.8 Summary**

This chapter expanded the HPS approach to incorporate AMEs into the existing conceptual architecture, implementation methodology, and equivalence analysis. A



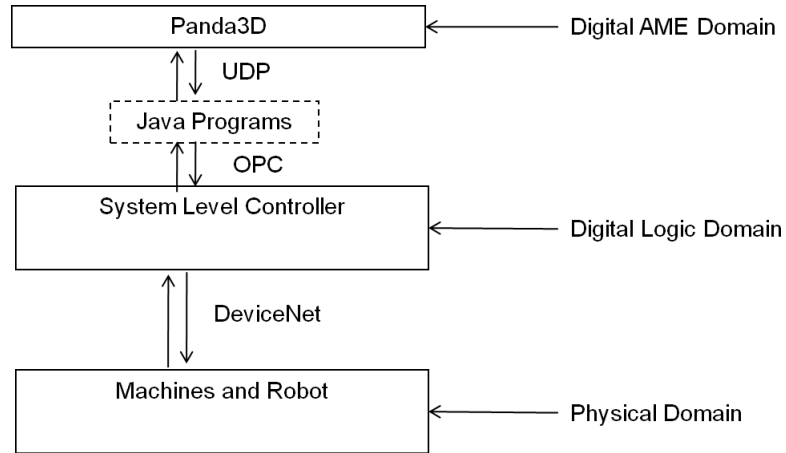


Figure 5.8: The Panda3D domain is bidirectionally mapped to the digital logic domain.

new domain termed the digital AME domain has been presented that allows AMEs to exist in a physics simulated virtual environment that can function within the existing HPS. AMEs such as humans and AGVs can now be included in validation and testing using the HPS approach, expanding it beyond PCEs.

## CHAPTER VI

### Future Technical Landscape, Conclusions, and Future Work

#### 6.1 Contributions

In this dissertation an ontology for an HPS and a formalized method for its application have been presented. The ontology allows an HPS to be broken down into conceptual components with attributes that are not application specific. The ontology was then applied to three different scenarios on a manufacturing line, including: adding a manufacturing cell that was not present previously, replacing a real robot with a simulated robot, and replacing an existing cell with a simulated cell. Included within these scenarios, is how a simulated workpiece can transition in and out of simulated regions.

This dissertation also presented an analysis methodology for three types of HPS equivalence metrics: structural equivalence, leaf equivalence, and communication equivalence. Equivalence analysis allows a quantitative comparison of the HPS's current state with that of the process after full physical implementation. The equivalence analysis methodology has been applied to two HPS scenarios which include the Reconfigurable Factory Testbed and the Manufacturing System Emulation Environment.

Lastly, the HPS conceptual architecture is expanded to include autonomous mo-

bile entities (AME). A new digital domain termed the digital AME domain is presented to allow an AME to move about a physics simulated three dimensional environment, while still existing as a part of the HPS as a whole. An implementation example is put forth in which a digital AME domain is created and mapped to the digital logic domain of the HPS.

## 6.2 Development of Technical Landscape

In order for the HPS approach to realize its full potential, three developments must be made.

- Vendors must make component simulations available that are interchangeable with their actual counterparts.
- Vendors must make actual components capable of processing simulated work-pieces.
- Process planners must adopt the “try before you buy” philosophy of process development.

The motivation for these developments can start from either the process planner or the vendors. The vendors are motivated to make this migration to component simulations because they can show more easily how their component compares to competing vendors, aiding them in attaining new customers. Process planners are motivated because they can compare existing solutions before making a complete financial commitment to one. The component simulations also can allow them to move ahead sooner with installation into the hardwired simulation phase, and then install physical components as they arrive with less difficulty due to the interchangeability of the component simulations.

### **6.3 Resulting Industrial Environment and Impact**

If the HPS approach were to be fully embraced by a large fraction of the industry, the impact on the industrial environment would be a new implementation procedure. Not only would all process planners be accustomed to the “try before you buy” philosophy of component purchase, but training could also be a part of the implementation procedure. Users of the components (programmers or operators) could begin to interact and be trained on the simulated version of the component before its actual counterpart arrives. Additionally, component simulations are much more portable than their actual counterparts; therefore, once component simulations have been used for one process implementation, those same simulated components can be moved to the next manufacturing process deployment site, and if newer or slightly different components are to be used from the same vendor, the component simulation could simply go through an upgrade instead of being a brand new purchase.

### **6.4 Future Work**

The assumptions described in Section 1.3 helped to define the scope of the challenges addressed in this dissertation. Future work could begin by relaxing some of these assumptions. One of the main assumptions is that all addressed processes are event based and non-continuous. This assumption could be relaxed by applying the HPS approach to continuous processes such as chemical processes. Ghosting and birthing might be handled differently because the workpiece is also not discrete. Relaxing the assumption about the quality of the simulation is another possible topic for future work. Exploring how simulation quality affects the HPS as a whole would help process planners understand if it is possible to use the HPS for things other than testing integration, communication, and logic. With high enough quality simulations

throughput could possibly be studied in the hardwired simulation phase. Finally the assumption that the interface of the components are only those of event based control can also be addressed by future work. Adding components where force and velocity are transferred at the interface would make the HPS approach more versatile but could raise issues dealing with bandwidth.

#### **6.4.1 Unidirectional Mapping**

Possible future work for the HPS approach includes developing a more general set of rules for domain implementation and bidirectional versus unidirectional mapping. The current HPS approach includes the development of three types of domains. There is no limit, however, to the number of possible domains present in an HPS. The number of domains depends on the needs of the current implementation. The work done in HPS begins to address this issue but requires more development. Rules outlining when to create a domain and what is essential to a generic domain need to be developed.

Bidirectional and unidirectional mapping are addressed in this dissertation, however, the implications of a unidirectional mapping with a domain such as the digital AME domain are not addressed. Unidirectional mapping to a domain that can visually represent the process gives that domain a monitoring capability. Future work could explore the implications of this in the conceptual architecture.

The author has done some work with unidirectional mapping where Second Life was used as a monitoring domain accessible over the internet (Fig. 6.1). Second Life is a three dimensional game engine environment where users all over the world can log in with their avatars and interact with the environment and each other. The RFT's four CNCs, and two robots are portrayed in the Second Life environment. When the robots or CNCs begin their process, it is viewable in the Second Life environment



Figure 6.1: A screen shot of the RFT in Second Life.

much the same way that the process is viewable in the Panda3d environment. Future work can explore the implications and usability of implementations such as this one.

#### **6.4.2 Parallel Component Monitoring**

Future work should also investigate the implementation of component simulations in parallel with their actual counterpart. Once run in parallel these simulations can be used to monitor their actual counterpart. Actual components monitored by simulated components can provide a much deeper insight into the process as it functions in real time.

#### **6.4.3 Implementation Studies**

Lastly, future work should include implementation studies where the HPS approach is compared to the traditional approach of manufacturing process deployment. This comparison would show in what conditions the approach is truly beneficial.

## BIBLIOGRAPHY

## BIBLIOGRAPHY

- [1] *Fidelity*, Tech. report, VV&A RPG Specail Topic, 2000.
- [2] *PABADIS‘PROMISE: Work package 8, system simulation, deliverable 8.3 common demonstration system*, Tech. Report FP6-IST-016649, <http://www.pabadis-promise.org/>, 2008.
- [3] *White paper: Structure and behavior of a PABADIS‘PROMISE system*, Tech. report, 2008.
- [4] P. Andreas, J. Fransoo, and A. Kok, *What determines product ramp-up performance?: a review of characteristics based on a case study at Nokia Mobile Phones*, Tech. report, Technische Universiteit Eindhoven, 2007.
- [5] F. Auinger, M. Vorderwinkle, and G. Buchtela, *Interface driven domain-independent modeling architecture for “soft-commissioning” and “reality in the loop”*, Proceedings of the 1999 Winter Simulation Conference, 1999.
- [6] Balaguer B., S. Balakirsky, S. Carpin, M. Lewis, and C. Scrapper, *USARSim: a validated simulator for research in robotics and automation*, In Workshop on Robot Simulators: Available Software, Scientific Applications, and Future Trends, 2008.
- [7] M. Bacic, *On hardware-in-the-loop simulation*, 44th IEEE Conference on Decision and Control, 2005.
- [8] Marko Bacic, *Two-port network modelling for hardware-in-the-loop*, Proceedings of the American Control Conference, 2007.
- [9] T Baines, S Mason, P. Siebers, and J. Ladbrook, *Humans: the missing link in manufacturing simulation?*, Simulation Modelling Practice and Theory **12** (2004), no. 7-8, 515 – 526.
- [10] S. Balakirsky, F. Proctor, C. Scrapper, and T. Kramer, *An integrated control and simulation environment for mobile robot software development*, Proceedings of IDETC/CIE 2008 ASME 2008 International Design Engineering Technical Conference & Computers and Information in Engineering Conference, 2008.
- [11] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis, *USARSim: providing a framework for multi-robot performance evaluation*, Performance Metrics for Intelligent Systems Workshop—2006 — PerMIS, 2006.
- [12] S. Balakirsky, C. Scrapper, and E. Messina, *Mobility open architecture simulation and tools environment*, Knowledge intensive multi-agent systems conference, 2005.
- [13] J. Banks, J. Carson, B. Nelson, and D. Nicol, *Discrete-event system simulation*, Prentice Hall, Inc., 2001.
- [14] F. Biocca and M. Levy, *Communication in the age of virtual reality*, Lawrence Erlbaum Associates, 1995.
- [15] C. Boer, A. de Bruin, and A. Verbraeck, *A survey on distributed simulation in industry*, Journal of Simulation **3** (2009), 3–16(14).



- [16] C. Boer, A Verbraeck, and Hans Veeke, *The possible role of a backbone architecture in real-time control*, Proceedings of the 2002 Winter Simulation Conference, 2002.
- [17] J. Dahmann, R. Fujimoto, and R. Weatherly, *The department of defense high level architecture*, WSC '97: Proceedings of the 29th Conference on Winter Simulation (Washington, DC, USA), IEEE Computer Society, 1997, pp. 142–149.
- [18] I. Davila-Rios, L.M. Torres-Trevino, and I. Lopez-Juarez, *On the implementation of a robotic welding process using 3D simulation environment*, Electronics, Robotics and Automotive Mechanics Conference, 2008. CERMA '08, 2008, pp. 283–287.
- [19] R. Diogo, C. Vicari, E. Loures, M. Buseti, and E. Santos, *An implementation environment for automated manufacturing systems*, Proceedings of the 17th World Congress for the International Federation of Automatic Control, 2008.
- [20] S. Elkosantini and D. Gien, *Integration of human behavioural aspects in a dynamic model for a manufacturing system*, International Journal of Production Research **47** (2009), no. 10, 2601–2623.
- [21] E. Endsley, *Modular finite state machines for logic control: Theory, verification and applications to reconfigurable manufacturing systems*, Ph.D. thesis, University of Michigan, 2004.
- [22] Y. Erensal and E. Duman, *An experimental analysis of human factors on the system performance*, 35th International Conference on Computers and Industrial Engineering, 2005.
- [23] L. Ferrarini and A. Dede, *A mixed-reality approach to test automation function for manufacturing systems*, Conference on Innovation, Research and Development, 2009.
- [24] R. Fujimoto, *Parallel simulation: parallel and distributed simulation systems*, WSC '01: Proceedings of the 33rd conference on Winter Simulation, 2001, pp. 147–157.
- [25] P. Gawthrop, D. Virden, S. Neild, and D. Wagg, *Emulator-based control for actuator-based hardware-in-the-loop*, Control Engineering Practice **16** (2008), 897–908.
- [26] B. Gerkey, R. Vaughan, and A. Howard, *The player/stage project: Tools for multi-robot and distributed sensor systems*, Proceedings of the International Conference on Advanced Robotics, 2003.
- [27] W. Grega, *Hardware-in-the-loop simulation and its application in control education*, 29th ASEE/IEEE Frontiers in Education Conference, 1999.
- [28] D. Gross, *Report from the fidelity implementation study group*, Tech. report, Fall Simulation Interoperability Workshop, 1999.
- [29] F. Gu, W. Harrison, D. Tilbury, and C. Yuan, *Hardware-in-the-loop for manufacturing automation control: current status and identified needs*, Proceedings of the Third Annual IEEE Conference on Automation Science and Engineering, 2007.
- [30] H. Hanisch, A. Lobov, M. Lastra, R. Tuokko, and V. Vyatkin, *Formal validation of intelligent-automated production systems: towards industrial applications*, International Journal of Manufacturing Technology and Management **8** (2006), 75–106.
- [31] W. Harrison and D. Tilbury, *Virtual fusion: hybrid process simulation and emulation-in-the-loop*, Proceedings of the 9th Biennial ASME Conference on Engineering Systems Design and Analysis, 2008.
- [32] W. Harrison, D. Tilbury, and C. Yuan, *From hardware-in-the-loop to hybrid process simulation: An ontology for the implementation phase of manufacturing systems*, submitted to IEEE Transactions on Automation Science and Engineering (2010).

- [33] E. Henderson, *How real is virtual real?*, Colloquium on Virtual Reality Person Mobile and Practical Applications, 1998.
- [34] S. Heo, K. Park, and H. Ahn, *Design and implementation of hardware-in-the-loop simulator for EHB systems*, SICE Annual Conference, 2003.
- [35] H. Hibino, T. Inukai, and Y. Fukuda, *Efficient manufacturing system implementation based on combination between real and virtual factory*, International Journal of Production Research **44** (2006), no. 18, 3897–3915.
- [36] Hironori Hibino and Yoshiro Fukuda, *Emulation in manufacturing engineering processes*, WSC '08: Proceedings of the 40th Conference on Winter Simulation, 2008, pp. 1785–1793.
- [37] J. Jackson, *Microsoft robotics studio: A technical introduction*, Robotics Automation Magazine, IEEE **14** (2007), no. 4, 82–87.
- [38] Sanjay Jain, Frank Riddick, Andreas Craens, and Deogratias Kibira, *Distributed simulation for interoperability testing along the supply chain*, WSC '07: Proceedings of the 39th Conference on Winter Simulation, 2007, pp. 1044–1052.
- [39] S. Jang, T. Park, and C. Han, *A control of vehicle using steer-by-wire system with hardware-in-the-loop-simulation system*, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2003.
- [40] S. Kanai, T. Miyashita, and T. Tada, *A multi-disciplinary distributed simulation environment for mechatronic system design enabling hardware-in-the-loop simulation based on HLA*, International Journal on Interactive Design and Manufacturing **1** (2007), 175–179.
- [41] J. Kelsick and J. Vance, *Discrete event simulation implemented in a virtual environment*, Journal of Mechanical Design **125** (2003), 428–433.
- [42] Deogratias Kibira and Charles R. McLean, *Generic simulation of automotive assembly for interoperability testing*, WSC '07: Proceedings of the 39th Conference on Winter Simulation, 2007, pp. 1035–1043.
- [43] P. Kneppell and D. Arango, *Simulation validation*, IEEE Computer Society Press, 1993.
- [44] A. Lassila, S. Sameh, T. Perera, T. Koch, and J. Chrobot, *Modelling and simulation of human-centred assembly systems - a real case study*, IFIP International Federation for Information Processing **159** (2005), 405–412.
- [45] W. Lee, M. Yoon, and M. Sunwoo, *A cost- and time-effective hardware-in-the-loop simulation platform for automotive engine control systems*, Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 2002.
- [46] Peter Lendermann, *About the need for distributed simulation technology for the resolution of real-world manufacturing and logistics problems*, WSC '06: Proceedings of the 38th Conference on Winter Simulation, 2006, pp. 1119–1128.
- [47] M. Lombard and Theresa Ditton, *At the heart of it all: The concept of presence*, Journal of Computer-Mediated Communication **3(2)** (1997), Available online: <http://www.ascusc.org/jcmc/vol3/issue2/lombard.html>.
- [48] M. MacDiarmid and M. Bacic, *Quantifying the accuracy of hardware-in-the-loop*, Proceedings of the 2007 American Control Conference, 2007.
- [49] D. Maclay, *Simulation gets into the loop*, IEE Review **43** (1997), 109–112.
- [50] A. Matta, M. Tomasella, and A. Valente, *Impact of ramp-up on the optimal capacity-related reconfiguration policy*, International Journal of Flexible Manufacturing Systems **19** (2007), 173–194.

- [51] C. McLean, S. Jaen, A. Craens, and D. Kibira, *A virtual manufacturing environment for interoperability testing*, Proceedings of the Delft University of Technology Conference, 2007.
- [52] Charles McLean, Sanjay Jain, Frank Riddick, and Y. Tina Lee, *A simulation architecture for manufacturing interoperability testing*, SCSC: Proceedings Of The 2007 Summer Computer Simulation Conference, 2007, pp. 601–608.
- [53] Charles McLean, Frank Riddick, and Y. Tina Lee, *An architecture and interface for distributed manufacturing simulation*, SIMULATION **81** (2005), no. 1, 15–32.
- [54] J. Moyne, J. Korsakas, and D. Tilbury, *Reconfigurable factory testbed RFT: A distributed testbed for reconfigurable manufacturing systems*, Proceedings of the Japan-USA Symposium on Flexible Automation, 2004.
- [55] D. Pace, *Modeling and simulation verification and validation challenges*, Johns Hopkins Applied Technical Digest **25** (2004), 163–172.
- [56] D. Pace and J. Sheehan, *Peer use in M&S V&V*, Tech. report, The Foundation for V&V in the 21st Century Workshop, 2002.
- [57] T. Park, C. Han, and S. Lee, *Development of the electronic control unit for the rack-actuating steer-by-wire using the hardware-in-the-loop simulation system*, Mechatronics **15** (2005), 899–918.
- [58] L. Pedro, A. Dias, L. Massaro, and G. Caurin, *Dynamic modelling and hardware-in-the-loop simulation applied to a mechatronic project*, Proceedings of COBEM 2007 (19th International Congress of Mechanical Engineering), 2007.
- [59] B. Powell, N. Sureshbabu, and M. Dunn K. Bailey, *Hardware-in-the-loop vehicle and powertrain analysis and control design issues*, Proceedings of the American Control Conference, 1998.
- [60] D. Ramaswamy, R. McGee, S. Sivashankar, A. Deshpande, J. Allen, K. Rzemien, and W. Stuart, *A case study in hardware-in-the-loop testing: Development of an ECU for a hybrid electric vehicle*, SAE Technical Paper Series **01** (2004).
- [61] J. Rumbaugh and M. Blaha, *Object-oriented modeling and design with UML*, Alan Apt, 2005.
- [62] G Shao, S. Leong, and C. McLean, *Simulation-based manufacturing interoperability standards and testing*, Proceedings of the 9th International Conference on Progress of Machining Technology, 2009.
- [63] B. K. Taylor, S. Balakirsky, E. Messina, and R. D. Quinn, *Design and validation of a Whegs robot in USARSim*, PerMIS '07: Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems, 2007, pp. 105–112.
- [64] L. Vin, Oscarsson J., A. Ng, M. Jagstam, and T. Karlsson, *Manufacturing simulation: good practice, pitfalls, and advanced applications*, Proceedings of the 21st International Manufacturing Conference (IMC21), 2004.
- [65] J. Viswanathan, W. Harrison, D. Tilbury, and F. Gu, *Using hybrid process simulation to evaluate manufacturing system component choices: Integrating a virtual robot with the physical system*, Proceedings of the 2011 Winter Simulation Conference, 2011(submitted).
- [66] R. Wells, J. Fisher, Y. Zhou, B. Johnson, and M. Kyte, *Hardware and software considerations for implementing hardware-in-the-loop traffic simulation*, The 27th Annual Conference of the IEEE Industrial Electronics Society, 2001.
- [67] G. Zulch, *Modelling and simulation of human decision-making in manufacturing systems*, WSC '06: Proceedings of the 38th Conference on Winter Simulation, dec. 2006, pp. 947–953.