ENGINEERING COMPLEX SYSTEMS WITH MULTIGROUP AGENTS

by

DENISE MARIE CASE

MSE, Kansas State University, 2013

B.S. ChE, University of Missouri at Columbia, 1985

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Abstract

As sensor prices drop and computing devices continue to become more compact and powerful, computing capabilities are being embedded throughout our physical environment. Connecting these devices in *cyber-physical systems* (CPS) enables applications with significant societal impact and economic benefit. However, engineering CPS poses modeling, architecture, and engineering challenges and, to fully realize the desired benefits, many outstanding challenges must be addressed. For the *cyber* parts of CPS, two decades of work in the design of autonomous agents and *multiagent systems* (MAS) offers design principles for distributed intelligent systems and formalizations for *agent-oriented software engineering* (AOSE). MAS foundations offer a natural fit for enabling distributed interacting devices. In some cases, complex control structures such as *holarchies* can be advantageous. These can motivate complex organizational strategies when implementing such systems with a MAS and some designs may require agents to act in multiple groups simultaneously. Such agents must be able to manage their multiple associations and assignments in a consistent and unambiguous way. This dissertation shows how designing agents as *systems of intelligent subagents* offers a flexible, reusable approach to designing complex systems. It shows how a set of flexible, reusable components were developed to create a new *organization-based agent architecture*, OBAA^{++}, specifically designed for *multigroup agents*. It presents the *Adaptive Architecture for Systems of Intelligent Systems* (AASIS), a new framework and system architecture that uses OBAA^{++} to enable both *complex, multigroup MAS* and systems of systems. This work illustrates the reusability and flexibility of the approach by using AASIS to simulate a CPS for an *intelligent power distribution system* (IPDS) operating two complex MAS concurrently: one providing continuous voltage control and a second conducting discrete power auctions near sources of distributed generation.

ENGINEERING COMPLEX SYSTEMS WITH MULTIGROUP AGENTS

by

DENISE MARIE CASE

MSE, Kansas State University, 2013

B.S. ChE, University of Missouri at Columbia, 1985

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
Scott A. DeLoach

Copyright

Denise Marie Case

2015

Abstract

As sensor prices drop and computing devices continue to become more compact and powerful, computing capabilities are being embedded throughout our physical environment. Connecting these devices in *cyber-physical systems* (CPS) enables applications with significant societal impact and economic benefit[1]. However, engineering CPS poses modeling, architecture, and engineering challenges[2] and, to fully realize the desired benefits, many outstanding challenges must be addressed[3]. For the *cyber* parts of CPS, two decades of work in the design of autonomous agents and *multiagent systems* (MAS) offers design principles for distributed intelligent systems and formalizations for *agent-oriented software engineering* (AOSE)[4]. MAS foundations offer a natural fit for enabling distributed interacting devices[5]. In some cases, complex control structures such as *holarchies* can be advantageous[6,7,8]. These can motivate complex organizational strategies when implementing such systems with a MAS[9] and some designs may require agents to act in multiple groups simultaneously[7]. Such agents must be able to manage their multiple associations and assignments in a consistent and unambiguous way. This dissertation shows how designing agents as *systems of intelligent subagents* offers a flexible, reusable approach to designing complex systems. It shows how a set of flexible, reusable components were developed to create a new *organization-based agent architecture*, OBAA^{++}, specifically designed for *multigroup agents*. It presents the *Adaptive Architecture for Systems of Intelligent Systems* (AASIS), a new framework and system architecture that uses OBAA^{++} to enable both *complex, multigroup MAS* and systems of systems. This work illustrates the reusability and flexibility of the approach by using AASIS to simulate a CPS for an *intelligent power distribution system* (IPDS) operating two complex MAS concurrently: one providing continuous voltage control and a second conducting discrete power auctions near sources of distributed generation.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Definitions

Acknowledgments

First and most, I thank my advisor, Dr. Scott DeLoach, for his guidance and advice, and for instructing me always with wisdom, efficiency, order, and good humor; there is still so much to learn. I thank my committee members, Dr. Anil Pahwa, Dr. Gurdip Singh, Dr. Gabriel Nagy and Dr. Dan Andresen, for their constructive comments and insightful questions, and the many outstanding researchers with whom I've worked including Dr. Torbin Amtoft, Dr. Doina Caragea, Dr. Sanjoy Das, Dr. William Hsu, Dr. Bala Natarajan, Dr. Mitchell Neilsen, Dr. Xinming Ou, Dr. Jacqueline Spears, Dr. Eugene Vasserman, and my research communities in AAMAS and MICAI. I've loved every moment; I could not have wished for a better experience.

I am grateful to my extended team: those who went before me, Dr. Matthew Miller, Dr. Jorge Valenzuela, Dr. Chris Zhong, Rui Zhang, and and those who worked beside me on our multi-year project, including Bodhisattwa Majumder, Mohammad Faqiry, Xiaolong Wang, Shafiul Alam, and especially Ahmadreza Malekpour. I thank our student developers, Matthew Brown and Greg Martin, for their contributions, and the lab partners from whom I've learned so much, including Dr. Scott Bell, outstanding mentor, grad student, and professor, and Matthew Cholick, a truly exceptional engineer.

I owe a deep debt of gratitude to the friends who encouraged and supported me throughout this journey, those I brought with me and those I met here, including Art, Bob, Brian, Emily, Greg, Heath, Janice, Jason, Joe, Karen, Karen, Kevin, Lanie, Laura, Lindsey, Lucas, Maureen, Michelle, Nancy, Nathan, Pam, Randy, Rick, Scott, Una, Vicki, and especially Andy. I could not have done it without you.

And to my family for their ongoing support while I ran off to school, I owe you. From my parents, the JPL engineer and editor extraordinaire married to the cutest, sweetest girl in

town, to my wonderful brothers and their families, our dinner-time conversations paved the way for a love of science that continues to bring great joy. And mostly, of course, to Steve, Meghan, Lilly, Eden, Travis, and Christine. You are still and will always be the greatest of all I've ever had anything to do with. I am so very grateful.

Dedication

To all beings throughout space and time,
naturally and artificially intelligent,
may our curiosity and investigations
realize something wonderful.

Preface

When I first entered college, I meant to stay for quite a while. My love was medicine and neuroscience and I'd hoped an MD and PhD would enable working at the forefront of an exciting and important field. While finishing my bachelor's degree, I discovered two people that I loved even more than school and I spent the most wonderful years being head room mom, filling the bleachers with extended family at soccer and baseball games, and watching a delightful group of children grow into amazing adults. When they finished college, my interest returned, and I was offered a most perfect chance to pick it up again. I find many of the fields I love are converging, and the areas of natural and artificial intelligence continue to evolve along fascinating and exciting trajectories. I am grateful to have had the chance to come, learn, and participate in the exciting world of science.

Chapter 1

Introduction

What we do, if we are successful,
is to stir interest in the matter at hand.
— *Julius Sumner Miller*[20]

This research effort proposes and evaluates abstractions, architectural elements, and a framework for designing *complex systems* in a way that supports the distributed design, development, and implementation of system components by multiple contributors in a flexible, reusable manner.

This chapter provides an introduction to the work and its motivation in Section 1.1, and summarizes the approach, thesis, and contributions of the work in Sections 1.2, 1.3, and 1.4, respectively. An overview of the document is provided in Section 1.5.

1.1 Motivation

As sensor prices drop and computing devices continue to become more compact and powerful, computing capabilities are being embedded throughout our physical environment.

Connecting these devices in *cyber-physical systems* (CPS) enables applications with significant societal impact and economic benefit, spanning many critical sectors including energy, transportation, healthcare, manufacturing, buildings, communities, agriculture, defense, and aerospace[1,21]. However, engineering CPS poses modeling, architecture, and engineering challenges[2] and to fully realize the desired benefits, many outstanding challenges must be addressed[3]. Several important research needs in CPS have been identified. Specifically, work in abstractions and architectures for CPS is described as urgently needed[22] and grand challenges in CPS have been identified that cross multiple sectors and areas of research[1].

Recent research efforts in the area of distributed cyber-physical control systems are making use of complex control structures such as *holarchies*[6,8] (Definition 2.20). When implementing such systems in a *multiagent system* (MAS) (Definition 2.7), correspondingly capable organizational strategies may be required[9]. Additionally, CPS (Definition 2.4) running on critical infrastructure or other valuable physical systems may be employed to support multiple cyber-systems, each pursuing a different set of goals and focusing on different problem domains[23].

In some cases, complex, multigroup organizational designs (Definition 2.14) in MAS may be warranted[24]. Implementing a MAS with multiple groups can provide additional flexibility and redundancy, and can reduce the communication overhead required. However, implementation in multiple groups introduces additional challenges related to integrating the groups and managing consistency between them. These complex (multigroup) MAS (Definition 2.16) may be structured in different ways depending on the application. Multigroup organizational designs may be flat or hierarchical. Groups may be temporary, as in possibly short-lived *coalitions*, or they may form more permanent structures such as open societies where individual agents come and go. Groups may operate under designated intermediaries as in federations[25] or use compound structures that include several different types of organizations. Various approaches have been proposed to offer flexible, reusable mechanisms for managing functionality required to form complex structures, such as a head-and-body

approach for designing *holonic agents*[9].

Some organizational designs require an agent to be part of more than one group simultaneously[26]. In such applications, an agent may be called upon to combine, create, or relay information *between* their multiple groups. Designing and building agents for these complex systems can be more difficult than developing traditional agents as the agent must be able to align its goals effectively while receiving goals from different groups.

The desire to develop additional architectural support for complex MAS lies in part with their flexibility and scalability. Decreasing device sizes and costs, combined with an increasing ability to process large amounts of information efficiently has resulted in significant research investments in these areas. Complex MAS are suitable for a variety of applications requiring complicated problem solving, offering the ability to employ distributed reinforcement learning and iterative, dynamic state-based reasoning[27].

One area well-suited to the application of complex MAS is that of electrical power distribution systems (PDS). PDS (Definition 2.5) carry electricity from power generation facilities to customers through a series of distribution lines and transformers. By their nature, PDS are generally hierarchically distributed, and, with cross-ties and interconnects, PDS may include complex interaction patterns between the various levels. The idea of architecting *intelligent power distribution systems*, or *smart grids*, as *holarchies* is receiving research attention[28,8]. As with other critical infrastructure CPS, intelligent PDS offers significant potential benefits coupled with significant challenges. Active research areas include voltage control[29], islanded operation and microgrids[30,31,32], online future markets[33], and other related areas[34].

To support complex systems such as an intelligent PDS, an architecture is needed that can systematically address the challenges of both complex systems and complex organizational structures within a system.

1.2 Approach

Our approach focused on developing reusable components to support both complex organizational structures within a system and the integration of multiple, possibly complex systems into integrated systems of intelligent systems (Definition 2.13).

For the cyber- or software-parts of CPS (Definition 2.4), our approach relies on two decades of work in the design of *autonomous agents* and *multiagent systems* (MAS). Prior work in this area provides key design principles for distributed *intelligent systems* and formalizations for *agent-oriented software engineering* (AOSE)[4].

Specifically, an approach was needed that was capable of supporting the holonic control algorithms being developed where intermediate devices act both to aggregate more distributed content, while concurrently participating as part of a higher-level control holon, and focused our work on developing an approach that enabled an agent to be part of multiple groups simultaneously[26].

To support the ability for an agent to participate in multiple organizations simultaneously, a new architecture was required that would:

- Be based on sound software engineering principles. There should be a clear *separation of concerns*[35] between the various organizations in which the agent participates. Additionally, each organization should be represented as a separate, cohesive component in the architecture and each of those components should be weakly coupled to the others. The architecture should provide a pattern that can be used for agents participating in one or more organizations.

- Provide architectural features that inherently support group participation. Rather than requiring significant hard-coded configurations for multiple organizations, the architecture should provide built-in support for participation in multiple (one or more), dynamically adapting organizations that may change over time.

- Provide built-in support for both *inter*-agent and *intra*-agent communication and pro-

cessing. The architecture should enable communication between agents residing on distributed physical hosts as well as for the internal communication required by the agent to manage its participation in multiple organizations.

Architecting complex, adaptive systems can be challenging. When the project began, no standard mechanisms were found for engineering agents specifically designed to enable complex cooperative systems by integrating collections of complex multigroup MAS.

1.3 Thesis Statement

Implementing systems of intelligent systems with multigroup agents offers a flexible, reusable approach for engineering complex systems.

This dissertation presents the novel *multigroup agent architecture* developed specifically for implementing complex systems. It introduces the design of each agent as a system of intelligent sub-agents and shows how this approach allows agents to act in multiple groups in a single system concurrently (e.g., one focused on voltage control). Further, it shows how the same mechanisms can be employed to allow agents to accept goals from multiple systems, driven by a different set of goals, at the same time. It includes experimental implementations with a collection of multigroup agents in a complex *intelligent power distribution system* (IPDS) that supports both grid control goals for volt-var management (as might be issued from an electrical distribution control center) and future market auction goals (as may be issued from independent market organizations).

This dissertation describes how multigroup agents enable a new level of *self control* that provides new supports for agent reasoning and intelligence. For example, IPDS agents associated with distributed sources of electrical generation may not fall completely under centralized grid control, and they may not fall completely under market control, although they can agree to and be authorized to support either or both systems. Multigroup agents

are capable of not only accepting and issuing goals from various independently-owned organizations, but can also be configured to carry out more personal goals for the primary stakeholder or owner of the associated equipment (e.g., photovoltaic solar generation panels). Applying independent goal-driven behavior to each agent is novel as agents typically carry out the goals of a single organization without this additional level of self control, reasoning, or autonomy.

Reusability is demonstrated by providing:

1. A standard goal-driven multigroup agent architecture.
2. Application of common organization-control features across problem domains and organizational designs.
3. Standard practices for specifying the desired behavior of complex, multigroup organizations.
4. Standard practices for implementing agents capable of operating the organizations specified.

Flexibility is demonstrated by:

1. Using the recommended practices and mechanisms to specify the desired behavior for a complex multigroup MAS organization.
2. Using the same practices to specify the desired behavior for a second, independent complex MAS organization operating in a different, but related problem domain.
3. Using the recommended practices and mechanisms to *implement intelligent agents* capable of operating both complex, independently-goal-driven systems concurrently.
4. Presenting experimental results based on trials demonstrating the success of the agents to issue and accept goals for different organizations united under a common set of system goals.

5. Presenting experimental results based on trials demonstrating the success of the agents to issue and accept goals for different systems, working towards a set of goals in a different problem domain.

1.4 Contributions

Contributions of this research include:

- A new *organization-based agent architecture*, OBAA^{++}, for implementing complex MAS[36].
- The *Adaptive Architecture for Systems of Intelligent Systems* (AASIS), a new framework and system architecture for implementing complex MAS applications with OBAA^{++} *multigroup agents*.
- A standard decomposition and specification process for defining goal-driven systems arranged in *complex organizations* concurrently operated by a *multigroup MAS*.
- Standard mechanisms for agents to manage *personal prioritization and execution* of assigned goals in a way that reflects the personal biases of the agent's owner or owners[37], including mechanisms for *conflict detection and management*, *bias management*, *resource management*, and *reasoning with utility functions*.
- *Engineering recommendations* for designing and creating systems of intelligent systems with multigroup agents[38,39].

This dissertation presents the AASIS framework for complex systems and the underlying OBAA^{++} agent architecture that together provide a flexible, reusable approach to implementing agents for complex systems. Based on my research, I believe this is first architecture specifically designed to support agents as systems of subagents operating under the guidance of multiple organizations and multiple systems concurrently.

1.5 Overview

This chapter provides an introduction to the work and its motivation, and summarizes the approach, thesis, and contributions of the work.

The rest of the document is organized as follows. Key definitions and a supporting conceptual model are presented in Chapter 2. Background information and related work is presented in Chapter 3 and an updated organization-based agent architecture for single organization MAS is described in Chapter 4. The new OBAA^{++} agent architecture for single-organization MAS is presented in Chapter 5. Chapter 6 introduces AASIS along with standard algorithms for specifying functionality in complex systems. The chapter includes standard approaches for customizing agent behaviors, and introduces standard mechanisms for managing goal consistency when agents receive assignments from multiple organizations and systems. Chapter 7 describes the Adaptive Organization-based Multiagent Systems Engineering (AO-MaSE) process, the recommended software engineering process for implementing complex systems with multigroup agents. The evaluation of OBAA^{++} and AASIS for complex CPS are presented in Chapter 8, including application to a *system of intelligent systems*, a sample *intelligent power distribution system* where agents operate two complex multiagent systems concurrently, one focused on grid volt-var control[23] and a second focusing on online auctions[40]. Chapter 9 summarizes the work, and offers conclusions and recommendations for future research.

Chapter 2

Definitions & Conceptual Model

You keep using that word.
I do not think it means what you think it means.
— Inigo Montoya, in The Princess Bride

This chapter provides key definitions and a conceptual model for designing complex systems using the proposed method. It uses cyber-physical systems as examples of complex systems with examples taken from the developing *smart grid* as imagined for a software-enhanced power distribution system for distributing electricity within a residential area.

The *Adaptive Architecture for Systems of Intelligent Systems* (AASIS) framework described in Chapter 6 includes a design process for decomposing complex systems, and provides specific examples of complex systems, some of which are arranged hierarchically. The hierarchical structure motivates complex organizational structures consisting of different groups at various levels and provides a means to evaluate the architecture and framework in terms of how well it manages complexity (as defined by multiple interacting organizations). AASIS was designed to work for additional types of complexity, and these examples are meant to provide an illustration.

Discussing any subject involving relationships between entities that are inherently complicated can be challenging. Agreeing on a vocabulary is a useful first step. In this chapter, terms are described that are used to characterize existing systems, and when discussing algorithms for distributed problem solving. The section begins with general terms and works towards very specific examples of the complex systems used as test cases. Approaches are suggested for decomposing and designing systems along with a method for mapping them into a reusable, flexible agent-based framework for implementation. The hope is that by providing a well-defined conceptual model of terms first, and providing easy access to terms from the table of contents, that the process for designing and implementing systems with multigroup agents may be presented a bit more clearly in the remaining chapters.

The definitions are intended to represent the way the terms are used in this dissertation and when discussing concepts and design aspects associated with the new OBAA^{++} architecture for multigroup agents proposed in Chapter 5 and the AASIS framework proposed in Chapter 6 for implementing complex systems.

This chapter defines basic types of systems in Section 2.1-2.5 including *cyber-physical systems* (CPS) and *power distribution systems* (PDS). It covers *agents*, *multiagent systems* (MAS), and *complex intelligent MAS* in Sections 2.6-2.14. These definitions are used to define *systems of intelligent systems* in Section 2.15. Sections 2.16-2.22 introduce several types of system designs, including *hierarchical* and *holonic*. Finally, examples of application-specific systems used in this research are covered in Sections 2.23-2.26 and a chapter summary is provided in Section 2.27.

2.1 Systems

The definitions begin with a very general, and widely-used concept, a system.

Definition 2.1 (System)**.** A *system* is a set of interacting or interdependent components forming an integrated whole[41,42].

Definition 2.2 (Delineated System)**.** A *delineated system* is a set of interacting or interdependent components forming an integrated whole defined in part by the presence of a *boundary*[41,42].

Definition 2.3 (Environment)**.** That which lies inside the boundary comprises the delineated system. That which lies outside the boundary makes up the system *environment*.

This dissertation refers to many types of systems. An overview of the relationships between the various entities is provided in Figure 2.1. In the Unified Modeling Language (UML) notation used, a $\triangle$ or triangle indicates the entity is a generalization of the related entity, while the diamond indicates aggregation (i.e., an entity with a diamond has one or more of its related entities). Use of a $\Diamond$ or white diamond indicates that the entities exist independently, that is, deleting the aggregated entity does not delete the related entity. The first two concepts, environment and system, were described above. The remaining system types will be described in the sections below, starting with the crucial differentiation between types of systems needed for the decomposition process: the division of physical systems and cyber-systems.

2.2 Physical System

A *physical system*, for the purposes of this work, refers to a system composed of physical items used together for a common purpose. Physical systems may include many types of physical components, including devices, sensors, actuators, connectors, and conduits. Generally, basic control systems such as Supervisory Control and Data Acquisition Systems (SCADA) systems may be considered parts of a physical system.

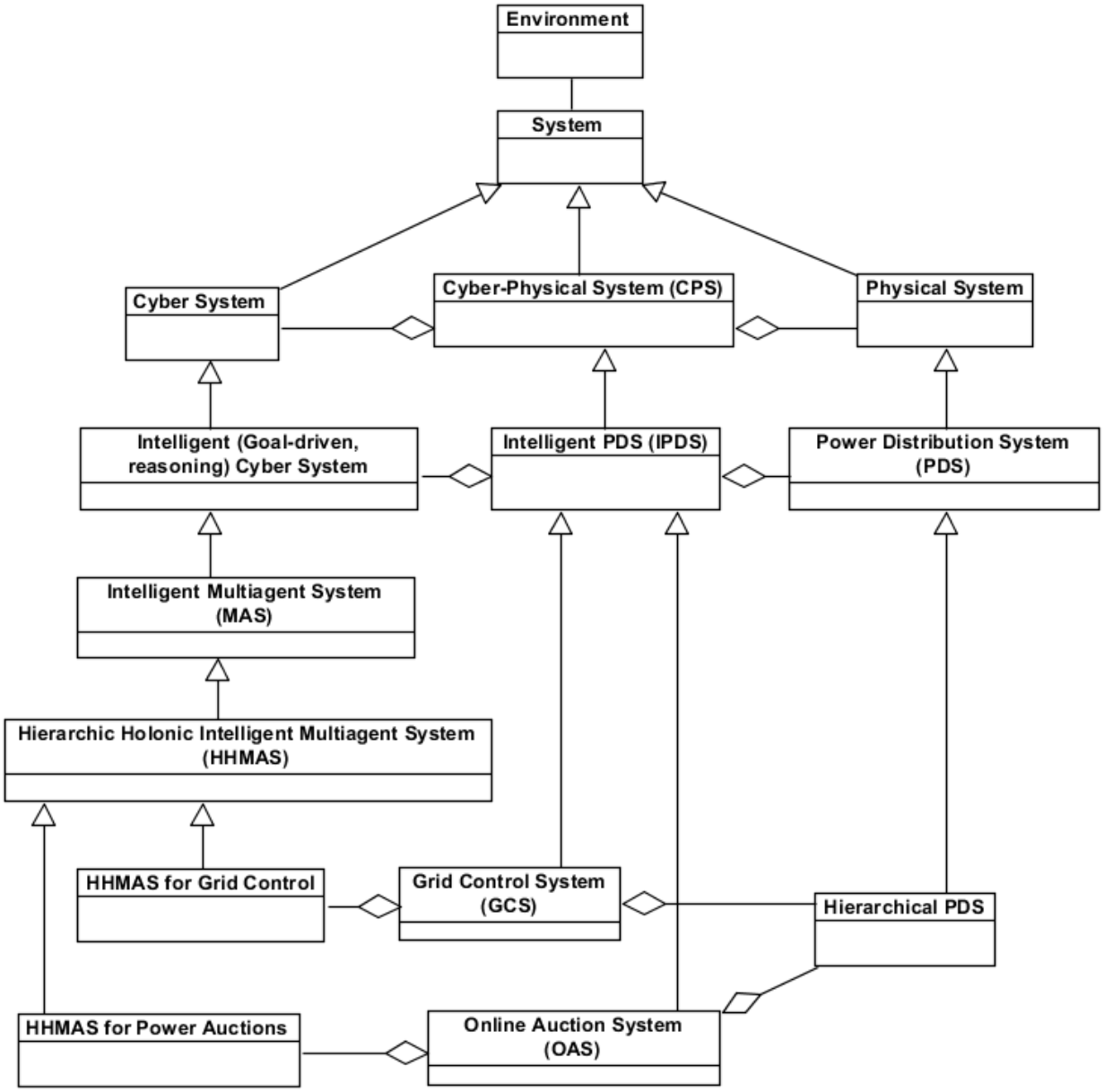


Figure 2.1: *Partial ontology of simple system types displayed an a class diagram.*

2.3 Cyber-System

A *cyber-system* refers to a system composed of interacting software components. For the purposes of this work, a multiagent system is considered to be purely a cyber-system.

2.4 Cyber-Physical System (CPS)

A *cyber-physical system* (CPS) is the composition of a cyber (software) system for computation and communication with an underlying physical system[2]. A CPS can thus be described as shown in Definition 2.4.

Definition 2.4 (CPS)**.**

$$S = C \circ P$$

$$\text{where } S = \text{a cyber-physical system}$$

$$C = \text{a cyber-system}$$

$$P = \text{a physical system}$$

The $\circ$ operator is the *composition operator*. In this work, the composition operator is used to define cyber-physical systems, because they require the *concurrent composition of the computing processes with the physical ones*[2]. That is, the *cyber functions* are applied to the *physical functions*. Order matters. The composition operator is not commutative; in general, $C \circ P \neq P \circ C$.

2.5 Power Distribution System (PDS)

Definition 2.5 (PDS)**.** A *power distribution system* (PDS) is a physical system for carrying electricity, typically encompassing the electrical conduit and associated equipment that runs from a substation to the supported set of distributed end users.

In this paper, several PDS configurations were selected in which the substation connects to a set of branching feeders that in turn connect to a set of neighborhood transformers. Each transformer is connected to 2-6 residential homes that consume electricity, some of which are also capable of producing electricity with solar PV.

2.6 Agents & Multiagent Systems (MAS)

The design of software for a CPS can follow different approaches. One design approach receiving significant research attention for PDS control is that of *multiagent systems* (MAS). There are multiple possible definitions for these terms; this dissertation uses the following.

Definition 2.6 (Agent)**.** An *agent* is an entity that perceives its environment and acts autonomously in accordance with the information gathered[43].

Definition 2.7 (MAS)**.** A *multiagent system* (MAS) is an interacting set of agents solving problems beyond the individual capabilities or knowledge of any single agent in the set[44].

Agents may be software, biological, or some combination of both. As an interacting set of entities, a MAS may be operate within a single executable on a single device (e.g., a workstation), or may run on many cores in the cloud. A MAS may also be distributed, running on many different devices, with each entity running in a different executable in a different location.

2.7 Intelligence

The definition of intelligence remains somewhat non-specific. A survey of conventional definitions may include aspects of *logic, abstract thought, understanding, self-awareness, communication, learning, emotional knowledge, memory, planning, creativity, and problem solving*[45]. To define artificially intelligent systems as those that simulate intelligence, remains correspondingly vague. For the purposes of this work, the following definition, which works for both software and organic entities, is used.

Definition 2.8 (Intelligent)**.** An *intelligent* entity is an entity that *responds* to its environment, and acts *autonomously*, *rationally*, and *proactively* to achieve its *goals*[46].

This definition removes basic reactive control systems from the definition, but includes all goal-driven, reasoning entities, with an ability to get some information about their environment.

2.8 Intelligent System

Definition 2.9 (Intelligent System)**.** An *intelligent system* is a goal-driven, reasoning system of autonomous interacting entities pursuing a common set of goals beyond the individual capabilities or knowledge of any single entity[44] as shown in Figure 2.2.

An intelligent system may include intelligent and non-intelligent (e.g., purely reactive)

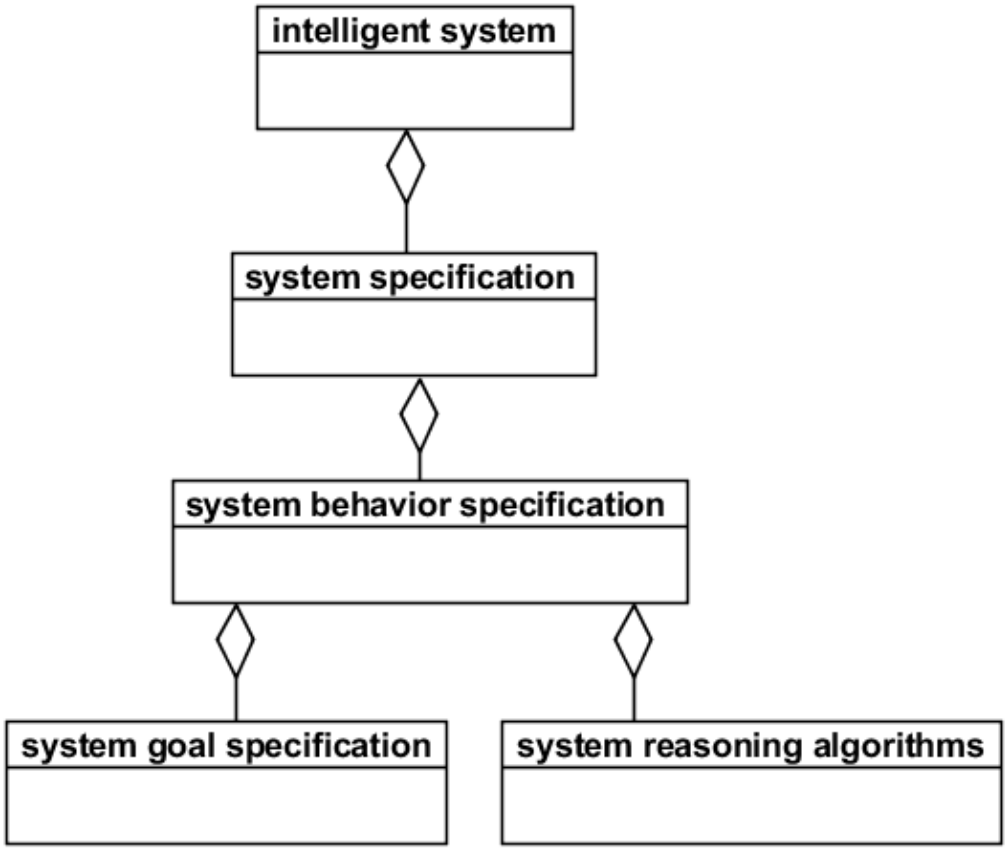


Figure 2.2: *An intelligent system is a goal-driven, reasoning system.*

entities. At least some of the entities must be intelligent, i.e. autonomous, environment-aware, goal-driven, and able to reason.

Hybrid intelligent systems may involve the integration of both biological entities, such as humans, and cyber-systems[14].

2.9 Intelligent MAS

An *intelligent MAS* is a *goal-driven*, *reasoning* MAS. The purpose of an intelligent MAS is defined by the set of goals for the system, just as biological systems are defined by their purpose.

Definition 2.10 (Intelligent MAS)**.** An *intelligent MAS* is a goal-driven, reasoning multi-agent system as shown in Figure 2.3.

An intelligent MAS may be viewed as a society or *organization* of agents (i.e., as a set of agents that interact together to coordinate their behavior and often cooperate to achieve collective goals)[47].

Definition 2.11 (Organization)**.** An *organization* is a set of agents that interact and coordinate their behavior to achieve a common set of goals[47].

In this work, an intelligent system is considered to be organized around goals that focus on a particular problem area. For example, one MAS may be driven by a set of *grid control goals* focused on volt-var control and overvoltage prevention, while another MAS may be driven by a set of goals related to online auctions.

A non-goal-driven MAS is possible, and the general definition of MAS does not specifically require the system to be goal-driven. Generally, outside these formal definitions, the term MAS is used to mean an intelligent, goal-driven MAS.

Organization-based Multiagent Systems (OMAS) provide an effective mechanism for designing large, complex, intelligent MAS[47,48,49]. OMAS provide a clear separation between

agents and the specification of the organization in which they participate, reducing the complexity of the system[49].

This work focuses on intelligent systems, and therefore, an organization is specified in terms of the goals the organization must pursue and the possible structures and behaviors, commonly defined with roles, plans, capabilities, policies, and norms. In organization-based MAS, the agents and entities that staff or play roles in the organization are kept separate from specification of the organization, supporting advanced adaptive behaviors[50]. In this dissertation, an organization specification is defined as shown in Definition 2.11.

Definition 2.12 (Organization Specification)**.** An *organization specification* (OS) defines the set of goals the organization pursues and the structures, behaviors, permissions, requirements, and constraints for achieving those goals.

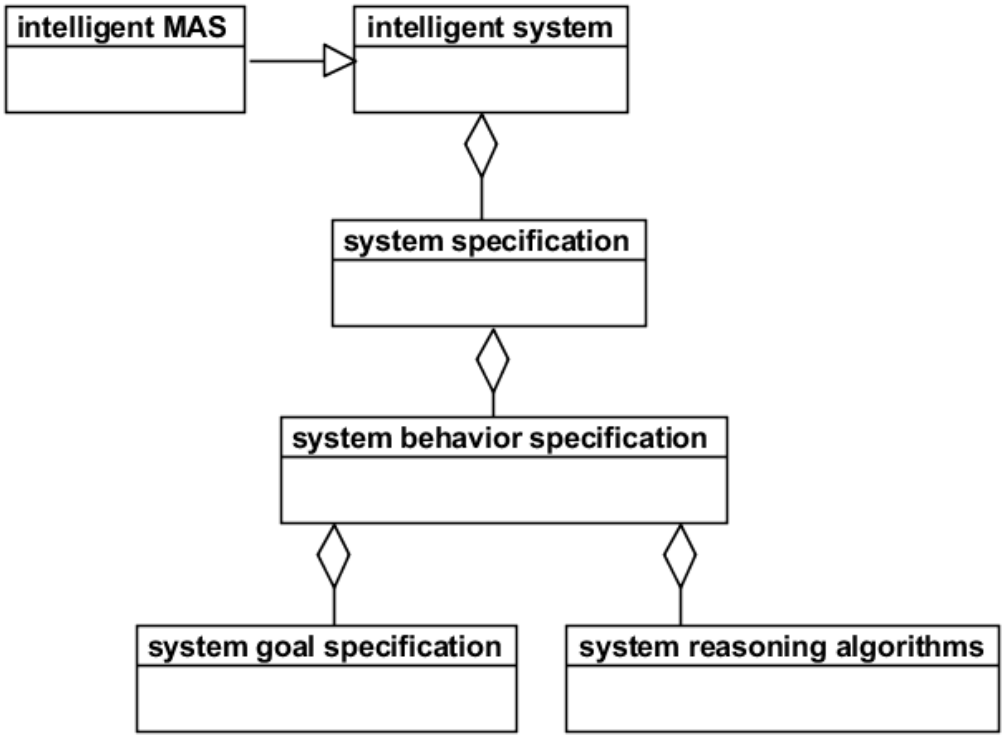


Figure 2.3: *An intelligent MAS is a goal-driven, reasoning MAS.*

a MAS *implements* an OS. An organization is defined by the combination of a MAS and the OS or OSes it implements. Thus, in this dissertation, an organization is dynamic. It defines the state of the goals and agents at a specific point in time during execution of the MAS¿

An organization is *staffed* by agents or other entities playing roles in the organization. Each role typically requires one or more capabilities and may reference one or more plans for playing the role. Models for reasoning about organizations typically include agents in the reasoning model, as described in Section 3.8.

2.10 Complexity

Managing complexity involves design choices; defining the difference between an organization and a system can be challenging. Arguably, single systems and single organizations could be quite complicated. In this dissertation, the term *complex* is used specifically to indicate *more than one of a type*. This term is used in the context of *complex systems* and *complex organizations* as described below.

2.11 Complex System

A *complex system* is an *interacting set of systems*, a.k.a. a system of systems, as described in Definition 2.13.

Definition 2.13 (Complex System)**.** A complex system is a system that includes an interacting or interdependent set of two or more systems.

This recursive relationship is shown in Figure 2.4. Every system operates to achieve its own set of goals, much like the circulatory system or the respiratory system in biology.

A system *component* may be a another system. In such cases, it may be helpful to refer to the larger, containing system as a *supersystem*, and to the contained, component system

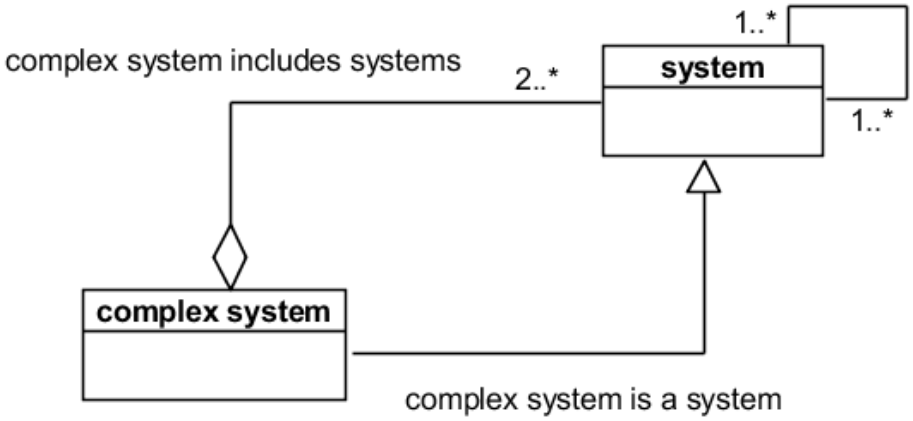


Figure 2.4: *A complex system includes an interacting set of systems and is a system.*

as a *subsystem*. If one or more of the systems are intelligent (i.e., goal-driven), the term *system of intelligent systems* applies.

An example of a complex system is an intelligent power distribution system that incorporates MAS for grid control and a MAS for online auctions. This example was used to evaluate the architecture as described in Section 8.1.

2.12 Complex Organization

To manage complexity, a MAS organization (Definition 2.11) may be decomposed or partitioned into more than one organization[47].

For example, the goals of a hierarchical system may be decomposed into nested organizations, each running similar types of goals with different organizations focusing on different parts of the system. There are many formalized approaches to designing MAS (see Section 3.6), and depending on the selected approach the definition will be different.

Definition 2.14 (Complex Organization)**.** A *complex organization* is an organization that includes an interacting set of organizations and is, itself, an organization as shown in Figure 2.5.

A system may be decomposed or partitioned into multiple organizations. The terms

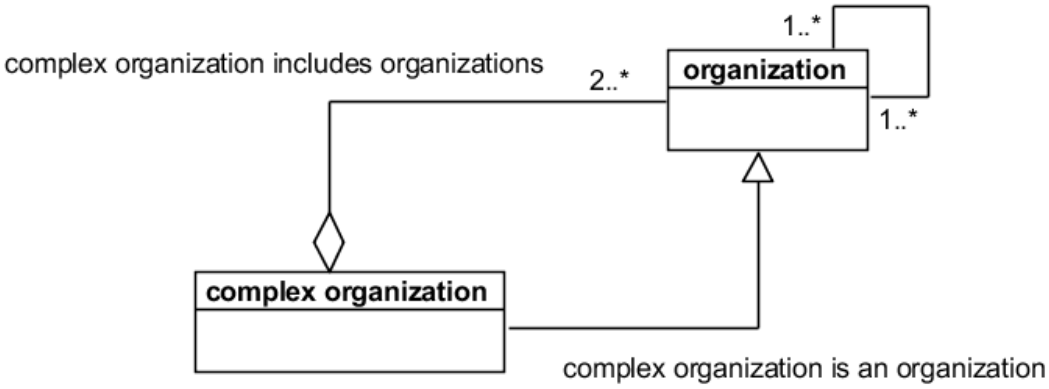


Figure 2.5: *A complex organization includes an interacting set of organizations and is an organization.*

organization and group are generally interchangeable. When dealing with a complex organization, the term *organization* is used refer to the higher, more complex organization in the system. The term *group* or *local group*, is used to refer to the smaller, contributing organization.

Definition 2.15 (Local Group)**.** A *group* or *local group* is a smaller, contributing organization in a complex organization. The term is synonymous with the *organization* entity shown in Figure 2.5.

As an organization (Definition 2.11) each group operates to achieve its own set of goals, called *local goals*. Typically all local goals reflect or contribute to the overall goals of the system[47].

In a hierarchical arrangement, the system goals are high-level goals, which can be decomposed into increasing lower-level goals to be given to lower-level organizations. Complex organizations may be nested hierarchically, which can lead to a recursive set of organizations of organizations within a system. At any point, if several lower-level organizations operate together to form a higher-level organization, the lower level organizations may be considered sub-organizations, or groups, of the higher-level complex organization.

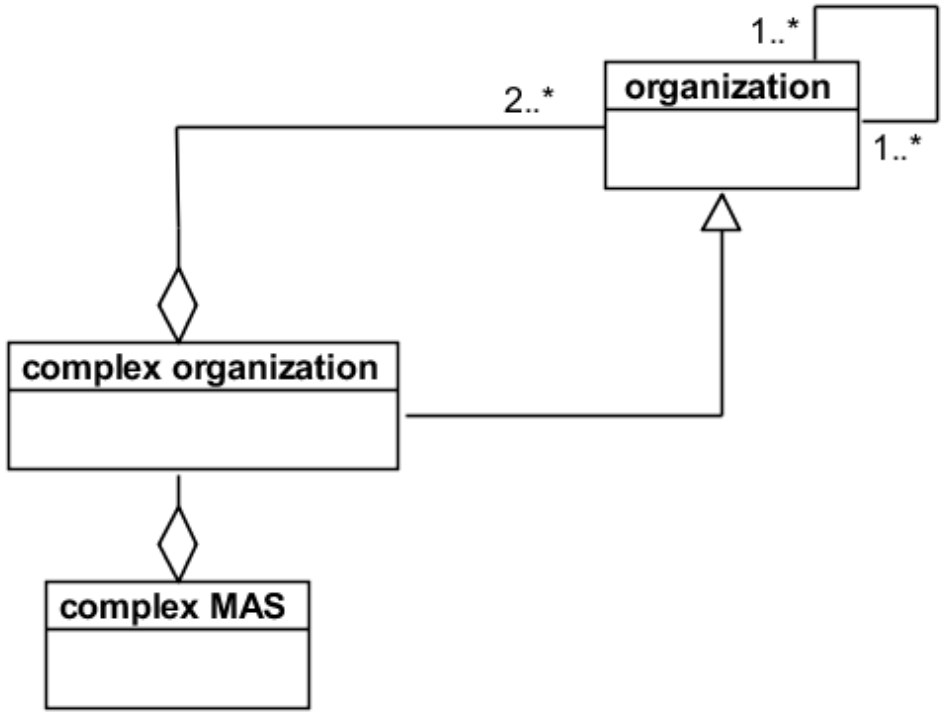


Figure 2.6: *A complex MAS has a complex (multigroup) organization.*

2.13 Complex MAS

A *complex MAS* is a MAS designed with a complex organization. To avoid confusion with the term complex system (a MAS is a system), a term such as *MAS implemented with a complex organizational structure* could be used, but even the abbreviation would be cumbersome, so the term *complex MAS* is used. Additional detail is provided in Chapter 6.

Definition 2.16 (Complex MAS)**.** A *complex MAS* is a multiagent system with a complex organization as shown in Figure 2.6.

It is possible to design complex systems that consist of multiple interacting MAS, each of which is driven by a different set of system goals. If the goals of the MAS are focused on a different problem-solving area, e.g., around online auctions, rather than voltage control, they are considered to be different MAS, even if the systems interact and make use of some of the same underlying devices. The systems may even involve the same underlying agents

as described in Chapter 6.

2.14 Complex Intelligent MAS

A *complex intelligent MAS* is an intelligent MAS designed with a complex organization.

Definition 2.17 (Complex Intelligent MAS)**.** A *complex intelligent MAS* is an intelligent multiagent system with a complex organizational structure as shown in Figure 2.7.

2.15 System of Intelligent Systems

Definition 2.18 (System of Intelligent Systems)**.** A *system of intelligent systems* is a complex, intelligent system as shown in Figure 2.8.

Synonym: complex intelligent system.

2.16 Hierarchy

Software, physical, and biological systems can often be viewed in terms of a *hierarchy*, a way of decomposing systems into levels. The top-level is considered to provided the most comprehensive view, which is progressively sub-divided into lower levels. In this work, the term hierarchy typically refers to a nested, recursive, partially-decomposable hierarchy, in which a top-level entity can be decomposed progressively to the lowest level, but each level is not solely the sum of its parts, as additional components or content may be added at each level. Further, each lower level entity is nested under exactly one higher-level entity, forming a *tree* structure, a constrained type of graph. More complex organizational structures (e.g., those including overlapping structures) are possible, but are not discussed in this work.

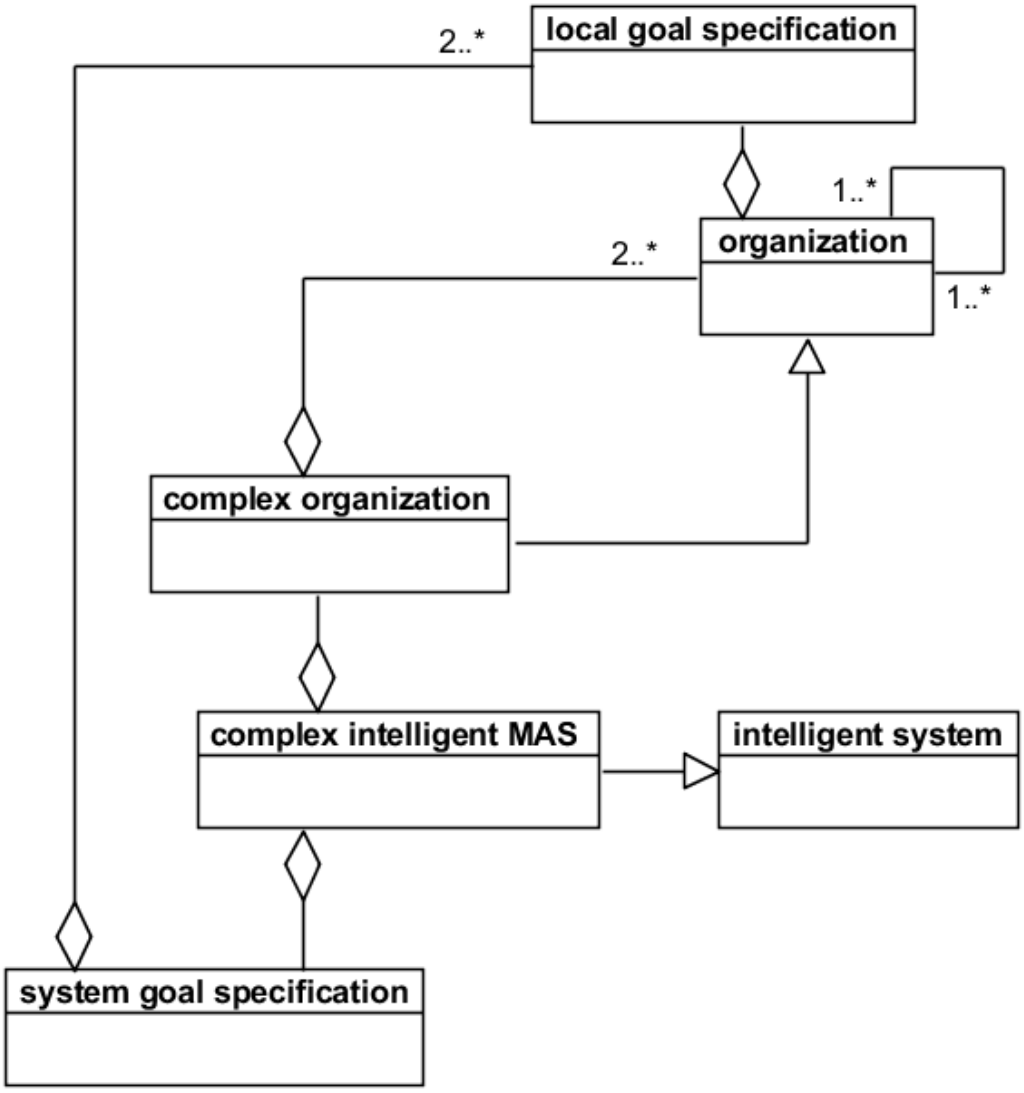


Figure 2.7: *A complex intelligent MAS has multiple groups, each with a local goal model.*

2.17 Hierarchical Systems

A *hierarchical system* is a system composed of interrelated hierarchical subsystems, with some lowest level of non-composed subsystem[51]. The lowest level may be arbitrarily selected to meet the objectives and requirements of system design.

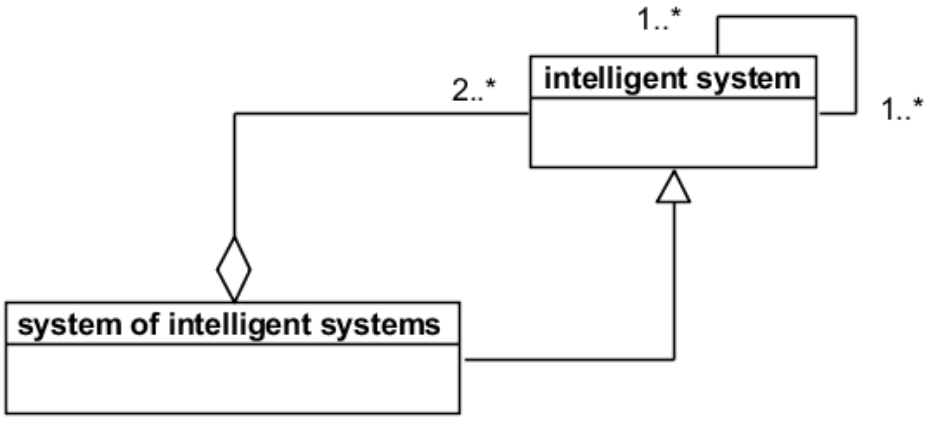


Figure 2.8: *A system of intelligent systems has multiple systems, each with its own system specification and goals.*

2.18 Holon

As part of this research, a *holonic* multiagent system (HMAS) is proposed to facilitate the large-scale integration of rooftop solar PV in residential level, minimize the power losses and deal with scalability issues in distribution system.

Such designs enable a type of recursive control often associated with nested cyber-physical components[24]. The word *holon* comes from *holos*, meaning whole, and *on* meaning parts[52] and refers to an entity that reflects both an entire sub-organization, yet simultaneously acts as a part of a larger organization. Advanced organizational structures such as holons provide ways to organize software into more cleanly separated, testable entities and can help support scalable systems[53].

A holon is defined as shown in Definition 2.19.

Definition 2.19 (Holon)**.** A *holon* is an interacting entity that can be viewed as a whole functional unit and simultaneously, as a part of a larger unit of organization[44,52].

This definition can be applied to both biological and software systems.

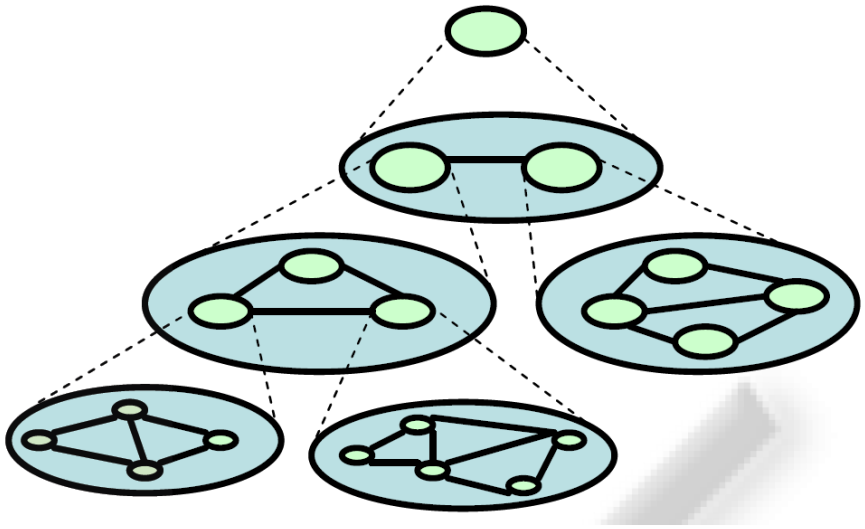


Figure 2.9: *A holarchy is an organization of holons*[8].

2.19 Holarchy

Definition 2.20 (Holarchy)**.** A *holarchy* is an organization of holons as shown in Figure 2.9[8].

Each holon acts as a semi-autonomous subsystem aiming to manage its resources and make decisions autonomously. Each holon functions as both subordinate to control from higher levels and as supra-ordinate, acting in a supervisory capacity to the holons acting as its dependent parts[44,52]. A holarchy therefore includes holons, and is, itself, also a holon. This results in a recursive structure that can be viewed as either a whole unit or as a part of a higher-level holon. At any point, if several lower-level holons operate together to form a higher-level holon, the lower level holons may be considered sub-holons of the higher-level, holon which can be referred to the super holon.

This recursive relationship is shown in Figure 2.10[10].

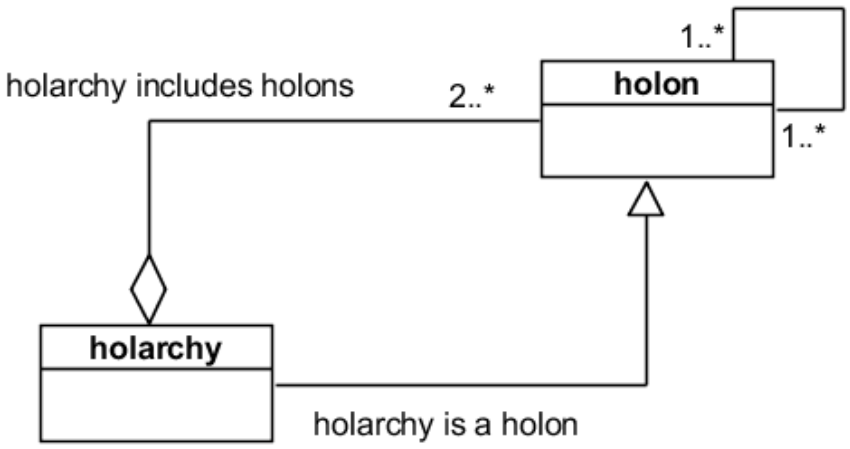


Figure 2.10: *A holarchy includes a set of interacting holons and is a holon*[10].

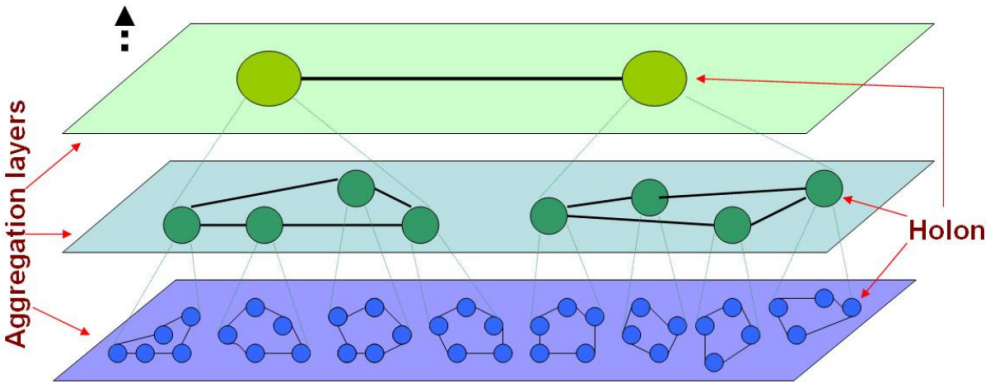


Figure 2.11: *A hierarchical holarchy is a hierarchical organization of holons*[8].

2.20 Hierarchical Holarchy

Definition 2.21 (Hierarchical Holarchy)**.** A *hierarchical holarchy* is a hierarchy of holons as shown in Figure 2.11[8].

In a hierarchical holarchy, each lower-level holon is part of exactly one immediately higher-level holon. The figure was specifically used to illustrate distributed entities in a PDS.

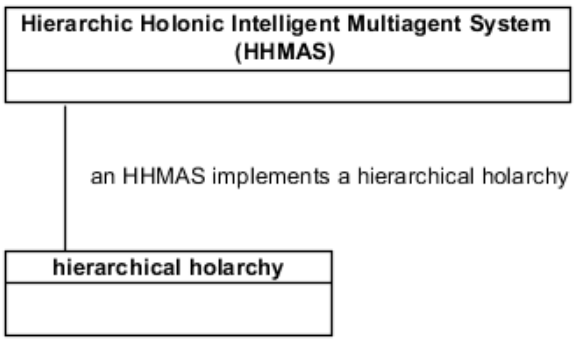


Figure 2.12: *A hierarchic holonic MAS is a multiagent system that implements a hierarchical organization of holons.*

2.21 Holonic MAS

Definition 2.22 (HMAS)**.** A *holonic multiagent system* (HMAS) is a is a complex multiagent system that implements a holarchy.

HMAS can be implemented using multigroup agents by having each holonic agent operate in two groups concurrently: both the one that the agent represents in its entirety and a second where it acts as part of a greater whole.

2.22 Hierarchic Holonic MAS

Definition 2.23 (HHMAS)**.** A *hierarchic holonic MAS* (HHMAS) is a complex multiagent system that implements a hierarchical holarchy as shown in Figure 2.12.

In a HHMAS, each lower-level holon is part of exactly one immediately higher-level holon. The figure was specifically used to illustrate distributed prosumers in a PDS.

An HHMAS is a special type of HMAS, in which the holons are arranged in a nested hierarchy. A top-level holon reflects a complex system. The top-level holon, viewed as a holonic group, includes one or more mid-level holons operating within the top-level group.

These mid-level holons, in turn, may include lower-level organizations represented by holonic agents running on lower levels, i.e., those considered to be further out towards the edges of the distributed system. HHMAS is a useful design approach when the goals and control algorithms in the system are recursive and can be reused between levels. Several HHMAS were implemented during the course of this work.

2.23 Complex CPS

As described in Section 2.11, a *complex system* includes an interacting set of two or more systems. A human, for example, could be considered a complex system, in that each human includes a nervous system, a respiratory system, and other systems each of which is focused on specific *goals*. The systems are interactive, and mutually supportive, yet can be considered separately to assess current functionality. Every cyber-physical system is an example of a complex system, since it is the composition of both a cyber-system and a physical system as shown in Definition 2.4.

A *complex CPS* is defined shown in Definition 2.24.

Definition 2.24 (Complex CPS)**.**

$$S = S_1 \cup S_2 \cup ... \cup S_n$$

$$\text{where } S = \text{a complex cyber-physical system}$$

$$n \in \mathbb{Z}$$

$$n \geq 2$$

When describing complex CPS, the *union operator*, $\cup$ is used. Systems involve a variety of software and physical components and devices which may work together or be included in multiple systems at the same time. The order of processing the supporting systems may not be clearly defined and may change over time. The union operator is commutative;

$S_1 \cup S_2 = S_2 \cup S_1$.

To form a complex CPS, either the software system or the physical system, or both, could be complex. For example, cyber-systems may be connected via messaging technologies, and physical systems may represent networked, distributed assets. Thus, a complex CPS may be formed by either the composition of a complex cyber-system with a single physical system as defined in Definition 2.25 or by the composition of a single cyber-system with a complex physical system as defined in Definition 2.26.

Definition 2.25 (Complex CPS with Simple Physical System)**.**

$$S = S_1 \cup S_2 \cup \ ... \cup S_n$$

$$C = C_1 \cup C_2 \cup \ ... \cup C_n$$

$$S_i = C_i \circ P$$

where S = a complex cyber-physical system

C = a complex cyber system

S_i = a cyber-physical system

C_i = a cyber-system

P = a physical system

$n \in \mathbb{Z}$

$n \geq 2$

Definition 2.26 (Complex CPS with Simple Cyber-System)**.**

$$S = S_1 \cup S_2 \cup \ ... \cup S_n$$

$$P = P_1 \cup P_2 \cup \ ... \cup P_n$$

$$S_i = C \circ P_i$$

where $S =$ a complex cyber-physical system

$P =$ a complex physical system

$S_i =$ a cyber-physical system

$C =$ a cyber system

$P_i =$ a physical system

$n \in \mathbb{Z}$

$n \geq 2$

When describing either complex cyber-systems or complex physical systems the *union operator*, $\cup$ is used. Complex systems involve a variety of elements which may work together and be used by multiple systems at the same time. The order of processing supporting systems may not be clearly defined and may change over time. However, each supporting cyber-system is a composition of the cyber functions on the physical functions (order matters), as discussed with Definition 2.4.

In turn, the design of each contributing system itself may be complex[1,54]. A complex CPS could also involve multiple cyber-systems and multiple physical systems, as may be envisioned for CPS-enabled cities.

2.24 Grid Control System (GCS)

Different examples of CPS that employ complex MAS have been developed to evaluate the architecture and framework and test proposed algorithms. The first is a *grid control system* (GCS), in this case, a CPS for volt-var control in a power distribution system. A GCS is a composition of a complex MAS for grid control with an underlying physical system as shown in Definition 2.27.

Definition 2.27 (GCS)**.**

$$S_G = C_G \circ P_P$$

$$\text{where } S_G = \text{a cyber-physical system}$$

$$C_G = \text{complex MAS for grid control}$$

$$P_P = \text{a power distribution system}$$

2.25 Online Auction System (OAS)

A second example is an *online auction system* (OAS), a CPS for online auctioning of distributed electrical generation between local prosumers in a PDS. A prosumer is an entity that can both *produce* and *consume* electricity.

A OAS is a composition of a complex MAS for online auctions with an underlying physical system as as shown in Definition 2.28.

Definition 2.28 (OAS)**.**

$$S_S = C_S \circ P_P$$

$$\text{where } S_G = \text{a cyber-physical system}$$

$$C_S = \text{complex MAS for online auctions}$$

$$P_P = \text{a power distribution system}$$

2.26 Intelligent Power Distribution System (IPDS)

The examples in this paper propose adding computational and communication capabilities to a PDS to create an intelligent power distribution system (IPDS). An IPDS is a general term for a CPS for a PDS. It can include one or more intelligent cyber systems of any type using the general Definition 2.29.

Definition 2.29 (IPDS)**.**

$$S_I = C \circ P_P$$

$$\text{where } S_I = \text{an intelligent power distribution system}$$

$$C = \text{a cyber system}$$

$$P_P = \text{a power distribution system}$$

This work describes an approach for implementing *complex systems*, and systems of intelligent systems. Since both a GCS and an OAS may integrate a common PDS, the systems may interact, sharing common sensors, and determining desired operation of associated inverters so as to maximize the achievement of both online auction goals and power control goals, as discussed in Section 8.1.

The cyber-system in Definition 2.4 may be complex. Using the recursive definition of

a complex system provided in Section 2.11, a flexible definition for an intelligent power distribution system (IPDS) was developed such that both a GCS and an OAS may be examples of a (simple) IPDS as shown in Figure 2.1. Further, if both the GCS and the OAS are running simultaneously on a common PDS, they create an IPDS with a complex cyber-system.

Thus, the definition of an IPDS can be expanded, and allowed to incorporate the addition of new systems focusing on additional areas as well as shown in Definition 2.30.

Definition 2.30 (Complex IPDS)**.**

$$S_I = \{C_1 \cup C_2. \ . \ . \ \cup C_n\} \circ P_P$$

where S_I = an intelligent power distribution system

C_i = a cyber system

n = the number of interacting cyber systems

P_P = a power distribution system

Systems that share resources, such as real and reactive power, create opportunities for cooperation. Mechanisms for managing goal consistency in complex, cooperative systems are presented in Chapter 6.

2.27 Summary

This chapter describes the terms used in this research. It defines basic types of systems including CPS and PDS. It covers *agents*, MAS, and complex intelligent MAS. It defines *systems of intelligent systems* and several types of system design, including hierarchical and holonic. Finally, examples of application-specific systems used in this research are

introduced. These definitions help lay a conceptual foundation for the background and related work described in Chapter 3.

Chapter 3

Background

To know that we know what we know,
and to know that we do not know what we do not know,
that is true knowledge.
— *Copernicus*

This research touches on several areas of computer science, some of which are relatively new. This chapter provides a brief overview of some of these related areas, both to provide some general background as well as to introduce additional prior work beyond what was mentioned in the introduction. The intent is to provide a descriptive background, somewhat broad without going too deep. More formal definitions of key terms are provided in Chapter 2. Section 3.1 introduces the challenges and potential of distributed artificial intelligence. Sections 3.2 and 3.3 focus on the objectives and motivation for complex, self-adaptive systems. Section 3.4 discusses the unique aspects of cyber-physical systems and Section 3.5 discusses multiagent systems.

This chapter also includes an overview of some of the related work that motivates and enables this research. Section 3.6 looks at some relevant work in the broader field of agent-

oriented software engineering and organization-based systems. Section 3.7 looks at important work in the area of goal-driven systems and Section 3.8 at some of the most successful approaches for modeling organizations and specifically OMACS, which is a crucial foundation of this work. Section 3.9 covers some key work in the area of tool development and specifically, prior work on agentTool3 which was used to create our behavior specifications. Section 3.10 provides an overview of prior work in engineering *organization-based* agent systems while Section 3.11 and 3.12 introduce prior work related to modeling goals and managing goal consistency, respectively and Section 3.13 introduces some research on complex system that offer especially helpful mechanisms for presenting and discussing the selected approach. Section 3.14 and Section 3.15 cover prior work enabling our approach to multigroup systems and the implementation of the voltage control and online auctioning implementation examples, respectively. Section 3.16 discusses recent research specifically in the area of HMAS for PDS, a key part of the motivation for this work, while Section 3.17 looks at additional interest in broader sectors and Section 3.18 covers related work with multigroup control structures and multigroup agents. Section 3.19 provides a chapter summary.

3.1 Distributed Artificial Intelligence (DAI)

The term *artificial intelligence* (AI) has been around at least since 1955, when a team of researchers proposed a research project on the *science and engineering of making intelligent machines*[55]. Since then, the field has grown tremendously, resulting in advances such as Watson, the Jeopardy-winning computer[56], Roomba, the domestic vacuum-cleaning robot[57], and the autonomous soccer-playing robotic teams competing in annual RoboCup competitions[58,59]. The subject engages not only researchers, but the public in general. When Stanford offered their free online course on Artificial Intelligence, over 58,000 of us from all over the world signed up[60,61].

The goal of using AI to fully emulate human intelligence remains beyond our abilities, but the application of AI principles to specific sub-problems has been fruitful[62]. A large and valuable sub-area is that of DAI, which Bond and Gasser called *the subfield of AI concerned with coordinated, concurrent action and problem-solving*[63].

In 1999, Ferber described the goal of creating *distributed artifacts capable of accomplishing complex tasks through cooperation and interaction*[43] and Weiss defined DAI as *the study, construction, and application of multiagent systems, that is, systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks*[64].

This definition, with its focus on a *system of agents* working towards a *common set of goals* is part of what I consider to be a critical aspect of intelligent systems (Definition 2.9) and intelligent MAS (Definition 2.10).

3.2 Complex Adaptive Systems

The study of *complex adaptive systems* in general uses the word *complex* in a somewhat different way than used previously in this dissertation. Complex in this case, may include systems with large numbers of elements, dynamic networks of interactions, or intricate interconnectivity, rather than simply consisting of more than one of a type[65].

Adaptive refers to the internally-driven change, adaptation, or mutation of a system in response to changes in the system environment and has been motivated by examples from nature and biology, complexity science, and other disciplines, including AI, cognitive science, and game theory[66,67,68]. Current examples include the new 3D-printed, self-charging, cooperative BionicANTs from Festo[69,70].

The adaptive response is often motivated by a specified set of goals[71]. Research into complex adaptive systems applies to social organizations, biological organisms, ecological communities, and DAI.

In this work, the term *complex system* (Definition 2.13) is used to refer specifically to

a *system of systems*. The convention is followed that a *single system* tends to be united under a *single set of goals*, much like the circulatory system or the respiratory system in biology. Although formed of many different components, each organ system performs a specific purpose.

A human, on the other hand, consists of many systems – it is a system of systems – and thus warrants the term *complex*. This work uses the terms *complex MAS* and *system of intelligent systems* to refer to entities composed of smaller entities, each of which pursues its own goal-driven objectives.

This research deals specifically with the features needed to support complex intelligent systems and to manage the inherent complexity through a variety of abstractions and design decisions that enable the separation of concerns in a reusable and flexible manner.

Work in artificially-intelligent complex adaptive systems has evolved from its foundations in multiagent systems, and the application of autonomous entities and organizations that can be programmed to work towards a variety of different, possibly interrelated, or even potentially-conflicting, goals.

By distributing the intelligence among the devices, systems can react more quickly to local events, and can employ on-line learning to enhance their responses. The benefits come with significant complexity, and the management of this complexity is a key part of what motivated this research.

3.3 Self-Adaptive Systems

Self-adaptive software systems are those capable of evaluating their own behavior in the context of a changing environment and modifying their behavior when they determine they are not accomplishing their objectives as well as they could[72,73,74]. The run-time adaption is made possible by supporting *self-* properties including such aspects as *self-healing*, *self-configuring*, and *self-optimizing* that may be considered hierarchically[72,75,76]:

1. **Highest**: self-adaptiveness, self-control, self-managing, self-governing, self-maintenance, self-evaluating, and self-organizing.

2. **Major aspects**: self-configuring, self-diagnosing (identifying anomalies, faults, failures), self-repair (recovering from failures), self-healing (discovering, diagnosing, reacting to failures, proactively avoiding failures), self-optimizing/tuning/adjusting (in terms of response time, throughput, utilization, and/or workload), and self-protecting, both reactive and proactive.

3. **Primitive aspects**: self-awareness (aware of its own states), self-monitoring, self-situated, context-awareness (aware of its environment).

Adaptive, resilient, self-managing systems may also be referred to as *autonomic* computing systems, where *new components integrate as effortlessly as a new cell establishes itself in the human body*[75].

3.4 Cyber-Physical Systems (CPS)

CPS is a relatively new term from the field of *embedded systems*, indicating a targeted computer system where software and hardware work closely together to enable functionality in dedicated devices as described in Section 2.4.

Devices range from smart phones and smart watches, to vehicles and traffic lights, to smart meters and smart inverters for sensing and managing electric power[1,21]. CPS may incorporate potentially massive numbers of connected devices working together. Rather than transporting massive amounts of information to a centralized source, intelligent algorithms that incorporate sensing, learning, and taking appropriate actions can be distributed among the various devices[77]. CPS devices can communicate and be programmed to pursue local optimums that can be increasingly combined into more centrally-optimum solutions[54].

Physical aspects of a CPS may be organized in different ways and may be either stationary or mobile. An example of a stationary CPS that tends to be arranged hierarchically is a residential electric PDS. In a traditional PDS, electricity generally flows from a single substation, down and out through a distribution network involving high-voltage, 3-phase feeder lines, which split into single-phase lateral lines, then down into neighborhood transformers, which further drop the voltage to a level suitable for supplying end users in homes. This hierarchical structure is more like a tree. Connected traffic lights may be arranged in more complex structures, such as graphs and grids. In some distributed sensor CPS, the physical sensors may be distributed randomly. In some cases, the such as CPS systems in transportation, the physical devices may be mobile.

Enabling advanced CPS systems involves subfields such as DAI, MAS, and adaptive systems. CPS design may incorporate aspects of control theory, game theory, machine learning, and distributed problem solving[27].

3.5 Multiagent Systems

Software systems can be designed in many ways. One option for designing a system is to create intelligent agents and multiagent systems as defined in Section 2.6. Engineers may design MAS in many different ways to best meet the needs of the system and to optimize the communication and processing required. A good overview of the problem domains suited to MAS is shown in Figure 3.1[11].

Some MAS are implemented as a single group of agents. Complex MAS many include many groups. Agents may be part of only one group, or in some cases, for example, when implementing control holons, agents may need to act in multiple groups concurrently.

Description	Usefulness	Performance	Robustness	Adaptability	Scalability	Cost
Client-server	Centralized, reliable, high-security environments	Fast; response times increase gradually as more requests are made	System fails when server goes down; solve by increasing number of servers	Difficult to adapt to changing circumstances or new business requirements	Congestion risk when adding more users; solve by increasing number of servers	Higher initial capital investment, higher maintenance costs
Service-oriented	Loosely coupled systems with business- and technology-domain alignments	Dynamic service composition and description parsing introduce performance overhead	Depends on the quality attributes of the service composition infrastructure to deal with failures	Can adapt at the service level using dynamic service discovery	Stateful services require exchange of service (meta)data, which increases coupling and reduces scalability	Cost effective as a result of reusability, composability, and standardization
Multiagent system	Inherently distributed, locally autonomous, highly dynamic environments	Efficient adaptation to local changes; lack of global information can result in myopic decisions	Goal-directed mechanism handles failures; if failure occurs, it will have only local impact	Goal-directed, context-sensitive process selection extends the benefits of loose coupling and adaptability to individual processes and workflows	Depending on the connectivity, communication channel can become a bottleneck when adding nodes; solve by increasing bandwidth	Economical with regard to required processing power; poor integration of agent-based design tools with common engineering tools

Figure 3.1: *Technology comparison of multiagent systems with client-server and service-oriented system approaches*[11].

3.6 Organization-based AOSE

A variety of approaches to the successful design and implementation of intelligent, agent-based systems have been proposed. These engineering methodologies may come with a recommended set of associated software engineering practices, models, and/or tools. Some examples of methodologies reviewed in this research area include:

- Adelfe (b. 2003)[78]
- AGR (b. 2003) agent-group-role based[47]
- ASPECS (b. 2010)[18]
- Gaia (b. 2003)[48]
- Gormas (b. 2011)[79]
- INGENIAS (b. 2005)[80]
- MOISE+ (b. 2002)[81]

- O-MaSE (b. 2004 MaSE)[12] (also OMACS, GMoDS, and OBAA)
- OMNI (b. 2004)[82]
- OperA (b. 2003)[83] (with associated model checking tool, OperettA[84])
- PASSI (b. 2005)[85]
- Prometheus (b. 2005)[86]
- ROMAS (b. 2012)[87]
- SODA (b. 2006)[88]
- Tropos (b. 2004)[89] (associated with JACK, GRL)

In some cases, the approaches may be quite similar, and in others, the different methodologies offer significantly different approaches[90] See also a discussion of some of the associated organizational models in Section 3.8.

O-MaSE is an organization-based, role-centered process framework that consists of three main components: a metamodel, method fragments, and guidelines[12]. The metamodel describes system components as shown in Figure 3.2. Method fragments define engineering roles, their activities, and the resulting work products, such as the goal models, role models, and plan diagrams that define the system. Each aspect of O-MaSE is supported by agentTool3 modeling tools, which support method creation and maintenance, model creation and verification, and code generation and maintenance.

O-MaSE was selected as the engineering process for its organizational focus, the executable goal model, the available tools, and its ease of use.

During the course of this work, a new O-MaSE-compliant process was developed and employed to assist with implementation of complex systems involving a variety of supporting organizations. The new process, called the adaptive organization-based multiagent system

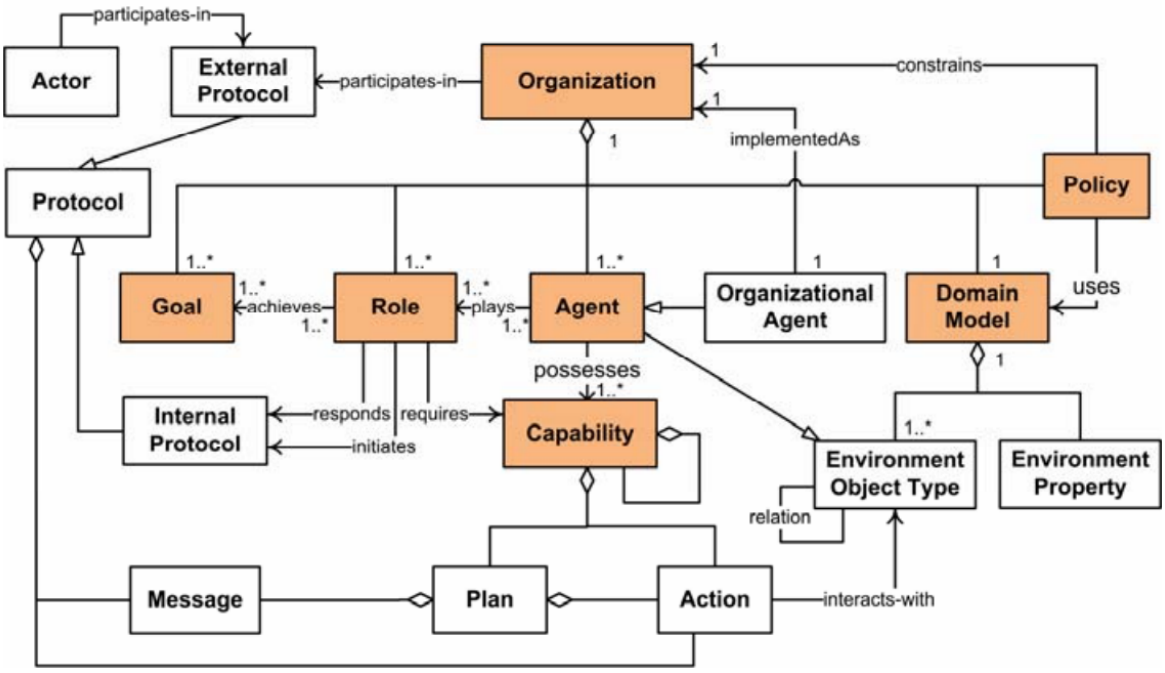


Figure 3.2: *O-MaSE meta model for engineering agent systems. OMACS components are highlighted*[12].

engineering (AO-MaSE) process has been expanded and revised to guide the development of the multiple organizations required[38,39].

Linnenberg et al. used the O-MaSE methodology and agentTool3 to develop DEMAPOS (DEcentralized MArket Based POwer Control System)[33] for power trading; their convention of combining entities capable of producing and/or consuming electricity into the notion of *prosumer* agents has been used in this dissertation as well.

3.7 Specifying Behavior

O-MaSE begins with specifying the goals for the MAS organization and offers the Goal Model for Dynamic Systems (GMoDS), which includes an executable goal model that will instantiate goals during runtime[91]. System goals are defined early in the O-MaSE process as a single *goal specification tree*. The overall *top goal* for the MAS is progressively decom-

posed and the *leaf goals* – those without any goal children – are those that get assigned to participating agents.

GMoDS goals can be customized by providing an optional set of goal parameters to the tree which can be cascaded down to the assignable goals. During runtime, new goals can be issued in response to various events (such as a preceding goal being successfully completed), or the parameters may be modified for existing goal assignments[14].

3.8 Modeling Organizations

Following from our selection of O-MaSE as our foundation for engineering systems and GMoDS for dynamic goal management, this work selected the Organization Model for Adaptive Complex Systems (OMACS) to represent the organization information to the agents[13].

OMACS defines an organization as a tuple as shown in Definition 3.1.

Definition 3.1 (OMACS organization)**.**

$$O = < G, R, A, C, \Theta, P, \Sigma, oaf, achieves, requires, possesses >$$

where:

- G - goals of the organization
- R - set of roles
- A - set of agents
- C - set of capabilities
- Θ - relation over G x R x A defining the current set of agent/role/goal assignments
- P - set of constraints on Θ
- Σ - domain model used to specify environment objects and relationships

- oaf - function P (G x R x A) $\rightarrow$ [0..∞] defining quality of a proposed assignment set
- achieves - function G x R $\rightarrow$ [0..1] defining how effective the behavior defined by the role could be in the pursuit of the specified goal
- requires - function R $\rightarrow$ P (C) defining the set of capabilities required to play a role possesses function A x C $\rightarrow$ [0..1] defining the quality of an agent's capability

OMACS also includes two additional derived functions to help compute potential assignment values: capable and potential.

- capable - function A x R $\rightarrow$ [0..1] defining how well an agent can play a role (computed based on requires and possesses)
- potential - function A x R x G $\rightarrow$ [0..1] defining how well an agent can play a role to achieve a goal (computed based on capable and achieves)

The OMACS model is shown in Figure 3.3.

A corresponding organization model for BDI-based JACK agent systems provides the underlying model for some of the most closely-related research[92,93].

Other methodologies include alternate ways of managing organizational structures and a comparison of several have been conducted[94].

Organizations are more fully implemented in O-MaSE and are more central in the model than in some of the alternatives. O-MaSE does not use norms or contracts, but provides *policies* which can be used to describe rules for how organizations must, or should if possible, behave[95,96]. OMACS' organization-based approach provides a key foundation for the design of the multigroup agents provided in this dissertation.

3.9 Frameworks and Tools

Another active research area that supports the development of complex systems includes agent and MAS-related modeling tools. Some of the modeling tools and approaches reviewed

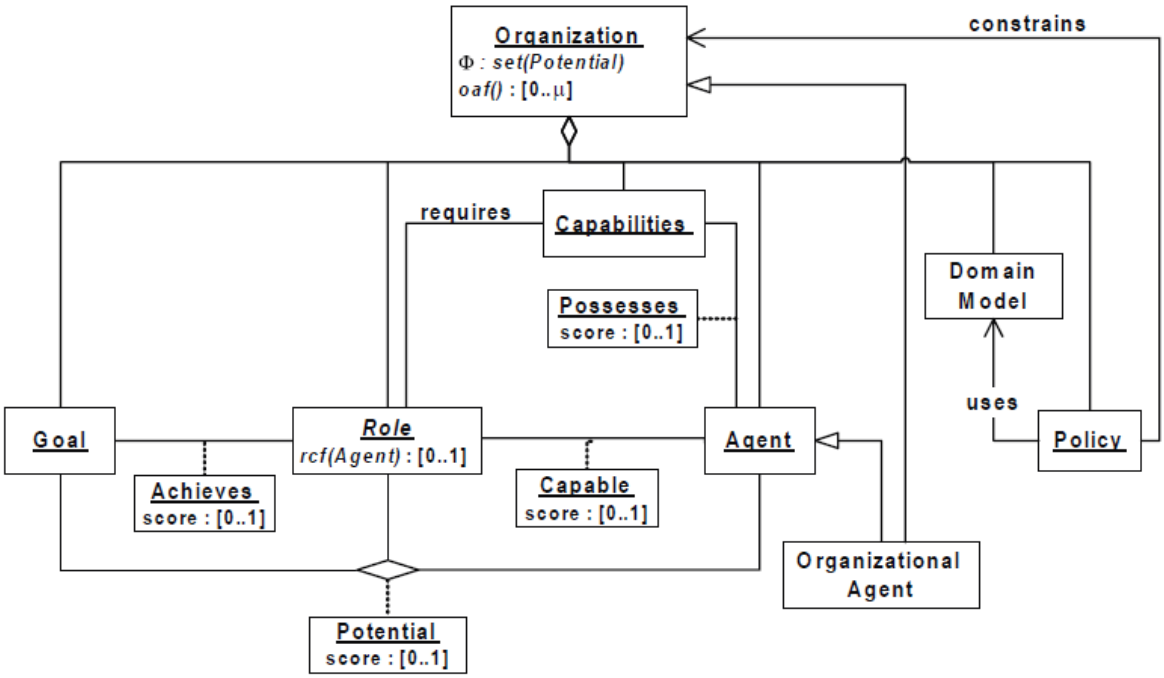


Figure 3.3: *OMACS model*[13].

include the following:

- Agent Development Kit. By Tryllian, Java, services-based[97].
- AgentBuilder for BDI agents[98].
- agentTool3. Used with O-MaSE, OMACS, GMoDS[99].
- Cougaar. Cognitive Agent Architecture from DARPA, focused on scalability, open source since 2004, cougaar.org, uses plugins for domain-specific capabilities[100].
- GIMT. For BDI agents[101].
- Grasshopper. Since 1991, FIPA, MASIF compliant, ACL messaging, shared thread pool[100].
- Ingenias IDE. Since 2005[80]
- Jack. Java Intelligent Agent Framework[102].

- JADE. Java Agent Development Framework, for Belief-Desire-Intention (BDI) agents, FIPA compliant, open-source since 2003, ACL/XML communications, each agent has own thread[103].
- Jason. Based on Agent Speak programming language for Belief-Desire-Intention (BDI) agents[104].
- MadKit.
- MASDK. Multi-Agent System Development Kit[105].
- PASSI. Since 2002[106].
- SkeletonAgent.
- SPARK agent framework[107].
- Tropos Modeling. Since 2002[108].
- Zeus. Since 1999. Agent-building toolkit for BDI agents, Java, FIPA compliant[109].

The 2010 review provides an overview and comparison of agentTool, Cougaar, Jack, JADE, and Jason[110], while earlier reviews include additional platforms, as well as earlier, inactive platforms[111,97].

The O-MaSE methodology works with agentTool3. *Goal specifications* (as shown in Figure 2.2 and Figure 2.3) can be developed graphically, in the Eclipse-based integrated development environment with the agentTool3 plugin. agentTool3 also supports the graphical implementation of role models that relate a role to the goal it achieves, and a required set of capabilities that an agent must have to play the role. This allows organizational behavior to be specified in a loosely-coupled manner, independent from the implementation of specific agents. A Google code project has been created that converts agentTool3 models to UML drawings in Violet[112].

3.10 Organization-based Agents

An agent architecture was designed to support OMACS called the Organization-based Agent Architecture (OBAA)[14]. As shown in Figure 3.4, an OBAA agent consists of two basic components, a general purpose control component (CC) and an application specific execution component (EC). The CC contains the *adaptive behavior logic* related to organizational assignments and is generally domain-independent. The EC contains the *application-specific* part of the agent in terms of roles and capabilities. The EC receives assignments, executes application-specific plans, and notifies the CC when events occur.

The CC has four basic components: Goal Reasoning, Organization Model, Reorganization Algorithm, and the overall Reasoning Algorithm (RA). The RA ties the other three components together. First, the RA gets an event from either another agent or its local EC. This event is passed to the Goal Reasoning component, which updates the organization goals and returns the set of active goals. The RA then stores the active goals in the Organization Model and removes any assignments related to the goals deleted from the active goal set. If the RA decides a total reorganization is required, the Reorganization Algorithm is called to compute a new set of assignments. If the RA calls for a partial reorganization, the Reorganization Algorithm only computes assignments for new goals using using the available agents. The new/updated set of assignments is stored in the Organization Model and the updates are sent to either other agents and to the local EC.

The EC receives assignments, executes application-specific plans and notifies the CC when events take place. The *Agent Control Algorithm* (ACA) takes assignment updates from the CC and provides the ability to select and initiate the execution of assigned roles. The ACA also transmits application specific events back to the CC when they are generated by executing roles. Each *role* in the EC provides an algorithm or plan for carrying out its role while *capabilities* provide reusable software functionality (e.g., communication) or a software interface to hardware-based sensors or actuators.

One of the main design goals of the OBAA architecture was to enable as much reuse

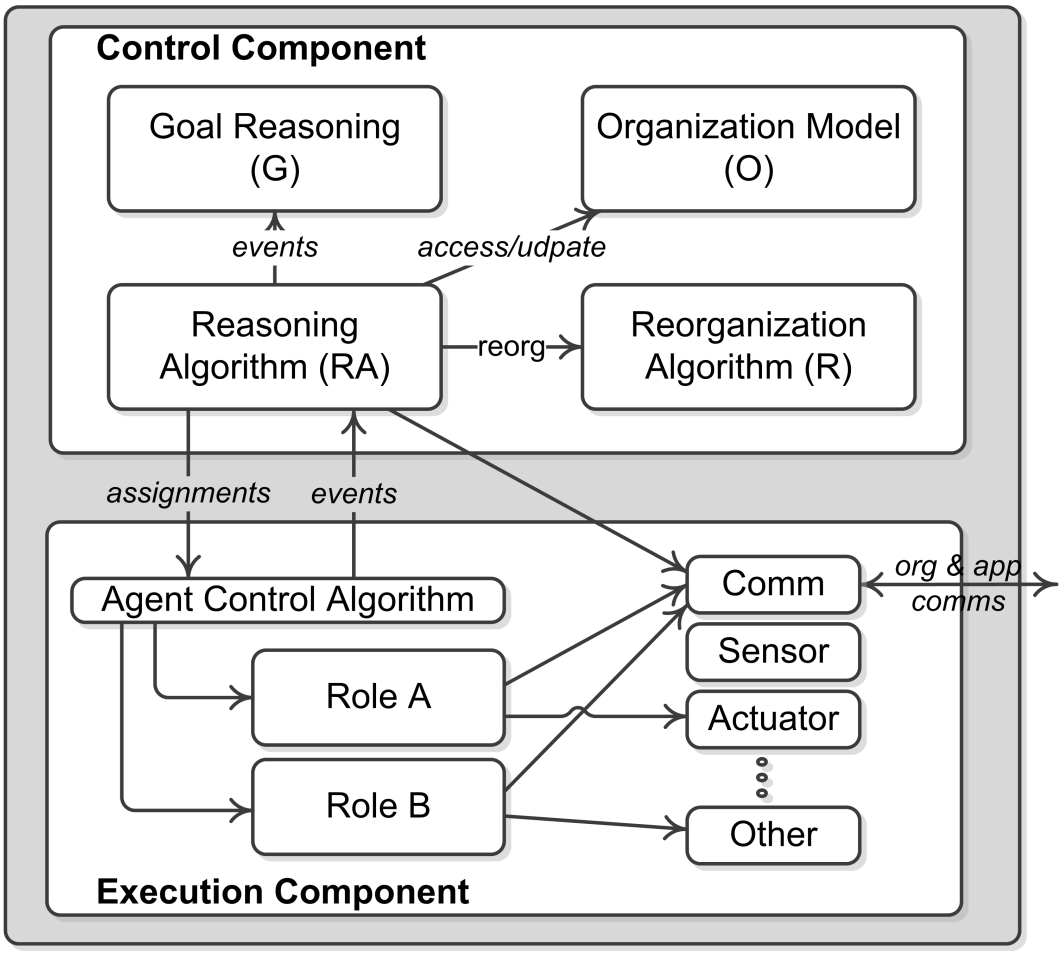


Figure 3.4: *Organization-Based Agent Architecture (OBAA)*[14].

as possible. Since in an OMACS-based system, much of the reasoning has standard forms (e.g., goal reasoning, reorganization, etc.), much of the code in the CC is reusable. Knowledge about the OMACS entities can be inserted into standard implementations of both the Goal Reasoning component and Organization Model component, while a set of standard Reorganization Algorithms are being developed for plug-and-play compatibility; these algorithms include centralized as well as distributed versions. While much of the EC is application-specific, the ACA can be reused and use of standard capabilities can make the job of developing the application specific roles much simpler.

3.11 Modeling Goals

Goals in agent-based systems are crucial. So crucial that it has been suggested that the true essence of the design paradigm of MAS is *not* that of *agent-based* design and implementation, but that of *goal-driven* design and implementation[93].

Some conventions for modeling goals have been developed suggest using a rounded rectangle for goals and a cloud shape for *soft goals*[113]. Soft goals are defined as: *goals that do not have a clear satisfiability definition, used in a number of methodologies, both agent-oriented and non-agent-oriented, for modeling non-functional requirements such as security, usability, flexibility.*[113]. While the selected O-MaSE framework and the associated agentTool3 models follow the recommended convention for goals, neither uses soft goals for describing non-functional requirements.

3.12 Managing Consistency

Formal semantics for declarative goal information have been used to reason about goals in order to detect and resolve conflicts[114]. Tropos and the *Goal-Oriented Requirement Language* (GRL) include the integration of scenario notation. GRL introduces modeling symbols for satisfaction levels and contribution types that address various degrees of the satisfaction of

goals (rather than all or none), as well as partial contributions and degrees of conflict[115]. This work motivated our some of our objectives around managing goal consistency, which were implemented using conflict detection and resource management mechanisms, see Section 6.13.2 and 6.13.4.

Prior work also suggested that our approach to managing goal consistency should include assessments of both *logical consistency* and *resource consistency*. Logical conflicts result when an agent is given goals A and B and either A $\Rightarrow \neg$ B, or B $\Rightarrow \neg$ A. In unparameterized goals, detecting conflicts may be difficult. For example, given two goals, where g_1 =*stop* and g_2 =*go*, the agent, unable to understand natural language processing, would need a way to determine these goals are mutually exclusive. However, the OMACS model provides for parameterized goals which can be used to make the the reasoning process easier. Given two goals, g_1 = *(go, 1 fps)* and g_2 = *(go, 0 fps)* (i.e., *stop*), it may be easier to build algorithms to detect conflicts.

Adding reasoning based on resource constraints would require adding new features to the OMACS model. In prior research, one resource conflict approach defines a resource requirement as a tuple of a specific resource type, t, and an amount, n[116]. The set of resource types is T. The resource set, R, is a set of resource requirements. Resource sets are combined where possible. An example is (energy; 20) and (energy; 30) becomes (energy, 50). Any resource type not included can be assumed to have a number of 0. Resources are either renewable or consumable. Resource sets are either necessary or possible and a resource summary is considered a tuple of these two sets, with the necessary set being a subset of the possible set.

An algorithm for adding a new set of goals was proposed to help detect and avoid resource conflicts in intelligent agents[116], followed by a suggested approach to detecting and avoiding goal interference based on protecting preconditions and invariants[92].

Requirements engineering and goal-driven design are related and additional work with resolving conflicts in cooperative design systems was helpful[117] along with continuing efforts

in goal modeling, operations, goal life-cycles, and semantics for Belief-Desire-Intention (BDI) Agents[118,119].

Although OBAA agents do not use a BDI approach for determining goal assignments, the *CAN* language developed for describing plan bodies[118] provided interesting options for reasoning about goal consistency from our goal names. An external method for detecting consistency between assigned goals (Section 6.13.2) is proposed in Section 6.13, but evaluating whether a similar application of the CAN language for reasoning in multigroup agents is recommended in Section 9.4, Future Work.

The *Persuasive ARgument for Multiple Agents (PARMA) Action Persuasion Protocol* has been suggested for BDI agents[120]. In this approach, BDI agents discuss proposed actions to achieve goals that have been assessed with *value indicators*. The idea of value indicators motivated the proposed conflict resolution approach to not only include resource conflicts, which in a sense reflect costs and requirements, but to also incorporate an assessment of value. Implementing mechanisms to provide an assessment of value in non-BDI agents allows community welfare as well as higher-ranking authority to influence conflict resolution.

Understanding *policies* is be helpful when managing goal conflicts. Policies specify *organizational rules*. Different types of policies include assignment policies, behavioral policies, and reorganization policies[13]. Each policy may be implemented as either a *law policy* or a *guidance policy*[121]. Law policies must be followed, while guidance policies are followed whenever possible and can provide a basis for managing conflicts[122].

These useful features of O-MaSE organizations contributed to architecture and our new agents are able to use policies and other organizational rules, limits, and constraints, to assist with their advanced *self-control* process of monitoring new goals, detecting potential conflicts, and reasoning about resolution approaches (see Section 4).

Research into conflict resolution in cooperative systems comes from fields including computer science, artificial intelligence, and social sciences[117]. Klein describes several primary *conflict resolution approaches* in computational systems:

1. Development-Time. This approach assumes that designers have the time, resources, and ability to ensure that a system is designed such that any possible conflicts are resolved during compilation.

2. Knowledge-Poor, Run-Time. This approach avoids the exhaustive effort required for the first option, and allows agents to detection, asset, and resolve conflicts as they occur during run-time. Examples given include backtracking and constraint-relaxation. These approaches were found to take little advantage of *conflict resolution expertise* while being encumbered with *restrictive formalisms*.

3. General. This approach, says Klein, comes the closest to *providing conflict resolution expertise with first-class status*. However, the paper suggests that none of the implemented systems using the general approach, nor any of those proposed, offer a *comprehensive theory of conflict resolution*.

Ring and Van De Ven lay out a typology of governance structures based on risk and trust as shown in shown in Figure 3.5[15]. Both markets and hierarchies have relatively low needs to establish trust (the top row) compared to recurrent contracts or relational contracts (bottom row). Markets, for example, do not require an agent to depend on a single trustworthy partner – they are free to participate or not – in the market as desired and time frames are generally discrete without undue time pressures. When time is crucial and economic efficiency is the objective, top-down hierarchical governance can efficiently help resolve conflicts and ensure effective operation[123].

The focus of this investigation was on the upper right quadrant, as generally our multigroup systems are holonic and hierarchical. When there is a high possible risk (right column), hierarchical control can be employed so that agents are not required to depend on *establishing trust*; a hierarchical relationship is authoritative and is trusted by definition[124]. Therefore at least for now, a dependency on policies, norms, and contracts to ensure collectively beneficial behavior was not needed. Testing the agents in multigroup organizations

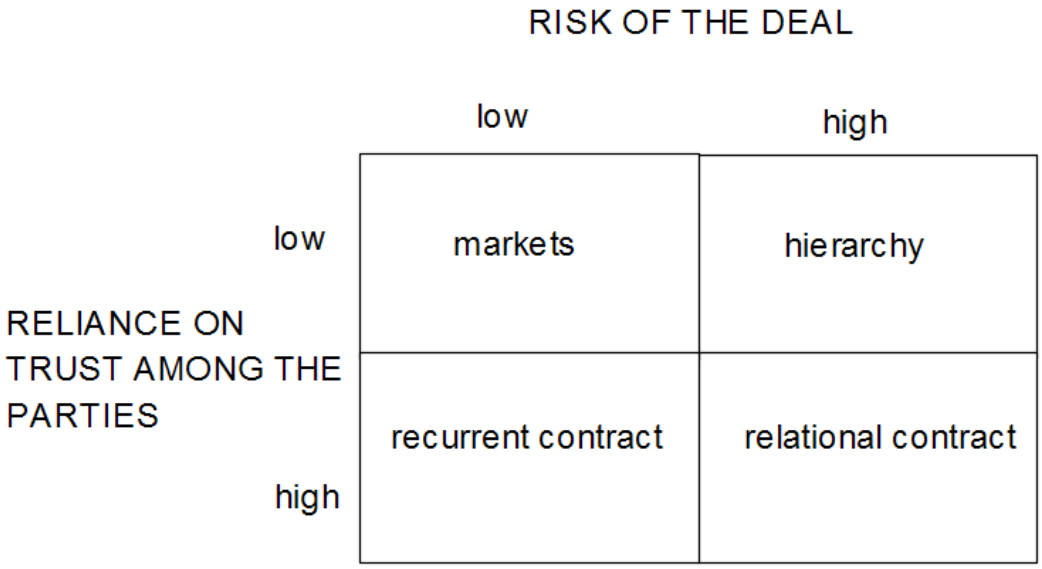


Figure 3.5: *Typology of governance structures based on risk and trust. Hierarchical authority can help resolve conflicts*[15].

where establishing high trust among peers, and employing policies and contracts may be required, has been recommended for future work (see Section 9.4). Establishing trust in distributed systems is challenging and an area of current critical research[125,126,127].

However, rather than enforcing conflict resolution entirely through hierarchical relationships, this idea has been used to motivate the development of the new agent-biasing mechanism described in later sections. The biasing features enable structures to support the management of potentially-conflicting goals, similar to the way an employee handles tasks from a variety of administrative and project managers.

3.13 Modeling Complex Organizations

Additional disciplines provide concepts helpful when designing supports for managing conflicts in complex, cooperative, holonic systems. Kineman provides an illustration of an advanced holonic system that introduces the idea of modeling multiple *contextual domains*, relational holons, and complex systems[16]. An example is shown in Figure 3.6. The work

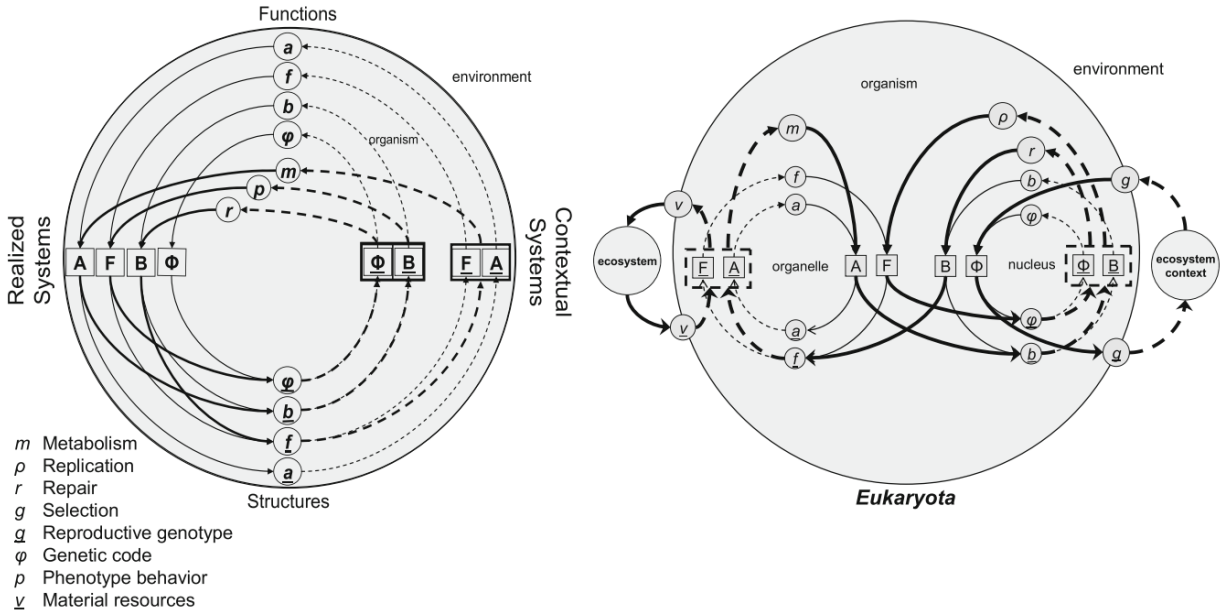


Figure 3.6: *Holon diagrams for Eukaryota showing interrelationships between contextual domains*[16].

discusses a holon as a composition of structure and function and explores the relationship to *adaptive interaction* with the agent's environment. This foundation is reflected in the inner organization of the new agents and the multiple inter-related contextual ontologies enabled by participation in multiple, independently-governed, yet related (or overlapping) organizations. Although not formalized in this effort, additional research into relational science, category theory, and the modeling of complexity could be helpful to further formalize reasoning about agents participating in multiple organizations and the associated need to detect and manage potential conflicts as suggested in Section 9.4.

Work with organization-based MAS has brought a more organization-focused design approach that supports loose-coupling between agents and the organizations they implement. Work with Moise, and Moise+ provide formalized insights into the required structural, functional, and *deontic* specifications of organizations[50,81]. Deontic refers to *duty* and the permission and obligation of agents to perform roles in the organization. The work done in this research follows the loose-coupling suggested with Moise and O-MaSE[50,81,49,12] and

applies some of their key concepts in organization specifications as described in Section 6.7.

3.14 Power Distribution Systems (PDS)

Power Distribution Systems (PDS) (Definition 2.5) transmit electrical power from power generation facilities to substations, down feeder and lateral lines to the neighborhoods where it is consumed by customers in homes and businesses as shown on the right side of Figure 3.7. The energy distributed via PDS drives almost every aspect of life in modern developed countries. Failure to deliver power reliably can have significant impacts. Unfortunately, according to some reports, 80% of customer power interruptions are due to failures in their PDS[128]. In addition, the potential for interruption is higher than necessary because many lines are deployed overhead as underground deployment costs five to ten times more. Considerable research is being done on applying agent systems and more recently, holonic MAS to power distribution systems[129,130,131,7,132].

Although PDS are critical to power systems, little real-time information is available to operators. Most of the planning and operations are based on archived information from historical load research. Typically, the only real-time information available is related to the feeder gateway at the substation. Thus, most PDS operate non-optimally and have problems recovering from abnormal events. Recent technological advancements and the increased use of renewable energy have made it obvious that current level of automation is insufficient. Many people are opting to install rooftop solar panels and energy storage devices in their homes, which poses new challenges and new opportunities to power companies. While controlling power quality can become more difficult in the presence of renewable resources, parts of the system may still be able to deliver locally generated power even after becoming disconnected from the rest of the PDS.

3.15 Online Auctions

Distributed renewable generation also brings new power producers to the market[133]. Rooftop photovoltaic (PV) panels allow home owners to generate more electricity than personally needed and this excess production could be voluntarily sold to nearby homes, alleviating additional transmission costs especially in rural areas[134]. Electricity is sold as a continuous quantity and the associated markets involve pricing that may change on a minute-to-minute basis. Forward markets assist with scheduling capacity and energy in advance[135]. The speed and complexity of the calculations needed to support distributed online auctions is a good fit for intelligent agents[136].

The interest in agent-led online auctions and the desire to evaluate suitable auctioning algorithms motivated the selection of the second MAS to be implemented by the IPDS agents. For this, a second, independently-goal-directed holarchy operating on the same underlying physical PDS was implemented to operate concurrently with the existing grid control holarchy to test the architecture and implementation mechanisms to see how well it worked for implementing multiple MAS with multigroup agents.

3.16 HMAS for PDS

PDS are naturally distributed and hierarchical in structure. The ability to proactively optimize the network and adapt to interruption events such as disconnects is obviously a desirable property in modern PDS. The requirements for *reactive* and *proactive* adaptation across highly distributed systems is a perfect fit for the use of MAS. PDS tend to branch out hierarchically from a single substation. At each level, an organization of supporting elements can be used as parts of higher-level organization. Such implementations may be referred to as a holonic organizational multiagent system (HOMAS)[53] or as simply a *holonic* multiagent system (HMAS)[24]. Organizations can be created that map a hierarchy of agents to the physical system. When recursive organizations are associated with an underlying physical

component, the multi-layer hierarchy may be referred to a as a *hierarchical HMAS* or HH-MAS. While MAS have seen significant attention in power systems, HMAS and HHMAS are just starting to be introduced to PDS[8,28,130,131].

Active research projects focus on power auctioning, negotiating, volt-var control, distributed communications, and other focus areas[137,138,139,140,141,142,143]. Recent research also applies *holonic multiagent systems* (HMAS) to *power distribution systems* (PDS)[132,130,131], sometimes in concert with specific agent-oriented software engineering methodologies.

Using HMAS to control a PDS is a natural fit. Intelligent agents are generally assumed to exhibit autonomy, reactivity, proactivity, and social ability[144], while MAS are, by nature, reactive and proactive. In addition, the social ability of agents allows them to work collectively towards the common good in a variety of configurations. Finally, the autonomous nature of agents allows them to make decisions based on local knowledge and constraints, thus allowing the system to adapt quickly and efficiently to its changing environment. Unfortunately, unrestrained MAS often exhibit a phenomenon known as emergent behavior, which can be either beneficial or harmful. One approach to harnessing the positive qualities of MAS while constraining emergent behavior is through the use of organization-based MAS[122,145]. In an organization-based MAS, agents are assigned to play well-defined roles in the organization in order to achieve the organization's goals. Organizational policies constrain the behavior of the organization and techniques and metrics have been developed that can predict the overall behavior exhibited by organization-based MAS[18].

An example hierarchic HMAS (HHMAS) for a PDS is shown in Figure 3.7. In such a system, each agent at level n may actually represent an organization of agents at level n-1, which may again represent an organization of agents at level n-2. While similar to traditional hierarchical control systems where control passes from the top levels to the lower levels, there is a major difference. Since each level consists of one or more organizations, if a connection is lost between level n-1 and level n-2, organizations at level n-2 can still operate autonomously, attempting to achieve their goals as best they can.

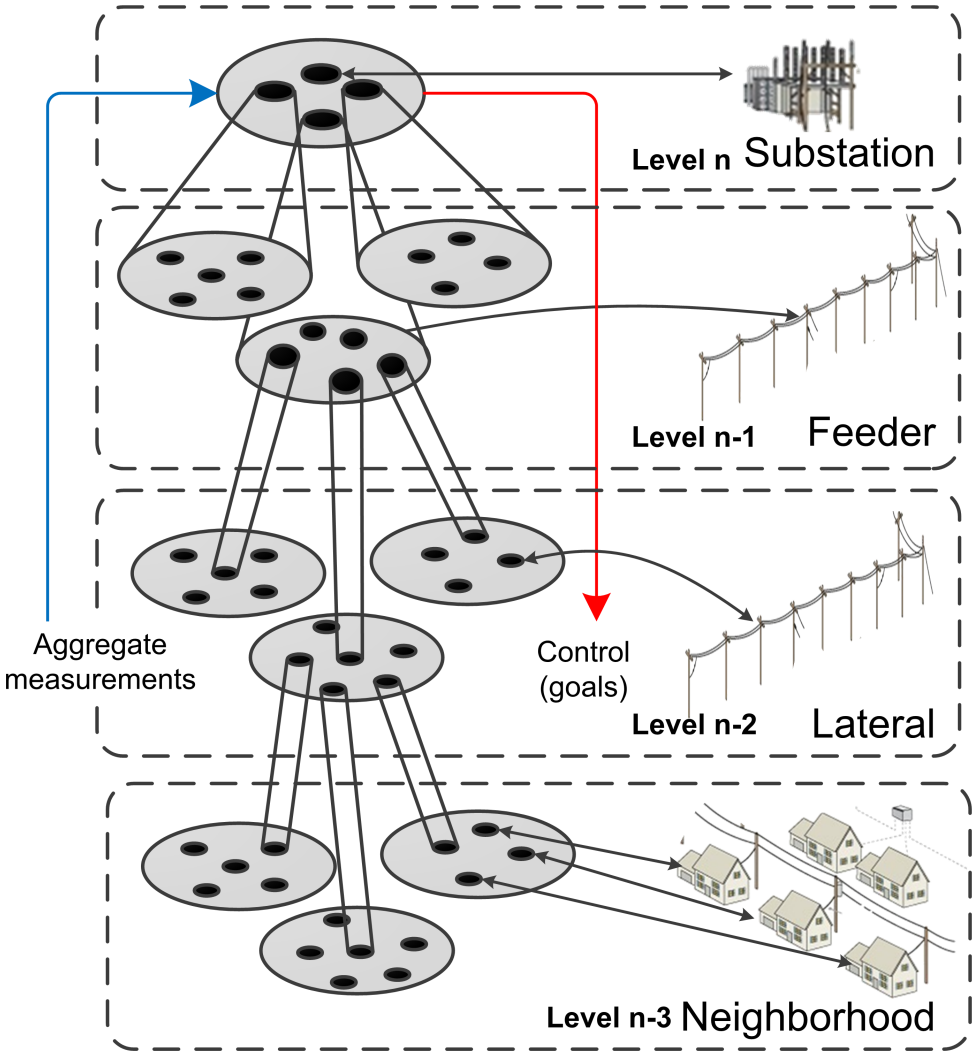


Figure 3.7: *Holonic multiagent system mapped to physical power distribution system*[7].

3.17 Intelligent Infrastructure

Research into control systems for infrastructure go beyond energy systems into transportation, healthcare, manufacturing, buildings, communities, agriculture, defense, and aerospace[1,21,146]. Holonic control systems have been proposed for sectors from manufacturing[147,17] to street lighting[148] to power distribution systems[7].

An example of a holonic control structure for prosumers in a power distribution system is shown in Figure 3.8[8]. Prosumers are capable of both producing and generating electricity. Groups of lower-level prosumer holons, such as homes, may be see as sub-holons within a higher, neighborhood-level holon. The higher control holon provides terms and conditions down to the lower-level holons. But information about power transactions and control parameters flow both directions – both down from the holon to the sub-holons and up from the sub-holons to the higher holon. Sub-holons also interact with their peers on the same level to negotiate power transactions. The control approach can be repeated as needed through increasing levels of aggregation[8].

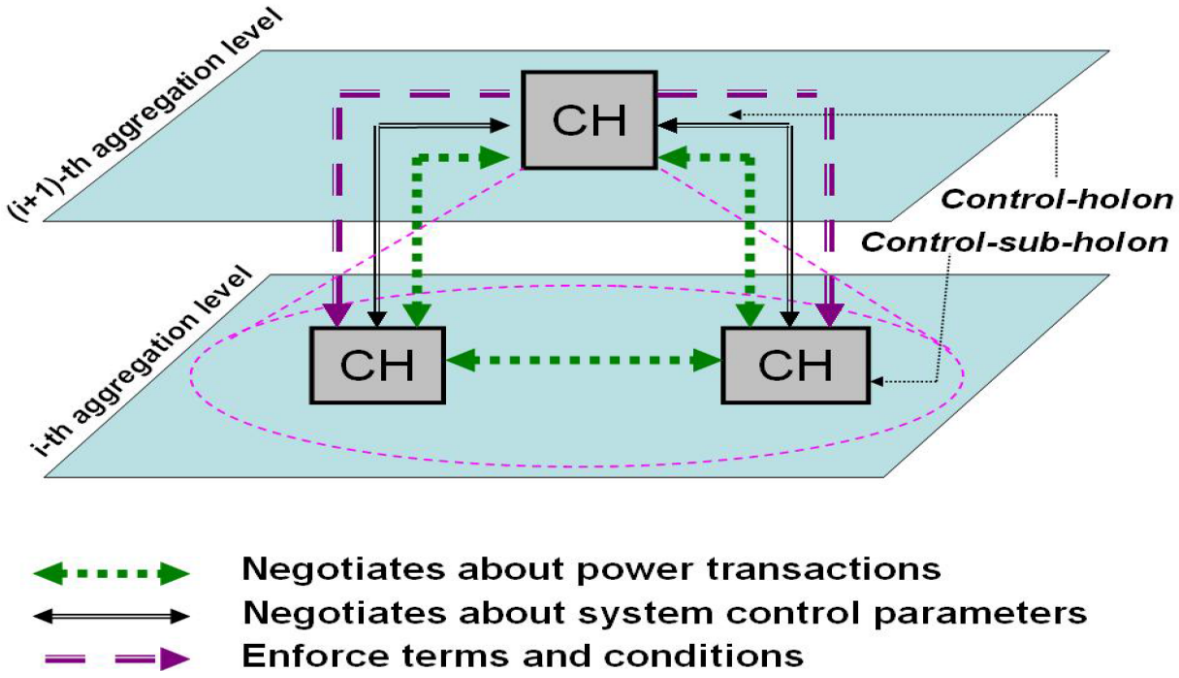


Figure 3.8: *Holonic control structure for prosumers in a power distribution system*[8].

3.18 Agents in Complex Organizations

Holonic designs often require agents to operate as both part of a higher holon and an a lower holon. The ADACOR architecture designs holons in a 5-level architecture that was especially interesting[17]. The five levels include planning, management, coordination, operation, and physical as shown in Figure 3.9. In ADACOR, different types of holons are placed at different levels. Product holons form the process planning level, task holons form the management level, supervisor holons form the coordination level, and operational holons form the lowest, operational level. Most holons interact vertically between levels, except for the operational holons which interact with peers in order to synchronize their activities. Task holons interact with operational holons for distributed monitoring and scheduling while the addition of the supervisor holons enables more global, centralized optimization. The adaptive ADACOR mechanism emerges in a bottom-up approach, built upon the individual self-organization of manufacturing holons[17].

ASPECS offers an agent-oriented software process for engineering complex systems[18]. This process and the associated framework enables agents to play roles in multiple groups as shown in Figure 3.10. ASPECS supports modeling systems at various abstraction levels through hierarchical behavioral decomposition based on roles and organizations. Systems are recursively decomposed until behaviors can be managed by atomic entities[18].

The goal of this research was to design an agent architecture that would work for a variety of complex organizational structures, as well as agents supporting different systems, that would provide explicit support for managing different objectives and domain requirements concurrently. A flexible, reusable architecture was needed for implementing goal-driven, intelligent agents capable of participating in different groups, and in different systems concurrently.

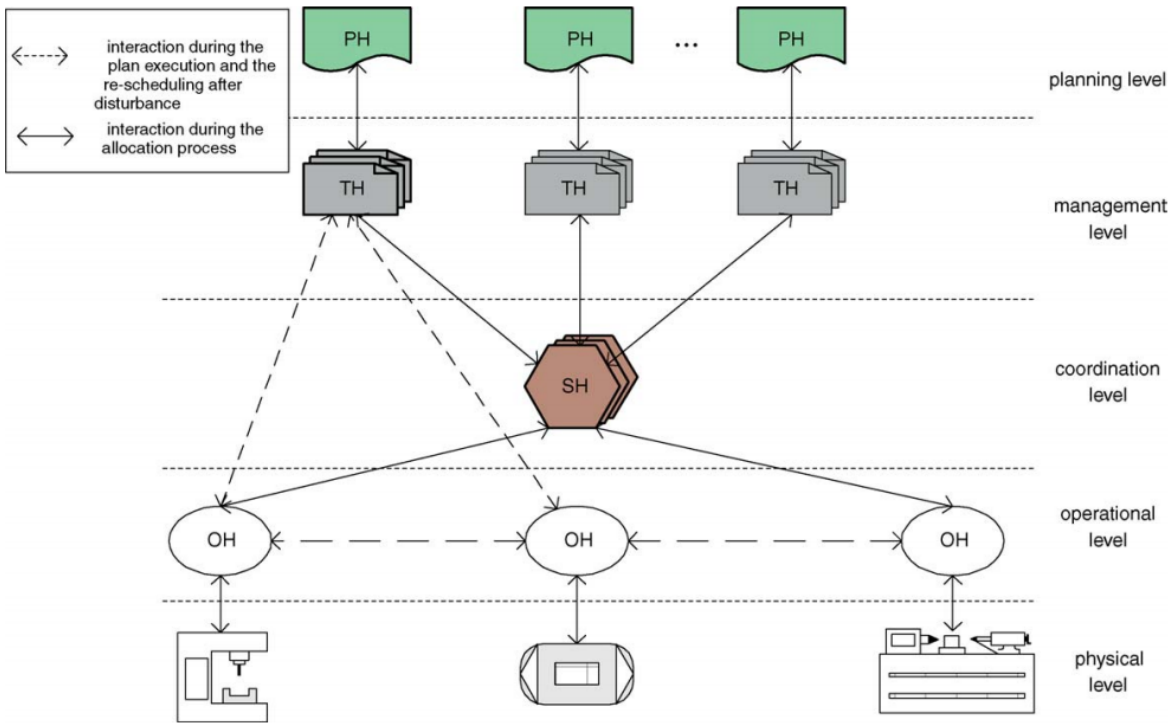


Figure 3.9: *The ADACOR holonic architecture for agile and adaptive manufacturing control arranges holons into five levels*[17].

3.19 Summary

This chapter presents a brief overview of key prior work that motivated and enabled this work, beginning with distributed artificial intelligence, complex adaptive systems and cyber-physical systems, continuing through prior work in the engineering challenges of MAS, and finally recent work in the specific application areas that motivated the simulation experiments for the multigroup agents. This background was used in the development of the updated organization-based agent architecture described in Chapter 4.

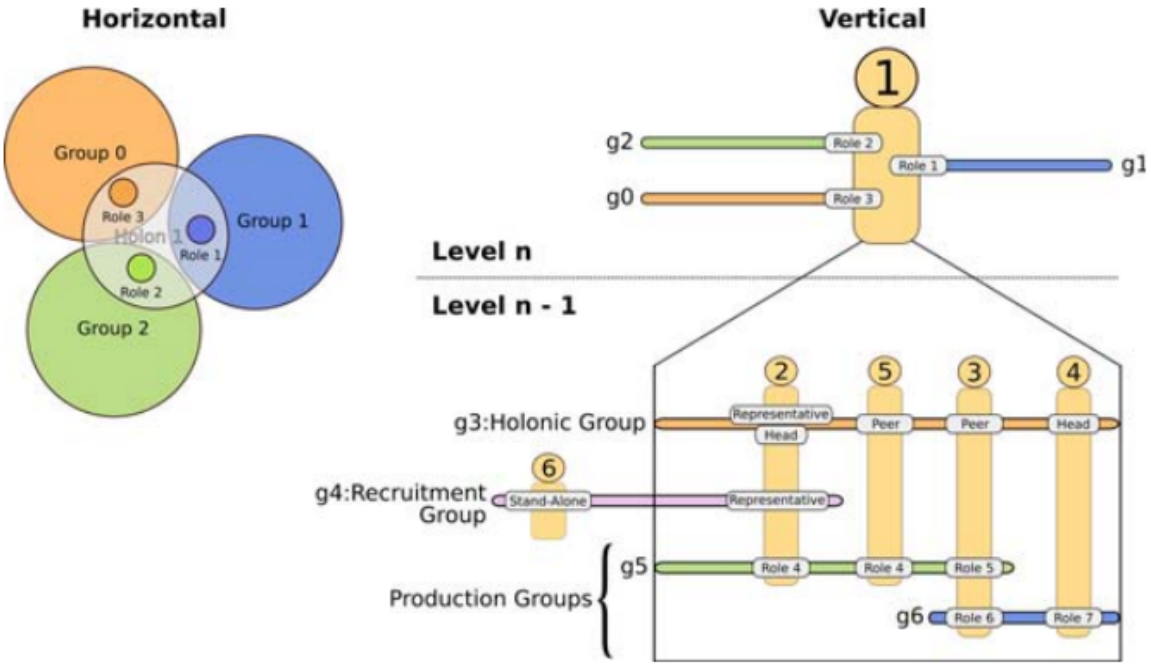


Figure 3.10: *The ASPECS process allows agents to play roles in multiple groups*[18].

Chapter 4

OBAA: Basic Agents for MAS

Great things do not just happen by impulse,
but are a succession of small things linked together.
— *Vincent van Gogh*

Building on key concepts from the *Organization-based Agent Architecture* (OBAA)[14], an updated agent architecture was developed. The updated architecture is modular, loosely coupled, extensible, and still designed for single-organization multiagent systems. This chapter provides an overview of the key components of the agent architecture that are needed to establish the background an prior work on which this dissertation builds. The next chapter, Chapter 5, shows how these basic agents provide a foundation for *persona* – new subagents that provide the necessary foundation for implementing complex *multigroup MAS*.

Section 4.1 introduces the motivation and objectives for updating the architecture. Section 4.2 introduces key areas of reusable functionality in *organization-based MAS*. Section 4.3 provides an overview of the updated *organization-based agent architecture*. Section 4.4 describes the *execution component* (EC) for performing application-specific functions and

Section 4.5 describes the *control component* (CC) for performing common aspects of organizational administration and participation. Section 4.6 describes architectural aspects of different *organizational decision-making styles* such as operating in a master-slave configuration. Section 4.7 summarizes the updated basic agent architecture and Section 4.8 provides a chapter summary.

4.1 Motivation

Organization-based MAS and specifically, the *Organization-based Agent Architecture* (OBAA) developed by Chris Zhong[14] provide many desirable features. OBAA works well for implementing single-organization MAS. The updated architecture continues to employ many of the features first suggested in OBAA, but has updated certain elements to be more modular and less tightly coupled. The updated architecture enables reuse of key elements to form distributed, complex systems as described in Chapter 6. The architecture continues to use many features originally designed and proposed by Chris Zhong[14] and Rui Zhuang[149].

4.2 Reusable Organization Functionality

At its most basic, an agent has capabilities and can be given assignments. The capabilities are used to execute plans. Agents in an application are equipped with capabilities and carry out application-specific plans.

Application-specific functionality can vary greatly – some applications may require movement in two dimensions (as in simple games), or movement in three dimensions (as in robotic teams). However, it is possible to design multiagent systems with a common core set of functionality related to assignments.

Definition 4.1 (Organization-based agent)**.** An *organization-based agent* is an agent capable of reasoning about its organization[13], with the ability to *reorganize*, or transition

from one organizational state to another, in response to updated goals or changes in the environment.

Definition 4.2 (OBAA)**.** The *organization-based agent architecture* (OBAA) is an agent architecture for organization-based agents that use OMACS for reasoning about its organization[13].

OBAA agents are designed such that all MAS created from OBAA agents can employ common features for creating, issuing, and accepting assignments[150].

4.3 Overview

The updated single-organization architecture provides additional reusable components that provide an enhanced foundation for extending the architecture as described in Chapters 5 and 6. These foundations will be used to enable systems of intelligent systems as discussed in Chapter 8.

An overview of the updated OBAA agent architecture is shown in Figure 4.1.

The agent is divided into two areas of functionality:

- *Control component (CC)*. The upper portion associated with common control component aspects, associated with generating, issuing, and accepting assignments
- *Execution component (EC)*. The lower portion associated with application-specific capabilities and plan execution

4.4 EC: Application-Specific Execution

The application-specific *execution component* (EC) is associated with capabilities and plan execution. At its most basic level, agents have capabilities and use them to execute plans. This basic functionality is part of the EC shown on the diagram. A simple autonomous

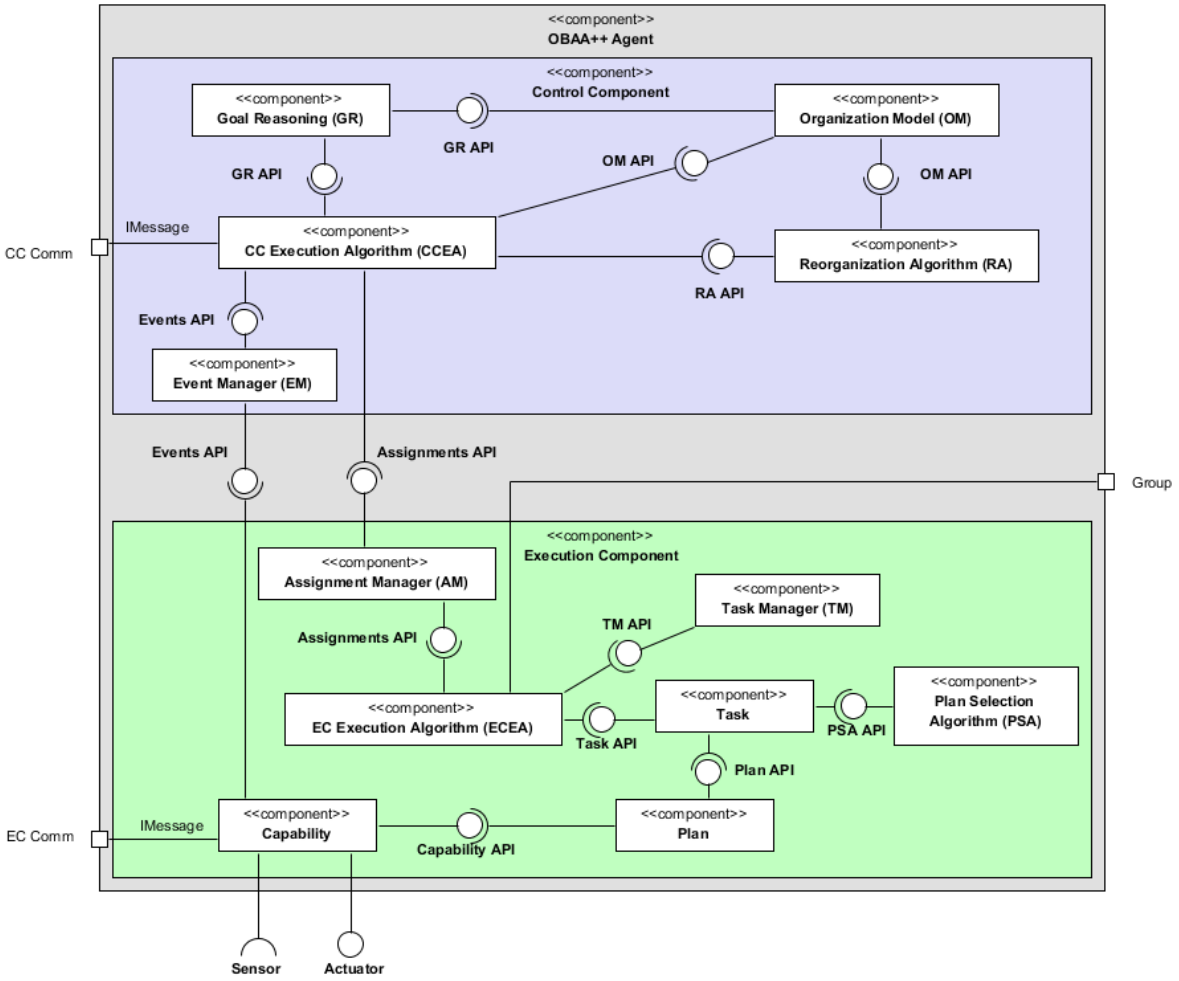


Figure 4.1: *Overview of the updated basic agent architecture.*

agent could use just this functionality to execute plans autonomously. As shown in the lower component on Figure 4.1, the EC includes seven identified components:

1. *Assignment Manager* (AM)
2. *EC Execution Algorithm* (ECEA)
3. *Task Manager* (TM)
4. Task
5. *Plan Selection Algorithm* (PSA)

6. Plan

7. Capability

As described in the following sections, these components enable the agent to execute application-specific functionality.

4.4.1 Receiving Assignments

Every EC includes an Assignment Manager (AM). The AM maintains the interface to the entity issuing assignments to the agent. In each agent, the AM is given assignments by the CC. Every assignment includes the EC, a role, and a goal. The assignment goal may include goal parameters, describing specific guidelines for achieving the goal.

As the MAS runs, the AM maintains three queues and, together, these queues reflect the current desired behavior for this agent. The AM maintains a separate queue for each of the following:

- Assignments. For new assignments.
- Deassignments. For current assignments that are being revoked.
- Goal Modifications. For changes to a current assignment, specifically changes to the guidelines, i.e., the goal parameters associated with an assignment to adjust the desired behavior associated with the assignment goal.

The AM can be customized to perform checks on assignments and assignment modifications before accepting them, and may be configured to provide feedback if an assignment is refused or the guidelines altered to reflect the specific preferences, biases, or other requirements of the agent. This is perhaps less of an issue for agents operating as they do in a single MAS, but is very helpful when extending these core components to handle multiple organizations as described later, in Chapter 4.

4.4.2 EC Execution Algorithm

The Execution Component Execution Algorithm (ECEA) operates continuously as long as the agent is alive and not disabled. The ECEA provides the new assignment updates maintained by the AM to the Task Manager, which prioritizes the assignments and maintains progress and status information for each assignment. With each computational cycle, the ECEA will execute a step in its assigned task (or tasks). After executing task steps in each cycle iteration, or if there are no active tasks, the ECEA will end the iteration with a call to execute the CC Execution Algorithm as described in Section 4.5.2.

The ECEA is customizable. A basic ECEA is shown in Algorithm 1.

Algorithm 1 EC Execution Algorithm

```
1:  procedure ExecuteECEA
2:      am ← assignmentManager()
3:      tm ← taskManager()
4:      cc ← controlComponent()
5:      while isAlive() do
6:          while am.hasAssignments() do
7:              tm.addTask(new Task(am.pollAssignment())
8:          while am.hasDeAssignments() do
9:              tm.removeTask(am.pollDeAssignment())
10:         while am.hasGoalModifications() do
11:             tm.updateTask(am.pollGoalModification())
12:         t = tm.getNextTask()
13:         if t ≠ ∅ then
14:             executeTask(t)
15:         cc.execute()
```

4.4.3 Managing Tasks

As described in the ECEA, the Task Manager (TM) is responsible for maintaining a prioritized list of *tasks* for the agent. Each task includes:

- The associated assignment.

- Status information, e.g., achieved, in progress, failed.
- The selected *plan* for accomplishing the task.

4.4.4 Choosing a Task Plan

MAS implementations include custom versions of the Plan Selection Algorithm (PSA) for selecting the best plan for accomplishing a task. The algorithm may vary in complexity, from simple look-ups to complex algorithms that may take into account the current status of the agent, participating humans, nearby entities, or the organization as a whole, as well as current objective functions and constraints. If the task plan is not yet determined and the task is active, the task will call the PSA to get the selected plan.

4.4.5 Executing Plans with Capabilities

When the ECEA calls for execution of the next step in its active task or tasks, the task will execute one step, or cycle, of the selected *plan*. Plans are executed by calling methods and functions from the agent's *capabilities*.

Agent capabilities generally come in three major types:

- *Sensor* capabilities. The ability to access devices that provide information or readings regarding perceptions of the environment, humans, or other agents.
- *Actuator* capabilities. The ability to access devices that be take actions.
- *Processing* capabilities. The ability to perform mathematical or other computational functions.

Capabilities may have parameters that can be set. The necessary information may be passed in via goal parameters or guidelines that provide specific information about desired behavior. An agent may continue with an ongoing task that has a parametrized goal, getting updated guidelines to vary the behavior in response to current organizational objectives.

Agents may be configured with their own preferences or biases, and these can be used to adjust the issued parameters to reflect individual preferences. Goal consistency and customization is discussed in more detail in Section 6.13.

4.5 CC: Organizational Control

In Figure 4.1, the common organizational functionality given to each agent is encapsulated in the control component (CC). Every OBAA agent has a CC. When the agents are first created, the agent creates and initializes the CC, which will in turn, initialize the EC with its first set of assignments. As shown in the upper component on Figure 4.1, the CC includes five key components:

1. *Event Manager* (EM)
2. *CC Execution Algorithm* (CCEA)
3. *Goal Reasoning* (GR)
4. *Organization Model* (OM)
5. *Reorganization Algorithm* (RA)

As described in the following sections, these components enable the agent to perform the functions necessary to generate, issue, and accept assignments.

4.5.1 Reacting to Events

Just as the application-specific EC is driven by the assignments provided by the CC, the CC is driven by *events* generated by the EC. These events are raised during the execution of plans via the methods in EC capabilities. When a method is called, it may raise various events (e.g., when an agent is disabled, execution of a task fails, or when a goal is achieved or modified). The CC Event Manager manages the queue of events provided by the EC.

4.5.2 CC Execution Algorithm

The Control Component Execution Algorithm (CCEA) operates continuously while the agent is alive and not disabled. With each execution cycle, the CCEA processes the events managed by the EM and checks for messages received from other control components. If, after processing the events and CC messages, the CCEA determines reorganization is necessary, it will call the Reorganization Algorithm (RA). The RA requires the ability to perform goal reasoning and understand and update knowledge regarding the current state of the organization and is described in Section 4.5.3 through 4.5.5.

Implementation of the CCEA depends on the *organizational decision-making style* employed by the organization, and on the *organizational role* of the CC within that approach. Some organizations operate in master-slave configurations, but other organizational styles are possible as well[151]. Organizational styles are discussed further in Section 4.5.3.

The CCEA is customizable. A basic CCEA during the ongoing continuous working state is shown in Algorithm 2.

Algorithm 2 CC Execution Algorithm

```
1:  procedure ExecuteCCEA
2:      reorgNeeded ← false
3:      em ← eventManager()
4:      ra ← reorganizationAlgorithm()
5:      while em.hasEvents() do
6:          processEvent(em.pollNextEvent())
7:      while ccComm.hasMessages() do
8:          processMessage(ccComm.getNextMsg())
9:      if reorgNeeded then
10:         reorganize()
```

4.5.3 Understanding the Current Organization

The CC maintains the current state of the organization in the Organization Model (OM). The OM includes information about agents and capabilities, current assignments, roles, goals, policies, and other aspects and relationships that describe the state of the organization. The architecture uses OMACS for the OM as defined in Section 3.8 and shown in Figure 3.3.

4.5.4 Goal Reasoning

The CC includes a Goal Reasoning (GR) component that uses a static goal specification to generate and update the set of active runtime instance goals. During initialization, all untriggered goals are instantiated and set as active. Active instance goals are issued to participating agents. As the agents work on the goals, new goals may be triggered, and goals may be achieved, failed, or obviated (as when achievement of one goal makes another goal unnecessary). The architecture can use GMoDS for the GR. GMoDS is described in Section 3.7.

4.5.5 Reorganizing

As execution continues, and the environment, the agents, and/or interacting humans change state, and as goals are added, modified, and removed, the organization may need to *reorganize*. During the process of reorganization, the set of assignments are updated. The roles that can achieve the set of active goals are reviewed, as well as agents capable of playing the roles. The capabilities with which each agent is equipped and how well a role fulfills a goal are used to determine the optimum set of assignments. The new assignments are issued to the EC Assignment Manager (AM) as a set of three queues, one for new assignments to be added, one for current assignments to be removed, and one for modifications to current assignment goals as described in Section 4.4.1.

4.6 Organizational Styles

OBAA organizations may employ different organizational decision-making styles. One common style employs a simple *master-slave configuration*, but other organizational styles are possible as well[151]. In the master-slave configuration, one master agent is responsible for registering all participating agents, for getting the initial goal guidelines, and for generating and issuing assignments for all of the organization participants including itself. All agents that are not the master act as slaves and register with the master to receive issued assignments.

4.6.1 Organizational Roles

The organizational approach defines a set of *organizational roles* the CC may play. For example, in the master-slave configuration, there are two organizational roles: (1) master and (2) slave. Other approaches, such as voting or other peer-based approaches, may have a different set of possible roles. Organizational roles should not be confused with the EC roles; EC roles are set by the RA during a reorganization. Organizational roles may be specified and remain static throughout the operation of an organization, or they could be dynamic, as when a new master is selected to replace a master that has been disabled or is otherwise unable to continue.

Organizational roles are used to keep track of which agents are active in the organization and can accept assignments within that organization. The CC master issues assignments and both the CC master and the CC slaves can accept assignments.

The behavior for a typical CC master is shown in Figure 4.2. All control components, including the master, begin by initializing their associated application-specific execution component. When the process is complete, the master moves to the continuous working state. During this state, the master checks for registration messages from participating slaves, and responds to events from its own associated EC and others, which are communi-

cated as messages via their associated slave CC. The master maintains a list of active agents to which assignments may be issues and creates and issues assignments to itself and others in accordance with the current state of the organization. At any time, if an end event is detected, because of system shutdown, or in the event participation privileges have been revoked, the agent will move to a terminating end state.

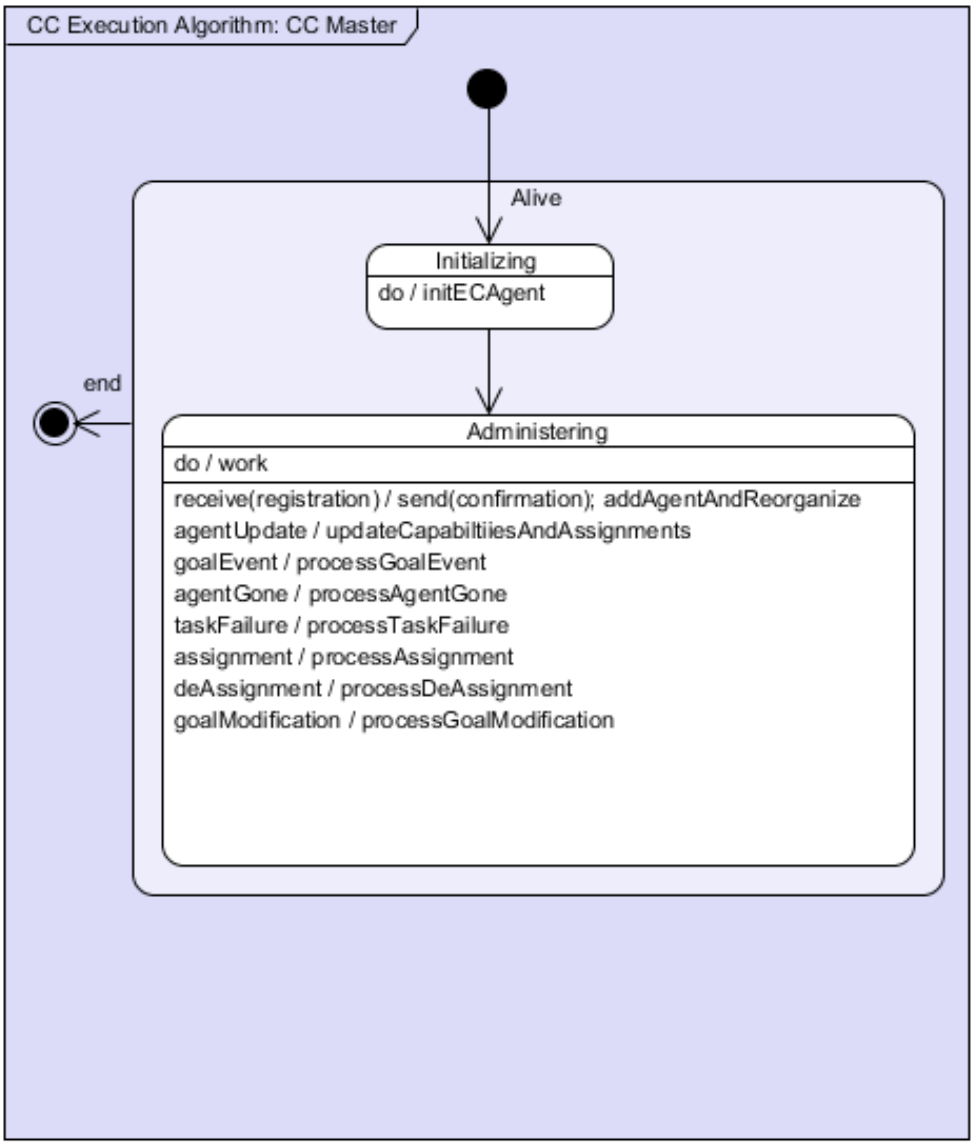


Figure 4.2: *Behavior of a basic control component master.*

The behavior for a typical CC slave is shown in Figure 4.3. As in the master, each slave

begins by initializing their associated application-specific execution component. When the process is complete, the slave sends a registration message to its master and moves to the confirmation registration state. The agent waits here until it receives a confirmation from the master. If the agent waits longer than a specified time limit, the agent times out and returns to the registration state to try again. If the agent does not get confirmation from the default master within a given amount of time, the agent elevate itself, if equipped, to act as a new master or can begin to perform independently, as during periods when the agent becomes disconnected from its master. After receiving confirmation, the agent moves to the continuous working state. In this state, the slave will continuously process any new assignments, deassignments (the removal of an active assignment), and goal modifications. It will also monitor events from its associated EC and send organization-related events, such as task failure events to the CC master for processing. At any time, if an end event is detected, because of system shutdown, or in the event participation privileges have been revoked, the agent will move to a terminating end state.

4.6.2 Organizational Capabilities

Like organizational roles, organizational capabilities are different from, but analogous to, EC capabilities. Each agent's CC is equipped with functionality that determine its ability to play various organizational roles. This may not be used much in single-organization MAS, but when systems become more complex and distributed, as described in Section 6, the ability for agents to on new organizational roles may be required.

4.6.3 Master-Slave Configuration

An example of a single-organization MAS built with basic agents is shown in Figure 4.4. The figure shows three agents operating in a *master-slave configuration*. The Supervisor Agent is acting as the organizational master, and is shown in more detail in Figure 4.5. The Forecaster agent and the Worker agent are both acting as slaves to the organizational

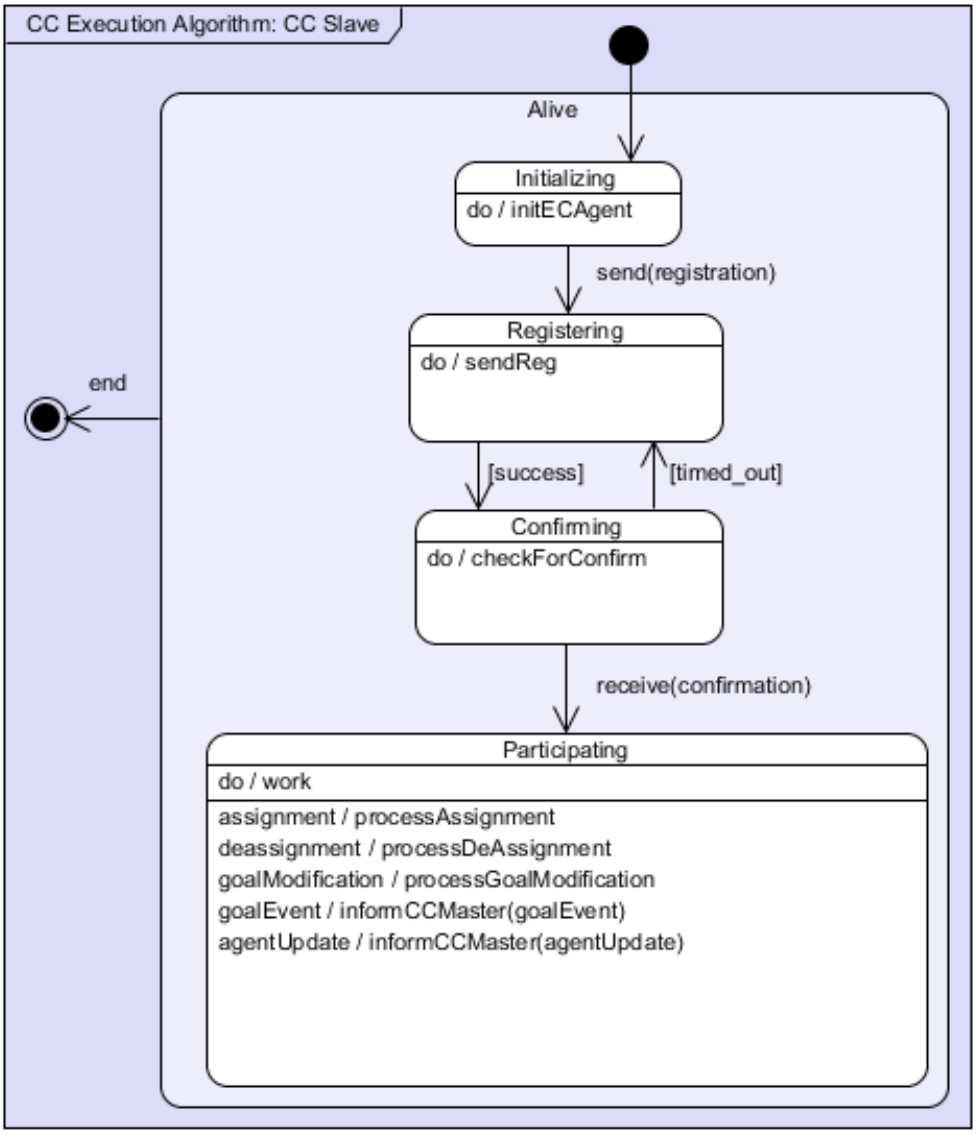


Figure 4.3: *Behavior of a basic control component slave.*

master. A participant device agent is shown in more detail in Figure 4.6.

4.7 Basic Agents (Persona)

Each organization-based agent has a general-purpose Control Component (CC) and an application-specific Execution Component (EC), allowing separation of the general features needed for group participation from the execution of application objectives.

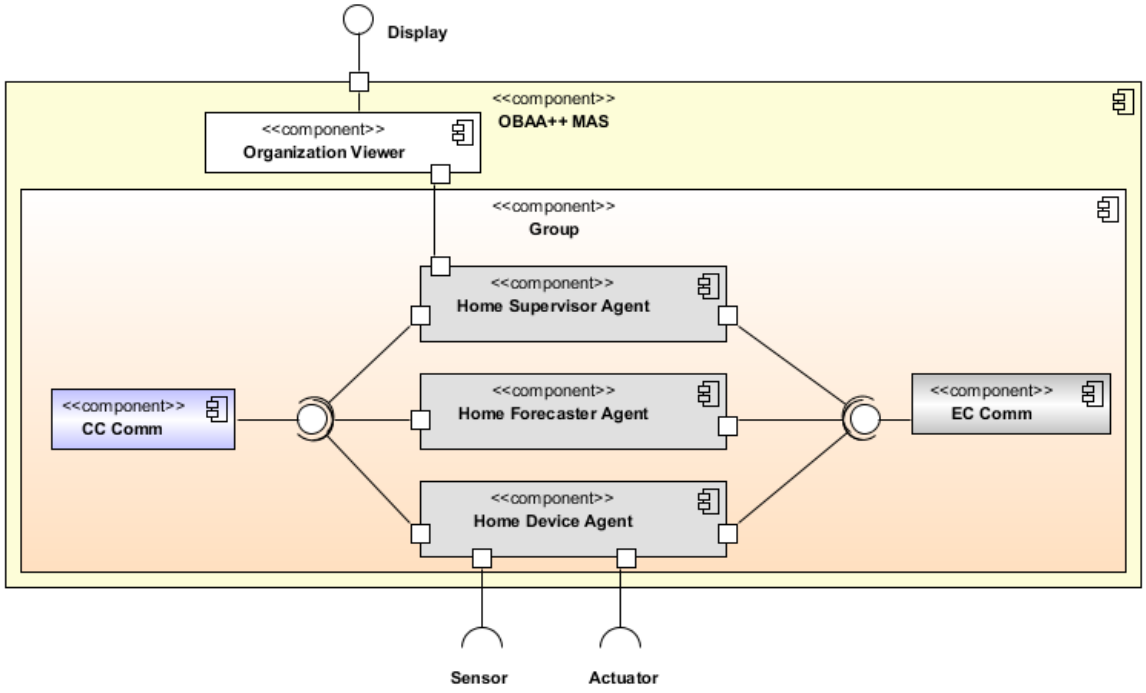


Figure 4.4: *Example single-organization multiagent system.*

The agent is derived from the basic EC shown as the lower component. The agent includes the functionality in the base EC class shown in the lower component as well as the CC functionality that has been abstracted into the upper component. The specific implementation of the CC reflects the organizational approach and the corresponding organizational role (or roles) the CC can play. In a similar fashion, the architecture provides for an application-specific implementation of the base agent that reflects the core functionality needed to serve as an agent in that type of organization. Most of the agent functionality is provided in the specific EC capabilities with which the agent is equipped.

In the next chapter, a special type of MAS is introduced, that uses this updated agent architecture as the foundation for a new type of agent called a *persona*. By using persona, and changing the definition of an agent (quite a bit), a flexible, reusable architecture was developed that enables complex MAS.

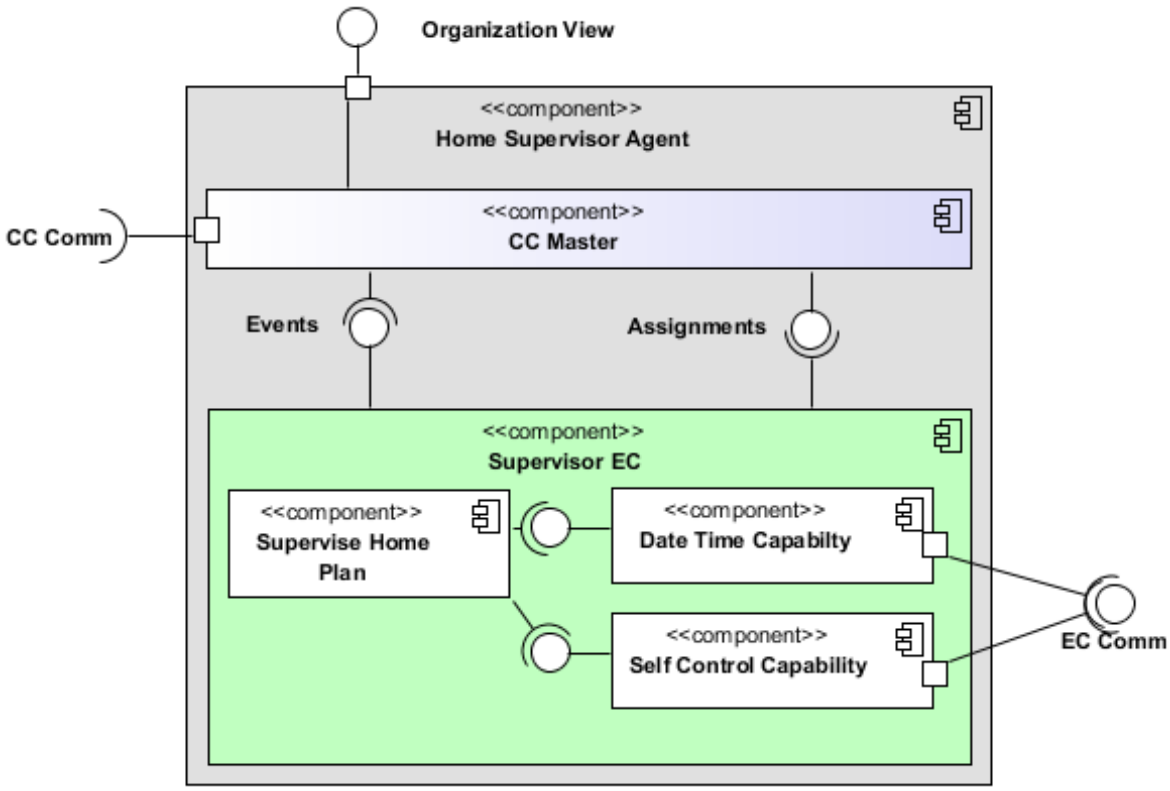


Figure 4.5: *Supervisor Agent acting as CC master in a master-slave configuration.*

4.8 Summary

This chapter presents an updated architecture for agents that create, operate, and participate in a MAS. It describes how the architecture, includes a control component (CC) for common organizational functionality and an execution component (EC) for application-specific behaviors. It describes key entities within each of these and provides an overview of the way the supporting components work together to enable single-organization MAS. In addition, this chapter describes various organizational styles for generating and issuing tasks, and provides a more detailed example of a MAS employing a master-slave configuration for assigning goals. It describes how each basic agent is designed with a *flexible*, domain-specific EC, while the supporting CC offers organizational control functionality that is *reusable* across many different types of applications.

It describes the motivation for updating the OBAA architecture so that these basic

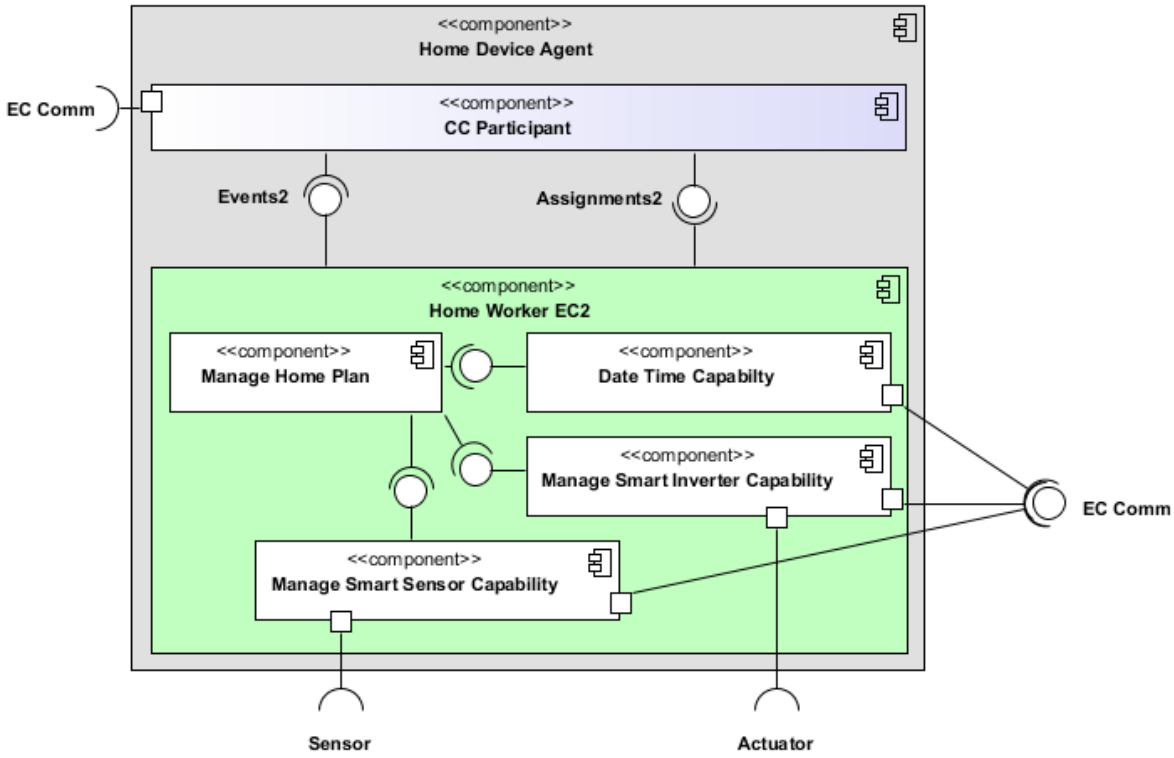


Figure 4.6: *Worker Agent acting as CC slave in a master-slave configuration.*

agents can be used in a special type of MAS that enables multigroup agents. The *basic agent* introduced here forms the foundation for a subagent, called a *persona*. By using this new persona type, and changing the definition of an agent (quite a bit), a new flexible, reusable agent architecture for complex, multigroup MAS was developed as described in Chapter 5.

Chapter 5

OBAA^{++}: Agents are MAS

I am large, I contain multitudes.

— *Walt Whitman*[152]

In the this chapter, a special type of *multiagent system* (MAS) is introduced, one that uses the updated basic agent architecture presented in Chapter 4 as the foundation for a new type of agent, or sub-agent, called a *persona*. These persona, along with a significant change to the architecture of an agent, enable a new flexible, reusable agent architecture for *complex, multigroup MAS*.

This chapter introduces the new *Organization-based Agent Architecture for Multigroup MAS* (OBAA^{++}). OBAA^{++} views each agent as a MAS of subagents or persona, with each persona built on the basic agent foundation described in Chapter 4. This shift in our view of an agent and its supporting standardized types of persona, provide a flexible, reusable architecture for implementing complex MAS.

Sections 5.1 and 5.2 provide motivation for developing a standard agent architecture to implement complex MAS. Section 5.3 shows how each multigroup agent is a MAS and Section 5.4 describes the implementation of an OBAA^{++} multigroup agent. Section 5.5

describes how each multigroup agent is implemented as a specialized MAS, an inner organization of persona, and Sections 5.6- 5.8 introduce the standard persona types. Section 5.9 describes communication *within* an OBAA^{++} multigroup agent and *between* OBAA^{++} multigroup agents. Section 5.10 provides a chapter summary.

5.1 Managing Complexity

One approach to managing complexity in distributed artificial intelligence is the application of organization-based agent architectures. Organization-based agent architectures provide standard patterns and models for designing and implementing organizations of agents in a MAS application[49,13,153]. Agents are computational entities that may function autonomously and/or collectively in MAS[43]. Some reader familiarity with autonomous agents and MAS is assumed; additional references providing more detailed background, characteristics, and motivations for agent-based systems are provided[154,155]. Agent systems may be considered a form of distributed artificial intelligent and have been used to implement control systems. Agent systems my be complex and the management of the many software components that drive emergent behavior can pose significant engineering challenges. Agent-oriented software engineering (AOSE), with its focus on goal-driven design, is seeing significant attention in the area of intelligent or smart systems, especially in the areas of infrastructure. K-State has active interdisciplinary research projects into agent-oriented control systems for the smart grid. This effort is motivated, in part, by the demands of the United States National Science Foundation (NSF) Intelligent Power Distribution Systems project[38,156,7].

5.2 Motivating Example

K-State developed an HHMAS (Definition 2.23) under the interdisciplinary *Intelligent Power Distribution System* (IPDS) project. In this project, intelligent agents work cooperatively in a distributed HHMAS to control an electrical power distribution system (PDS) with

a high degree of renewable penetration[7]. The HHMAS is a *complex MAS*, consisting of multiple organizations, reflecting the hierarchical nature of an electrical distribution system as described in Section 3.14.

A system for grid control was developed using holonic design principles, with the standard holarchy being divided into three layers. The highest layer functions at the substation level, with an intermediate feeder layer of groups, which are in turn composed of smaller groups at the lowest level associated with *neighborhood transformers*, each of which provides electricity to 4-6 homes, some of which may be equipped with *photovoltaic* (PV) solar generation panels that can produce enough *distributed generation* (DG) during a sunny midday to power the load of several homes.

When fast-moving clouds cover the PV panels, it can trigger an undesired and potentially harmful rapid drop in voltage. This can be partially alleviated by increasing the amount of reactive (non-useful) power from each set of PV panels. Help can be provided by agents controlling other PV panels, even from a distance, so long as the *phase* of the supporting lateral line is the same.

Just as the physical PDS system is arranged hierarchically, the proposed control algorithms were designed in recursive organizations of hierarchical *holarchies*.

The complexity of such systems can introduce additional challenges. When implementing holarchies with a complex MAS, an agent may be assigned goals from different organizations within a single system. Further, in complex CPS, agents may be assigned goals from different systems – and there are several ways an agent could be assigned a goal that might be in conflict with an assigned goal from another organization.

The new version of OBAA, OBAA^{++} was specifically developed to provide a set of reusable foundations that would enable multigroup agents. In OBAA^{++}, each multigroup agent is designed as a goal-driven *inner organization* of sub-agents[36]. To clarify the distinction between the agent and its sub-agents, the term *persona* is used for the specialized sub-agents. Each agent has one persona for each affiliated group and one *self persona* that

acts as the central, goal-directed brain of the agent, responsible for initiating and managing the roles and commitments within that affiliated group. Each agent is initialized with a set of goals that drives the behavior of the self persona. They include goals for joining and maintaining membership in specified affiliated groups. When a self persona determines that its agent requires a new affiliation, the self persona creates a new persona to join the group. Self persona function similarly for each agent in a complex MAS.

5.3 A MAS of Persona

Multigroup agents are those designed specifically to support participation in multiple groups concurrently. OBAA^{++} multigroup agents implement each agent as a system of intelligent persona, each built using the Organization-based Agent Architecture (OBAA). Each OBAA^{++} persona consists of two basic components, a general purpose Control Component (CC) and an application-specific Execution Component (EC)[14,36]. The CC manages the common functions associated with registering, and issuing or accepting goal assignments, which the EC manages execution of domain-specific plans for carrying out assigned goals.

Implementing agents as an inner organization of subagents called persona provides a reusable approach flexible enough to implement different types of multigroup MAS. In addition, the system enables different control systems running on a common physical system, to share functionality related to common sensors and control devices. The agent is goal-driven, and the same mechanisms used to specify goals in MAS can be used to customize agent behaviors. For example, a single goal model is created for each home agent that can specify the relative importance of the different affiliated groups in which the agent participates. Parametrized goals allow the standard home goals, reused among the home agents, to be customized to reflect the behavior desired either by the power company providing power quality direction, the local market organization providing online auction direction, or even by the homeowners themselves to reflect their biases towards community assistance or

maximizing personal profit.

5.4 Multigroup Agents

As mentioned in Section 1.1, designing and building multigroup agents is more difficult than developing traditional agents because each multigroup agent must be able to align its goals and behavior with the goals of all of its affiliated groups. Implemented as a special type of MAS, each multigroup agent operates as an organization of persona[36].

The internal organization of each agent has a unique self persona that acts as the master of the organization. Upon instantiation of the agent, the self persona is created and given a set of goals that drives the behavior of the self persona. These goals are generally application specific and includes goals for discovering, joining, and maintaining membership in the appropriate affiliated groups. The details of how the agent determines its affiliated groups are also application specific. However, when a self persona determines that its agent requires membership in a group, the self persona creates a new persona to join that group. The self persona instantiates the new persona with an appropriate set of goals and the new persona carries out behaviors to achieve its goals. The self persona typically functions identically for each agent in a complex MAS.

The framework is designed to allow a single agent to participate in multiple affiliated groups at the same time. Each persona can be designed, implemented, and tested individually before being integrated into a single agent by supplying appropriate configuration information for each agent, which can include the affiliated groups to join, whether the agent is responsible for creating those groups, and how the agent should contact the group. Application specific behavior is designed and implemented the same as in OBAA agents using goals, role behaviors, and capabilities.

Specifically, OBAA^{++} was designed to enable complex systems that are highly:

1. Autonomous. Agents must be able to operate autonomously.

2. Social. Agents must be able to communicate with affiliated agents.

3. Secure. Interaction with outside groups is limited to the associated persona, allowing the isolation and implementation of continuous authentication, authorization, and validation measures to help provide a defense against malicious or faulty behavior.

4. Flexible. Agents must implement the communication protocols, exchanges, security measures, and other practices required by the systems they implement.

5. Extensible. Agents must provide a way to easily modify and extend functionality.

6. Scalable. Agents must be designed for significant scalability, both in complexity and in the number of distributed elements managed within a single complex system.

7. Intelligent. Agents can be both reactive and proactive, responding intelligently to their environment, employing reasoning, and providing advanced self-control to manage potentially-conflicting goals issued from multiple stakeholders.

OBAA^{++} treats each agent as an integrated group of sub-agents called persona. In this architecture, a single agent may simultaneously execute its own internal goal-driven behaviors, while also participating as a member of one or more affiliated organizations. Multigroup agents may create and manage multiple dynamically-generated *affiliated groups*, to achieve the overall objectives of a complex, multigroup MAS.

5.5 Inner Organization of Persona

Successful companies, like many organizations, tend to begin with a strategic plan first, long before hiring begins. In a similar manner, the organizations that our agents will form and participate in are originally designed by specifying the goals for the organization.

For consistency, every goal specification in a multigroup simulation begins with a single top goal to succeed at the full set of goals the organization attempts to accomplish.

The top goal is progressively broken down through an iterative approach. Typically, the organization goal specification begins with a few simple high-level goals for each major objective required.

The agent itself is designed as an organization of sub-agents as shown in Figure 5.1. OBAA^{++} assists the designer by providing some key communication and control features as reusable components, sub-agents, and capabilities. Agents can be designed, implemented, and tested individually by supplying appropriate configuration information including the list of affiliated groups, whether the agent is responsible for initially running any of those groups, and how the agent should contact and begin participating in the group. Application-specific behavior is designed and implemented as in OBAA^{++} using goals, role behaviors, and capabilities.

The goal of the new architecture is to allow a single agent to participate in multiple groups, or organizations, simultaneously.

OBAA^{++} employs a standard approach to defining persona that corresponds to the system decomposition performed during design. Each agent has a unique *self persona* that acts as the master of the organization. Upon instantiation of the agent, the self persona is created and given a set of goals for the agent including standard goals such as discovering, joining, and maintaining membership in the agents affiliated groups. Each agent also has one persona for each of its affiliated groups, and may have one or more worker persona. OBAA^{++} agents have three *persona types* as described in the following sections.

5.6 Self Persona

The lone *self persona* in each agent is equipped with a *Self Control Capability* that enables reasoning about the detection and management of potentially conflicting goals.

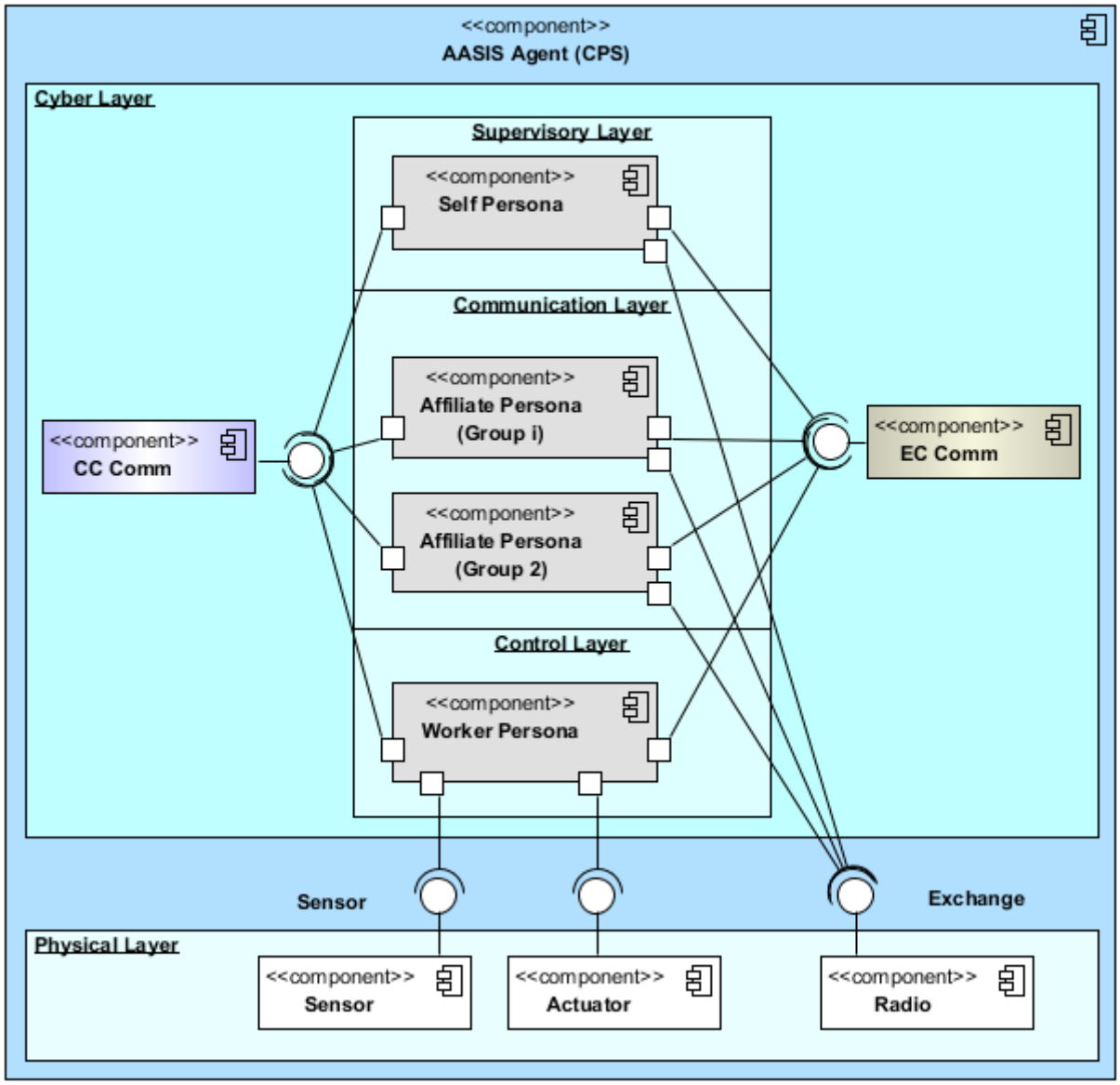


Figure 5.1: *An OBAA^{++} multigroup agent operates an organization of sub-agents called persona.*

5.7 Affiliate Persona

Affiliate persona are responsible for relating with other agents in the context of an external group. The architecture provides a basic set of functionality associated with registering inner participants, providing intra-agent communication, issuing agent goal assignments, and responding to internal events, are provided with the underlying multigroup agent architecture.

5.8 Worker Persona

Worker persona operate much like independent intelligent agents. They may be equipped with both equipment and processing capabilities that enable them to function when disconnected from the larger group. Where appropriate, they are equipped with: (a) sensor capabilities, (b) actuator capabilities, and (c) core processing capabilities. This provides a single set of secure access points to operate associated capabilities. If requests for sensor readings or control actions come from affiliated organizations, the requests can be validated against the installed checks, monitoring, limits, and security requirements encoded in the core equipment access capabilities. All other types can either use these capabilities in their plans, or can send requests to the workers to execute the capability on behalf of an organization.

5.9 Communication

OBAA^{++} multigroup agents communicate along two dimensions: externally and internally as shown in Figure 5.2, External organizations are oriented horizontally (encompassed by dashed rectangles) while internal agent organizations are oriented vertically (encompassed by solid rectangles). For clarity, the self persona of each agent is omitted.

As shown, each persona communicates internally with other persona as well as exter-

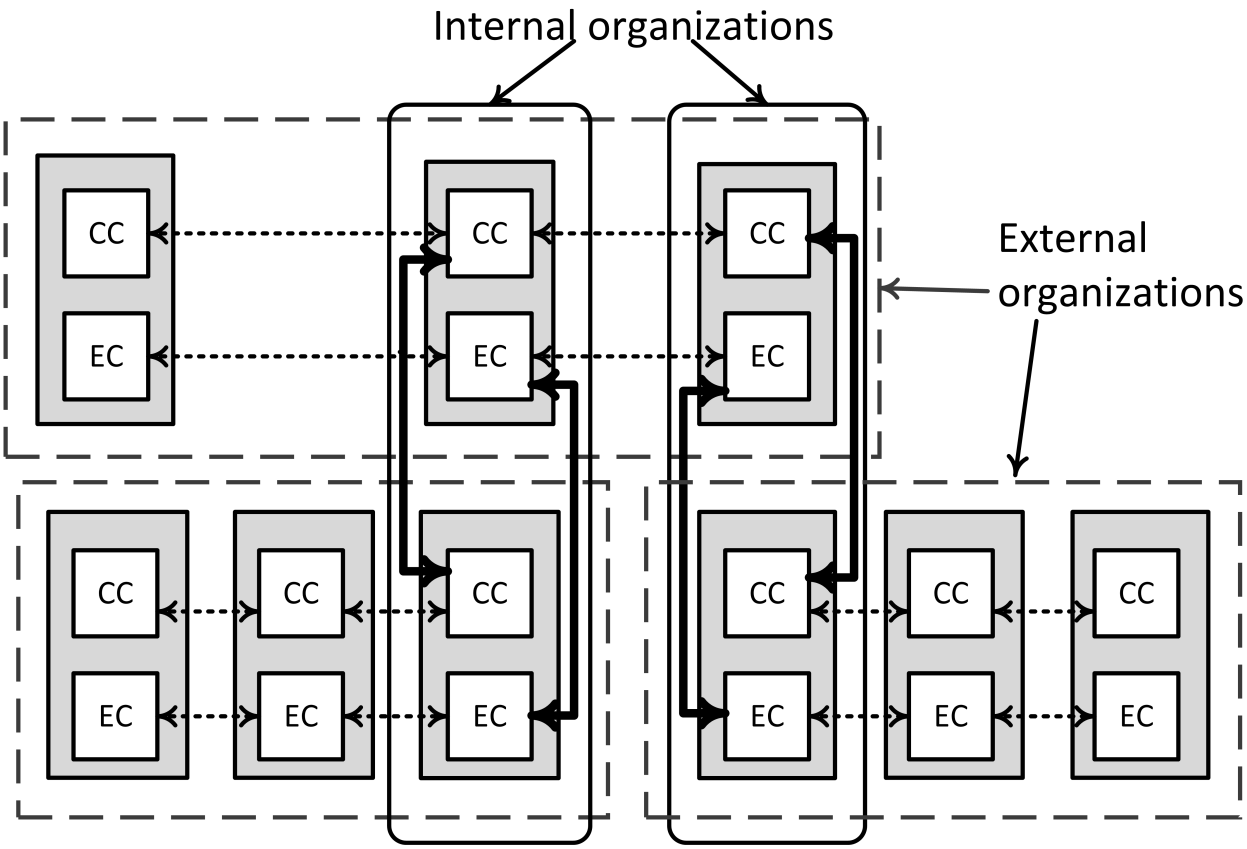


Figure 5.2: *Inter- and intra-agent communications.*

nally with other agents in its affiliated group. Within an agent, persona communicate by passing messages via two internal communication queues, which are shown as bold arrows vertically within internal organizations. The persona CCs communicate directly via the CC communication queue, while persona ECs communicate with each other via the EC communication queue. These two queues correspond to organizational communications and application specific communications. The OBAA communication between the EC and CC within each persona allows application specific information to be transmitted to the CC and organizational information to be transmitted to the EC.

Authentication and authorization protocols can be employed via specially-designed endowed capabilities to meet the security requirements of each different system.

Each system can employ custom communication protocols for connecting, registering, and operating the system. Affiliated agents can communicate over private channels as

needed to support the distributed algorithms and processes required.

Within an agent, the CCs are used to create the organization-specific behavior. Only the self persona issues goals to itself and the other inner persona. All other inner persona accept their goals from the self persona. All other organizational behavior is implemented within the ECs of the inner persona. An affiliate persona may be issued a goal to administer an organization. In this event, it will implement a domain-specific plan to attempt to connect to all participants, initialize a new executable goal model, register agents, and begin issuing organization-specific assignments to external agents.

5.10 Summary

This chapter introduces the new OBAA^{++} agent architecture designed specifically for complex, multigroup MAS. Each OBAA^{++} agent is a MAS of subagents called persona, and each persona is built on the basic agent foundation described in Chapter 4. It shows how this new view of an agent, along with the three standardized types of persona, *self persona*, *worker persona*, and *affiliate persona*, provide a flexible, reusable foundation for implementing complex MAS.

In the next chapter, a new framework and system architecture, the *Adaptive Architecture for Systems of Intelligent Systems* (AASIS) is introduced that uses multigroup OBAA^{++} agents to build complex systems.

Chapter 6

AASIS: Framework for Complex Systems

Pull a thread here and you'll find
it's attached to the rest of the world.
— Nadeem Aslam[157]

Special functionality is required to support participation in complex, multigroup multiagent systems (MAS). A new approach was standardized and is introduced as the new *Adaptive Architecture for Systems of Intelligent Systems* (AASIS). AASIS provides design and implementation guidelines and a supporting set of foundational architectural features to use OBAA^{++} agents to support complex, multigroup participation in a reusable and flexible way.

Section 6.1 and Section 6.2 introduce the new AASIS framework and its layered architecture. Section 6.3 describes the process for decomposing complex systems while Section 6.4 and Section 6.5 introduce the idea of local groups and affiliates. Section 6.6 describes the process for working with multiple organization models in a complex application. Section 6.7

defines the information needed to specify organizations in AASIS. Section 6.8 describes standard goal that govern the high-level behavior of multigroup agents and their customization with goal parameters called guidelines. Section 6.9 introduces standard control mechanisms, while Section 6.10 describes the standard plan types executed by multigroup agents. Section 6.11 describes standard capability types for multigroup agents, including both *innate* and *endowed* capabilities and suggests recommended practices for implementing additional agent capabilities. Section 6.12 shows how the process can be applied to holonic organizations, as a special type of multigroup MAS. Section 6.13 introduces mechanisms for managing consistency and bias and Section 6.14 provides a chapter summary.

6.1 AASIS Framework

The Adaptive Architecture for Systems of Intelligent Systems (AASIS) was specifically designed to provide a flexible, reusable framework for implementing complex systems. It includes mechanisms and recommendations for decomposing complex systems, specifying their behavior using an organization-based approach, and implementing the systems with multigroup agents.

The framework uses the new Organization-based Agent Architecture (OBAA^{++})[14] designed to support reactive and proactive organization-based agents.

6.2 Layered Architecture

An illustration of the AASIS layered architecture for implementing a complex cyber-physical system using the AASIS framework is shown in Figure 6.1.

There are two main layers – first, the cyber layer than sits on top of of the physical layer. Then the cyber layer is further divided into three layers: (1) supervisory layer, (2) an intermediate communications layer for participating in affiliated groups, and (3) a working layer for interfacing with the sensors and actuators associated with devices in the physical

layer. The approach is similar to the ADACOR holonic control architecture[17] shown in Figure 3.9, but in AASIS, the top two levels are combined into a single supervisory layer.

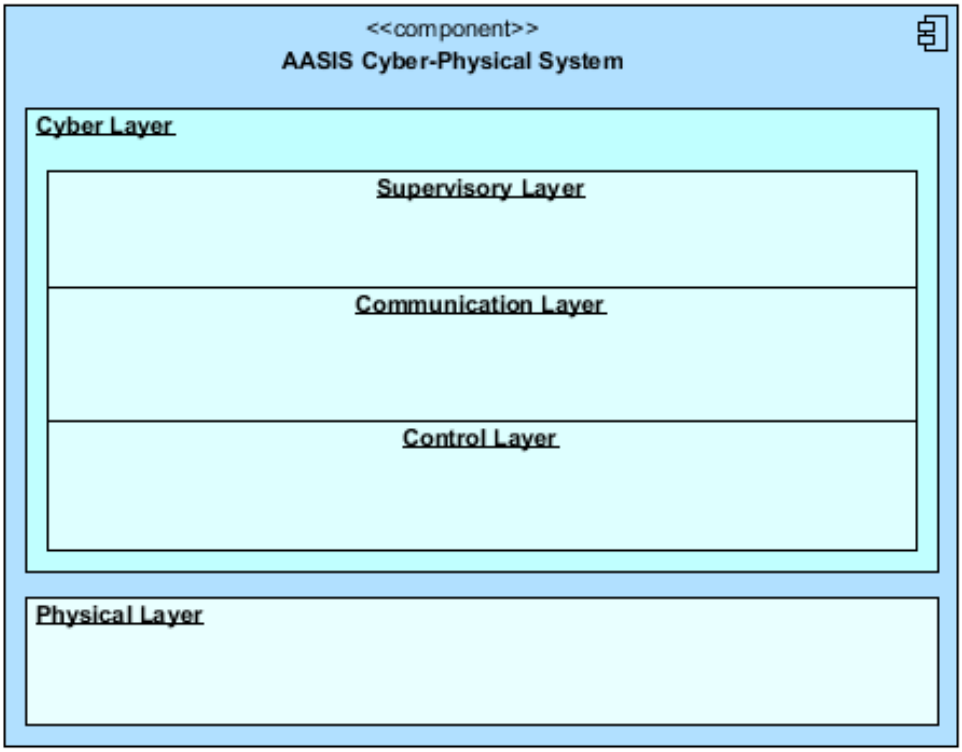


Figure 6.1: *AASIS employs a layered design approach. The cyber layer includes supervisory, communication, and control layers which sit above the physical layer, containing sensors, actuators, and communication hardware.*

Heads and bodies are employed in holonic organizations, but AASIS also works with more flexible master/slave roles. Current implementations require a default master or head for each organization.

6.3 Decomposing Complex MAS

One way to manage complexity is to decompose a complex system into components. This process is illustrated with an example of a cyber-physical system as it warrants the term complex. For systems that don't include a physical system component, the rest of the

decomposition would remain the same.

The decomposition begins with the definition of CPS as the composition of a cyber-system (software system) and an underlying physical system:

$$S = C \circ P$$

$$\text{where } S = \text{a cyber-physical system}$$

$$C = \text{a cyber-system}$$

$$P = \text{a physical system}$$

In this work, the *composition operator*, $\circ$, is used when decomposing cyber-physical systems, because the *cyber functions*, specific to each system, are applied to the *physical functions* and order matters. See Definition 2.4.

Two specific applications that motivated the desire to develop a means for implementing systems of intelligent systems have been mentioned. Cyber-systems were needed to support two different problem domains in the future power distribution system. Therefore, there is a single physical electrical PDS, common to both systems, designated as P_1 and two envisioned cyber systems: (1) a system to support power quality and voltage control, C_1, and (2) a system to conduct online auctions, C_2.

A PDS in a residential area is shown on the far right hand side of Figure 6.2. The physical PDS carries electricity in a hierarchical manner from the single substation (level 1), out into a series of 3-phase feeder lines (level 2), which branch into three sets of single-phase lateral lines (level 3), out to neighborhood transformers (level 4) before flowing to home consumers (level 5). Homes appear as part of the most distributed organizations, the neighborhoods.

The first cyber-system, C_1, involves power quality throughout the distribution system from the substation (level 1) down to individual homes (level 5). The second cyber-system, C_2, begins where neighborhoods are joined under a lateral line (level 3) and does not operate at either the feeder or substation levels. There may be many such power exchanges,

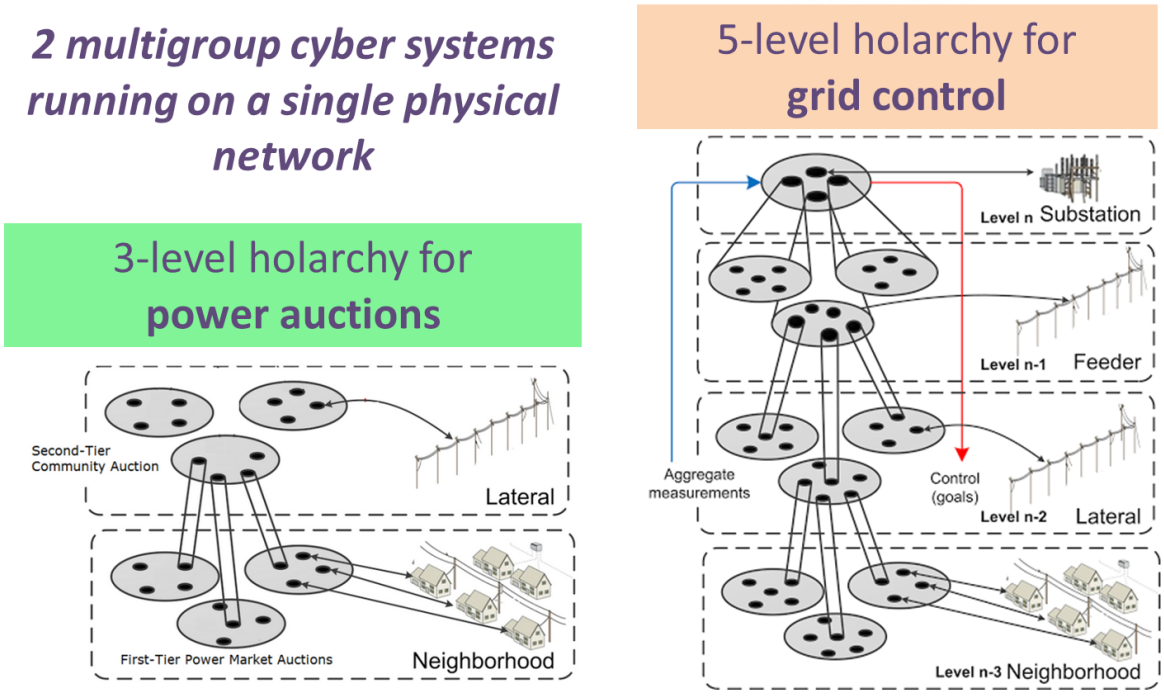


Figure 6.2: *Two complex multiagent systems running on a common physical power distribution system.*

each of which begins at the lowest level with a group of homes (level 5), their associated neighborhood transformers (level 4), several of which are consolidated under a lateral line (level 3).

Two CPS can be built on the same PDS:

$$S_1 = C_1 \circ P_1 \text{ and}$$

$$S_2 = C_2 \circ P_1$$

where S_1 = CPS for power quality control

S_2 = CPS for online auctions

C_1 = power quality cyber-system

C_2 = online auction cyber-system

P_1 = power distribution system (PDS)

However, the two systems may not be completely independent. Decisions about participating in online auctions may impact the flexibility for managing power quality and decisions for managing power quality may impact the amount of distributed generation (DG) available for online auctions. Although only C_1 will issue control actions in P_1, both C_1 and C_2 will access P_1 sensors. Both C_1 and C_2 could employ forecasts predicting weather and expected electrical production and consumption. Mechanisms are needed to integrate the two systems along with mechanisms to allow them to employ common functionality required in both systems.

The selected approach supports common access to underlying physical devices, includes the functionality of both CPS S_1 and S_2, as well as additional functionality for integrating the two systems, enabling a mechanism for conflict detection and resolution (as well as customization) to resolve potential conflicts between the two CPS. The complex IPDS was designed as a single integrated CPS to carry out both the power quality goals and the online

auction goals as follows:

$$S_0 = (C_0 \cup C_1 \cup C_2) \circ P_1$$

where S_0 = CPS for IPDS

C_0 = common and higher integration cyber-system

C_1 = power quality cyber-system

C_2 = online auction cyber-system

P_1 = physical PDS

When describing complex cyber-systems, the *union operator*, $\cup$ is used. Cyber-systems involve a variety of components which may work together, be reused, and be shared among the various cyber systems. The greater *set of cyber functions* that includes the union of all interacting functionality is applied to the *physical functions*. Order of the supporting cyber or software systems may not be clearly defined. The union operator is commutative; $C_1 \cup C_2 = C_2 \cup C_1$.

This chapter includes the description of two different complex MAS, C_1 and C_2. Implementation of both in a complex CPS is described in Chapter 8.

6.3.1 Two Types of Complex MAS

Following the hierarchical nature of PDS, both S_1 and S_2 were implemented as a multi-level *hierarchic holonic multiagent system* (HHMAS). An HHMAS is a type of MAS where the system can be decomposed hierarchically into a system of nested agents called *holons*[158]. Each holon may manage and represent an entire lower-level organization while acting as a participant in an organization higher up the control hierarchy. Holonic design enables the reuse of control logic at each level and provides a means for propagating multiple distributed local optimizations up the hierarchy (called a *holarchy* in the HHMAS) to support increas-

ingly centralized control objectives. The two holarchies, one for online auctions and one for power quality control are shown in Figure 6.2. In this figure, each holon appears as an oval. The C_1 holarchy is shown towards the right of Figure 6.2. There is a single holon at the substation level (level 1), with multiple feeder holons (level 2), each of which as multiple lateral holons (level 3), which in turn has multiple transformer holons (level 4), which are the final level of organizations, containing home holons (level 5), which are not currently subdivided further. The C_2 holarchy is shown on the left of Figure 6.2. The multiple ovals reflect multiple holons starting with lateral holons (level 3), containing transformer holons (level 4), and home holons (level 5).

In organization-based MAS, each organization (or group) of agents interact to achieve a set of organization goals. A complex organizational structure such as a holarchy can be viewed as having many groups. Agents can act to achieve assigned goals and in response to communications received from other agents. In the goal processing system used, assigned goals can be modified by updating custom goal parameters (e.g., a goal to manage reactive power with a smart inverter can be modified to increase the amount of additional reactive power requested). The requirements for the two complex MAS were different, providing several good tests for the flexibility of the proposed system architecture.

Characteristics of the two complex MAS are compared in Table 6.1. In addition, work is being done on a third complex MAS for state estimation where full instrumentation is unavailable[159].

6.3.2 Specifying Multigroup HHMAS

Both complex MAS are implemented as an HHMAS. Several approaches to implementing HHMAS have been proposed. Our approach implements each holon in the holarchy as a goal-driven organization.

Each holonic organization is considered to exist at the more *centrally located* level, that is the higher level (with a lower number). Thus, C_1 has a single holonic organization operating

Aspect	C_1	C_2
domain	power quality	online auctions
direction	power company	local market
goals	continuous	discrete
action focus	immediate	future
holarchies	single	multiple
levels/holarchy	5	3
up flow	messaging	messaging
down flow	goal params	messaging
solution	non-linear	linear
environment	dynamic	static
power focus	reactive	real

Table 6.1: *Characteristics of two selected complex multiagent systems.*

at level 1 while C_2 has none. Both have several holonic organizations operating at the lateral level 3. The set of hierarchic holonic organizations in the C_1 holarchy and the C_2 holarchy, O_{C_1} and O_{C_2}, respectively, are defined as follows:

$$O_{C_1} = O_{C_1,1} \cup O_{C_1,2} \cup O_{C_1,3} \cup O_{C_1,4}$$

$$O_{C_2} = O_{C_2,3} \cup O_{C_2,4}$$

where

$O_{k,l}$ = the set of level l organizations of cyber-system k and

$O_{k,l} = \{o_{k,l,1}, o_{k,l,2}...o_{k,l,n}\}$ where

$n \in \mathbb{Z}$

n = the number of organizations in level l of cyber-system k

When describing complex organizations, the *union operator*, $\cup$ is used to combine sets of organizations. Organizations involve a variety of components which may work together, and be part of multiple organizations at the same time. The respective cyber system includes the greater *set of organizations* that includes the union of all organizations in that cyber system. The order of processing the supporting organizations may not be clearly defined and may

change over time depending on the goals and characteristics of the cyber system and the state of the environment. The union operator is commutative; $O_{C_1,1} \cup O_{C_1,2} = O_{C_1,2} \cup O_{C_1,1}$. When individual organizations, e.g., $o_{k,l,1}$ $o_{k,l,2}$ form a set, they are generally defined as an ordered tuple.

This reflects the set of all holons at each level, but does not indicate their holonic location, such as which level 3 lateral holon contains a specific level 4 transformer holon. To specify the equivalent of a given holon such as the first online auction level 3 lateral holon, $o_{C_2,3,1}$, their location in the hierarchy must be included as shown in the following:

$$o_{k,l,1} = h_{k,l,1} \cup \{o_{k,l,1,1}, o_{k,l,1,2} ... o_{k,l,1,m}\} \cup \{a_{k,l,1,1}, a_{k,l,1,2} ... a_{k,l,1,x}\}$$

where

$o_{k,l,1}$ = a holon in level l of system k

$h_{k,l,1}$ = the head of $o_{k,l,1}$

$o_{k,l,1,i}$ = body holon in $o_{k,l,1}$

$a_{k,l,1,j}$ = non-holonic agent in $o_{k,l,1}$

$m, x \in \mathbb{Z}$

m = the number of level $l+1$ body holons in $o_{k,l,1}$

x = the number of level l non-holonic agents in $o_{k,l,1}$

This reusable approach is flexible enough to specify the different types of holarchies required. Each holonic organization employed a common level-specific goal model and role model to drive the desired behavior. A set of level-specific goal models was developed for C_1 and another, independent set was developed for C_2. This allowed the models and behavior specifications for the two systems to evolve independently, reducing undesired coupling in an already complex system.

6.4 Local Groups

Organizations in MAS can be given a very specific, defined meaning. For simplicity, rather than sub-organization, the term *local group* to define an organization within a complex organization. In a complex hierarchical organization, there may be one local group acting at the top of the hierarchy, several local groups operating at various intermediate levels, and still more local groups operating the lowest levels. Each local group has *local goals*.

These aspects are addressed by the AASIS framework as will be described in the following sections.

6.5 Affiliates

The term *affiliated entities* or *affiliates* are entities that could – or are – acting within a local group. The entities are said to be affiliated with that local group. In intelligent, goal-driven systems, suitably-equipped goal-driven entities may be assigned one or more of the local group's local goals.

Affiliate persona are responsible for relating with other agents in the context of an external group. In AASIS, they are equipped with: (a) a *Connect Capability* that provides the authorization and authentication protocols necessary to contact another agent in the group and (b) a *Communication Capability* that provides the ability to compose, encode, send, receive, and decode group messages. *Group Participants* are equipped with a group-specific *Participate Capability* enabling them to register with the group and receive goal assignments from the group. The default *Group Administrator* is equipped with the group-specific *Administer Capability*. One default administrator is assigned per local group. However, in the event the default administrator becomes incapacitated or suspect, a participant could be elevated to serve as a new or interim administer. The capabilities could be serialized and sent or switched on when a new administrator is required.

6.6 Multiple Organization Models

By definition, a complex MAS (Definition 2.16) includes two or more organizations, which are called *local groups*. Each local group includes a set of local goals as shown in Figure 2.6.

As described in Section 3.6, there are many approaches to designing, specifying, and implementing MAS. This work selected the *Organization Model for Adaptive Complex Systems* (OMACS) to represent the organization information to the agents[13]. OMACS defines an organization as a tuple o=<G, R, A, C, Θ, P, Σ, oaf, achieves, requires, possesses>.

A single OMACS model is sufficient for any adaptive system operating as a single organization; however, when designing a multigroup MAS, such as holonic systems, there will be an OMACS instance, o_i, for each of the organizations in the complex MAS.

When using this approach, AASIS defines a *multigroup MAS* as a specific type of complex MAS, C, implemented as a complex organization, O_C, represented as an ordered tuple of organizations as shown in Definition 6.1.

Definition 6.1 (Multigroup MAS)**.**

$$C = O_C = \{o_{C,1},\ o_{C,2},\ \ldots\ o_{C,n}\}$$

where

C = a complex (multigroup) MAS

O_C = the set of organizations in C

n = the number of organizations in the multigroup MAS

When individual organizations form a set, they are generally defined as an ordered tuple.

In each organization, $o_{C,i}$, the set of agents can be denoted as A_i. Therefore, the set of multigroup agents in C is defined in AASIS as shown in Definition 6.2.

Definition 6.2 (Multigroup Agent)**.**

$$\mu = \{a \mid \exists a \in A_i,\, A_j \mid a \in o_i \wedge a \in o_j\}$$

where

$$\mu = \text{the set of multigroup agents in } C$$

$$o_n = \text{organization } n$$

$$A_n = \text{set of agents in } o_n$$

$$i \neq j$$

6.7 Specifying an AASIS Organization

Successful companies, like many organizations, tend to begin with a strategic plan first, long before hiring begins. In a similar manner, the organizations that our agents will form and participate in are originally designed by specifying the goals and desired behavior for the organization. Staffing the organization with agents is an implementation aspect, and subject to specifics of the physical system size, scope, and available devices and software available on the system.

The information needed to specify the behavior of an organization in AASIS is shown in Figure 6.3.

This information is developed by following the Adaptive Organization-based Multiagent Systems Engineering (AO-MaSE) process, an O-MaSE-compliant software engineering process described in Chapter 7. The AASIS framework provides reusable types and components that support flexible implementation details to provide for a many types of systems as shown in Chapter 9.

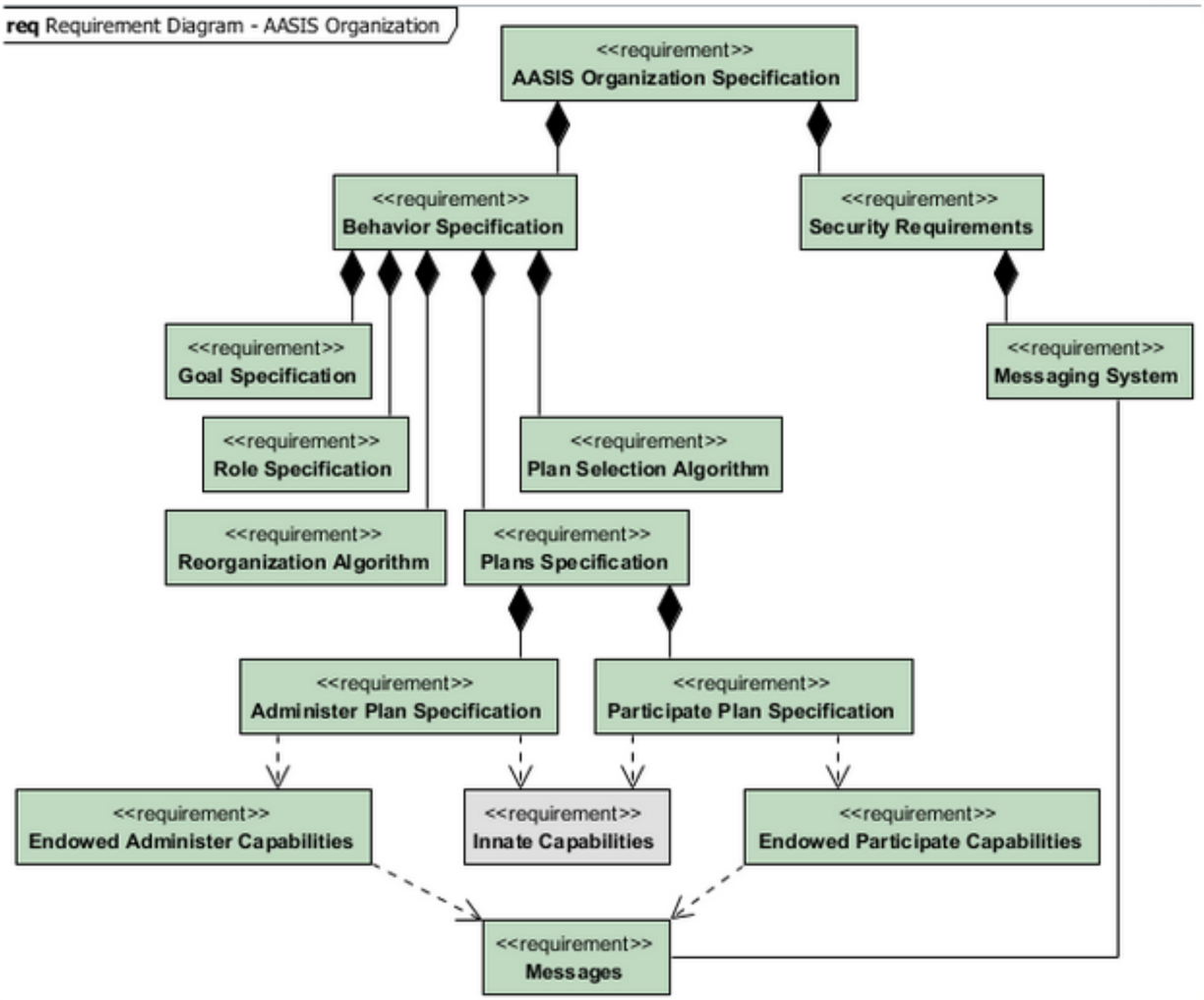


Figure 6.3: *AASIS organization specification requirements.*

6.8 Goals and Guidelines

Every goal specification in a multigroup simulation begins with a single top goal to *Succeed*. The top goal is progressively broken down through an iterative approach. Typically, the organization goal specification begins with a few simple high-level goals for each major objective required.

In the hierarchic holonic implementations, there is one goal specification for each type of organization in a system. There is exactly one type of organization for each holonic level in that system. There may be more than one organization operating at each level and all organizations within that level use the same goal specification, however, the guidelines, or goal parameters, are be different for each organization.

Holonic goals are highly recursive, providing aggregation functions when working up the holarchy and providing methods for distributing goals or power quantities when working down the holarchy. The goals for each HHMAS, designated as $G_1$and G_2 for S_1 and S_2, respectively, can be viewed as the union of the goals of each holonic organization in the HHMAS.

Therefore, the goal decomposition of the goal specification for C_1, which has five levels of agents (1-substation, 2-feeder, 3-lateral, 4-neighborhood, 5-home):

$$G_{C_1} = G_{C_1,1} \cup G_{C_1,2} \cup G_{C_1,3} \cup G_{C_1,4}$$

$$\text{where } G_{C_1} = \text{the goals for the grid control system}$$

$$G_{k,l} = \text{the goals for level } l \text{ of cyber system } k$$

Likewise, the goal decomposition of the goal specification for C_2, which has three levels

of agents (3-lateral, 4-neighborhood, 5-home):

$$G_{C_2} = G_{C_2,3} \cup G_{C_2,4}$$

$$\text{where } G_{C_2} = \text{the goals for the online auction system}$$

$$G_{k,l} = \text{the goals for level } l \text{ of cyber system } k$$

Use of the union operator indicates that the same goal, with the same name may be reused across different organizations. If $g_i \in G_i \wedge g_j \in G_j \wedge g_i = g_j$ then g_i and g_j should refer to the same type of goal[1].

For consistency, the two levels in the online auction system C_2 are numbered the same as the corresponding levels in the grid control system, C_1.

Because multigroup agents are built to be autonomous and to function during periods of disconnect, goal models are developed for even the lowest level of participants, in this case homes, even though they may not run organizations themselves. This design decision will be discussed further in Chapter 8, when the organizations are implemented with multigroup agents.

The goal specification for each multigroup system was constructed independently with the standard *Goal Specification Builder* process described below.

AASIS is designed so that evolution of one system does not impact the evolution of a different system. Development of both the grid control system and the online auction system follow their own, independent, iterative AO-MaSE development process. Changes to either system naturally affect the self control process in a participating agent, as reflected in the standard process developed for specifying goals as shown in the Goal Specification Builder process for multigroup systems shown below.

[1]Different instances of the goals will be created in the respective organizations when the GMoDS for the organization is initialized or updated during execution.

Algorithm 3 Goal Specification Process for Multigroup MAS

```
1: procedure GoalSpecificationProcess(A, S)
2:     BuildSystemGoalSpecifications(A, S)
3:     UpdateAgentGoalSpecifications(A, S)
```

Algorithm 4 Multigroup Goal Builder

```
1: procedure BuildSystemGoalSpecifications(A, S)
2:
3:     for each level type, l in S do
4:         m = new GoalModel
5:         if S_l ≠ S then
6:             ag = new AdministrateGoal(S,l)
7:             ag.addParameter(S.ChildConnections)
8:             ag.addParameter(S.LevelGuidelines(l))
9:             m.add(ag)
10:        if S_l ≠ S then
11:            pg = new ParticipateGoal(S,l)
12:            pg.addParameter(S.ParentConnection)
13:            pg.addParameter(S.LevelGuidelines(l))
14:            m.add(pg)
15:        m.write()
```

6.9 Standard Control Mechanisms

AASIS employs a standard, customizable set of goals, roles, and plans to implement complex MAS with $OBAA^{++}$ multigroup agents[36]. Affiliated agents must first establish a secure connection based on the authorization and authentication requirements of governing MAS. Participants must first establish a connection, a goal typically assigned to the self persona. Once connections are established, or reestablished after a connection has been interrupted, the affiliate persona assigned to the participation goal in that local group will get a goal to attempt to register with the default administrator of the organization. Similarly, the agent that is configured to start operation as the default group administrator will attempt to register all participants listed in the organization specification provided to the default group administrator. The organization specification includes the goal model the administrator will use to create goal instances during run time, the assignment algorithm, and success criteria. The approach is *flexible* in that organizations can customize their desired behavior, but the

Algorithm 5 Multigroup Agent Goal Builder

```
1: procedure UpdateAgentGoalSpecifications(A, S)
2:
3:     for each level type, l in S do
4:         m = new GoalModel
5:         ag = new TopGoal(S)
6:         ag.addParameter(S.ChildConnections)
7:         ag.addParameter(S.LevelGuidelines(l))
8:         m.add(ag)
9:         if S_l ≠ S then                                   ▷ not bottom level
10:            ag = new AdministrateGoal(S,l)
11:            ag.addParameter(S.ChildConnections)
12:            ag.addParameter(S.LevelGuidelines(l))
13:            m.add(ag)
14:        if S_l ≠ S then                                   ▷ not top level
15:            pg = new ParticipateGoal(S,l)
16:            pg.addParameter(S.ParentConnection)
17:            pg.addParameter(S.LevelGuidelines(l+1))
18:            pg.addTrigger(S.AdministrateGoal,
19:                S.TriggerAdminEvent, S.ParentConnection, S.LevelGuidelines(l+1)
20:            pg.addTrigger(S.AdministrateGoal,
21:                S.UnTriggerAdminEvent, S.ParentConnection, S.LevelGuidelines(l+1)
22:            m.add(pg)
23:        m.write()
```

core set of organizational events (such as achieving a goal) and conditions (what it takes to succeed) are standard and reused across all local groups.

6.10 Plan Types

Following the three layers in a cyber-system (see Section 6.2), there are three standard types of plans:

- Self control plans.
- Affiliate plans.
- Autonomous worker plans.

6.10.1 Self Control Plans

Every OBAA^{++} multigroup agent is driven by a *self control* plan. This plan operates for the life of the agent. The plan is used when the self persona is assigned to the Self Control Role to achieve the Self Control goal. This can be customized, and further decomposed as needed reflect the objectives of the associated device, robot, or processing unit. Self Control is a parametrized goal, and can be customized with a set of object or simple parameters to reflect the behavior desired for the agent by its owners.

6.10.2 Affiliate Plans

OBAA^{++} persona that will be assigned to act within affiliate organizations are given plans. The specifics of the plans are dependent on the type of organizational decision-making style the local group is using. For example, if the group is designed to work in a master-slave configuration (as most of ours are), the affiliate plans will be one of two main types, or a hybrid permitting both:

- Organization administration plans. These plans tend to provide functionality similar to the OBAA CC master CC Execution Algorithm (CCEA) described in Section 4.5.2. It includes the ability for an affiliate persona responsible for issuing goal assignments to understand the current state of the organization, understand the active goals, re-arrange, and issue assignments to itself and others in the local organization.

- Organization participation plans. These plans tend to provide functionality similar to the OBAA CC slave CCEA also described in Section 4.5.2. It includes the ability for an affiliate persona to accept an assignment to play a role to achieve a goal in the local organization.

6.10.3 Autonomous Worker Plans

Many OBAA^{++} agents include one or more autonomous worker plans. These plans may operate for the life of the agent. The plan typically includes the necessary plan or plans to utilize sensors, control actuators, and conduct various processing and analytical capabilities associated with the native physical equipment available to the associated device, robot, or processing unit.

6.11 Capability Types

AASIS employs a set of *standard capability types*, some of which are designed to provide the control component (CC) functionality for operating an external, affiliated organization in much the same way that internal persona participate within the agent. While the implementation details of these standard types vary with the specific security and other requirements of the associated system, these types are reusable and expected in any AASIS implementation. Much like OBAA, the CC functionality is supplemented with domain-specific EC capabilities as appropriate.

Additional common capabilities. As a online system, all persona types are typically equipped with (a) a *Synchronization Capability* to provide correct date and time awareness and all but the self persona are given (b) an *Inner Participate Capability* to allow the persona to accept goal assignments from the self.

Capabilities can be designed, implemented, and tested individually. In addition, each agent is given a set of *initialization guidelines* as goal parameters that describe external agent connections, the group in which they will cooperate, and the agent that will act as the default group leader. Additional application-specific behavior is designed and implemented the same as in OBAA agents using goals, role behaviors, and capabilities.

AASIS is designed to support agents participating in different systems concurrently. As such, some capabilities used in the plans may be either *innate* or *endowed*.

6.11.1 Innate Capabilities

Innate capabilities are those that the agent already has, generally related to the devices (i.e., the sensors and actuators), that the agent has been designed to use. Additional innate capabilities may include special learning or analytic capabilities built for the data associated with those devices such as data predictors and forecasters, or data cleanliness or anomaly detection algorithms.

6.11.2 Endowed Capabilities

Endowed capabilities are those specifically developed to support the needs of the organization, including capabilities to connect, register to participate, accept assignments in the organization, issue assignments, or send organizational messages.

6.12 Hierarchic Holonic Organizations

Specifying hierarchic holonic organizations for a complex MAS begins with creating a complex goal model, G. G represents the full set of goals that the system will attempt to achieve and must be decomposed to provide a subset of the goals to each local group to pursue. Similarly, the complex MAS has a set of roles, R, as well as plans and capabilities required to perform the plans.

6.12.1 Holonic Goals

Hierarchic holonic organizations designed in the context of two types of goals: One goal for the *head* or *super holon* administering the collective holon, and one goal for the *body* or *subholon* that participate within the holon. As a recursive organizational structure, the goals for each level of a holarchy should be able to be represented with the same goal model. The original work began with holonic goal specifications customized by the equipment available

at each level, but as work continued, a way to avoid this was developed and a truly holonic, reusable, recursive approach to goal specifications for holarchies was developed. Following O-MaSE, *each local goal model* in the complex goal model, G, undergoes refinement before the process continues.

6.12.2 Holonic Roles

Following O-MaSE, after developing the refined goal model G, the complex role model specification R is developed. Roles allow us to specify the organizational behavior of an organization without requiring knowledge of the actual agents that will participate. That is, roles offer a useful way to allow the specification of how the organization should function to be developed independently of the specifications for each agent.

For each organization, we begin by defining roles to achieve each goal in $G_{k,l}$ where k is the system, l is the level within the system, and G is the goal model for the organization.

The role specification for C_1, which has five levels of agents (1-substation, 2-feeder, 3-lateral, 4-neighborhood, 5-home) includes the set of roles defined at each level:

$$R_{C_1} = R_{C_1,1} \cup R_{C_1,2} \cup R_{C_1,3} \cup R_{C_1,4}$$

$$\text{where } R_{C_1} = \text{the roles for the grid control system}$$

$$R_{k,l} = \text{the roles for level } l \text{ of cyber system } k$$

Likewise, the role specification for C_2, which has three levels of agents (3-lateral, 4-neighborhood, 5-home) includes the set of roles defined at each level:

$$R_{C_2} = R_{C_1,3} \cup R_{C_1,4}$$

$$\text{where } R_{C_2} = \text{the roles for the online auction system}$$

$$R_{k,l} = \text{the roles for level } l \text{ of cyber system } k$$

Use of the union operator indicates that the same role, with the same name may be reused across different organizations. If $r_i \in R_i \wedge r_j \in R_j \wedge r_i = r_j$ then r_i and r_j should refer to the same role.

Algorithm 6 Role Specification Process for Multigroup MAS

```
1: procedure RoleSpecificationProcess(A, S, G)
2:     BuildSystemRoleSpecifications(A, S, G)
3:     UpdateAgentRoleSpecifications(A, S, G)
```

Algorithm 7 Multigroup Role Builder

```
1: procedure BuildSystemRoleSpecifications(A, S)
2:
3:     for each level type, l in S do
4:         m = new RoleModel
5:         if S_l ≠ S then
6:             ar = new AdministrateRole(S,l)
7:             m.add(ar)
8:         if S_l ≠ S then
9:             pr = new ParticipateRole(S,l)
10:            m.add(pr)
11:        m.write()
```

In more complex systems, additional roles to achieve each goal can be added. A mechanism for selecting which role to use must be added to the assignment algorithm used by the control component master for the organization.

6.12.3 Holonic Plans

Following AO-MaSE, after completion of the refined goal model and role model, the plan specifications are developed. Plans provide the details of how a role is to be performed and can be specified by the organization designers without knowing anything about the actual agent who will be selected to perform the plan.

Following the process above, the plan specification for C, P_C, includes the set of plans

Algorithm 8 Multigroup Agent Role Builder

```
1: procedure UpdateAgentRoleSpecifications(A, S)
2:
3:     for each level type, l in S do
4:         m = new RoleModel
5:         if S_l ≠ S then                                   ▷ not bottom level
6:             ar = new AdministrateRole(S,l)
7:             m.add(ar)
8:         if S_l ≠ S then                                   ▷ not top level
9:             pr = new ParticipateRole(S,l)
10:            m.add(pr)
11:        m.write()
```

defined at each level: as follows:

$$P_C = P_{C,1} \cup P_{C,2} \cup P_{C,3} \cup ...P_{C,n-1}$$

where P_C = the plans for system C

n = the number of agent levels in system C

$P_{k,l}$ = the plans for level l of cyber system k

Plans may be quite similar at each level, differing according to the different devices available at each level of the corresponding physical system. Use of the union operator indicates that the same plan, with the same name may be reused across different organizations. If $p_i \in P_i \wedge p_j \in P_j \wedge p_i = p_j$ then p_i and p_j should refer to the same plan.

Plans can be built with the following algorithm.

Algorithm 9 Plan Specification Process for Multigroup MAS

```
1: procedure PlanSpecificationProcess(A, S, G)
2:     BuildSystemPlanSpecifications(A, S, G)
```

In more complex systems, additional plans that perform each role can be added. A mechanism for selecting which plan to use must be added to the plan selection process used by the plan executor.

Algorithm 10 Multigroup Plan Builder

```
1: procedure BuildSystemPlanSpecifications(A, S)
2:
3:     for each level type, l in S do
4:         p = new Plan
5:         if S_l ≠ S then
6:             ap = new AdministratePlan(S,l)
7:             m.add(ap)
8:         if S_l ≠ S then
9:             pp = new ParticipatePlan(S,l)
10:            m.add(pp)
11:        m.write()
```

Plans are implemented by calling methods. These methods are grouped into sets called capabilities. Again, plans and capabilities can be developed to specify desired organization behavior before any agents have been identified to implement the system. This principle of defining the organization behavior fully before staffing the organization with agents is crucial to loosely coupled complex systems defined by multiple participating entities.

The communication layer for a mid-level holon is shown in Figure 6.4.

The communication layer for a leaf-level holon is shown in Figure 6.5.

6.13 Managing Consistency and Bias

In IPDS agent-organizations for the Grid Control System (GCS), there are currently two main drivers: (1) increasing or decreasing reactive power to offset sudden changes in distributed generation and (2) the desire to maintain efficiency by keeping the voltage level from node to node. Voltages should always be keep within hard limits of 0.95 to 1.05 at all nodes. This was used to develop several IPDS examples that motivate the research:

1. A home agent controlling associated PV may have a personal goal to adjust its smart inverter setting to compensate for its home's net change in power from the grid. This goal may mean the agent should decrease reactive power injection. At the same time, the agent could get an updated goal from the neighborhood organization to increase

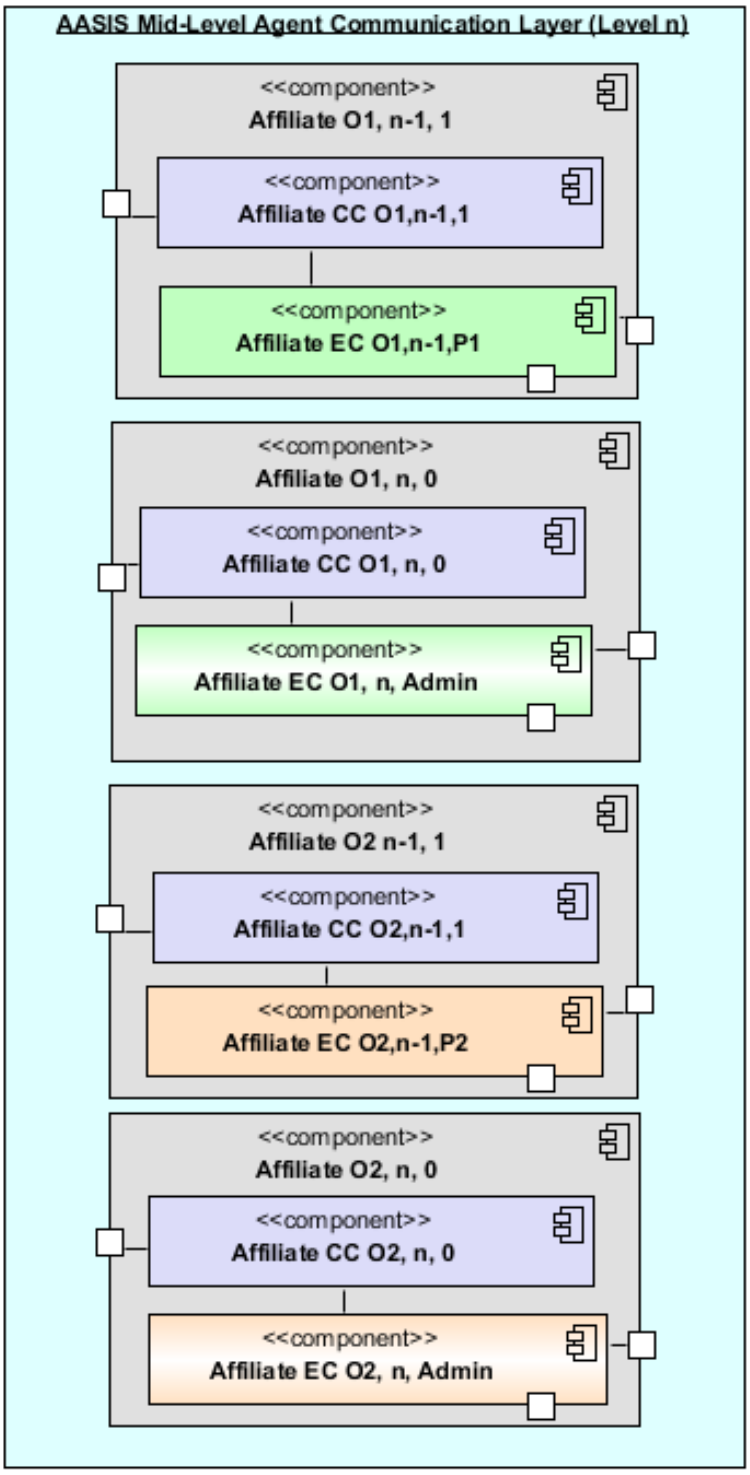


Figure 6.4: *Communication layer for a level n agent in a mid-level holon.*

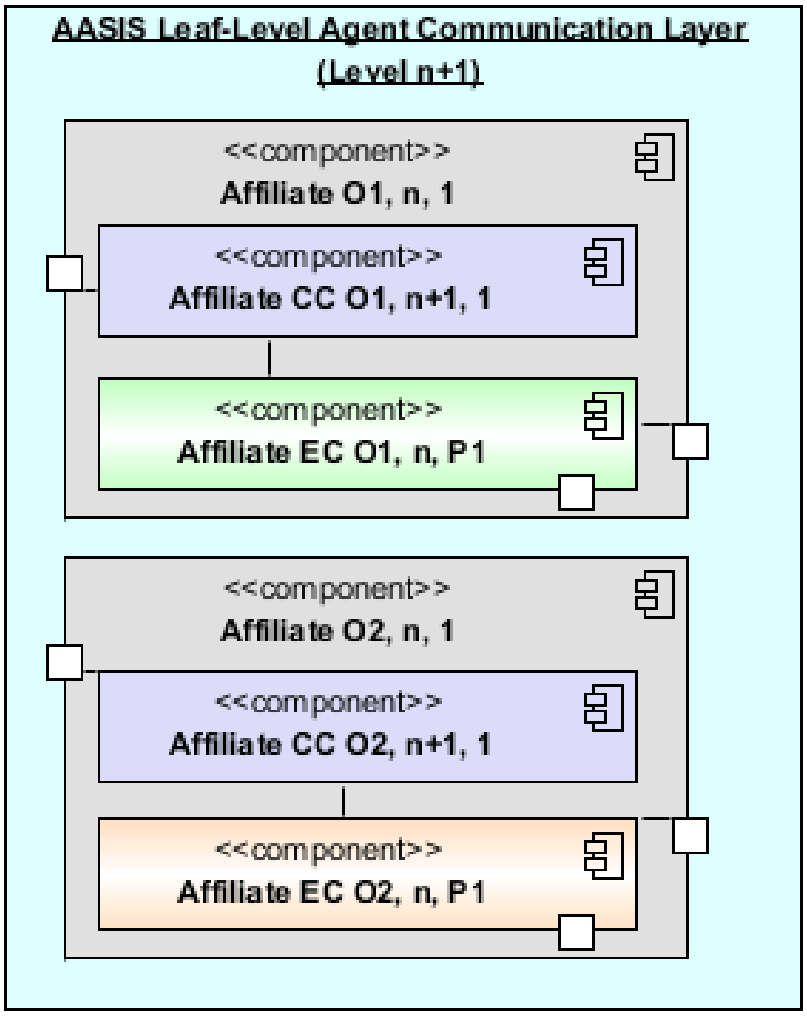


Figure 6.5: *Communication layer for a level n+1 agent in a leaf-level holon.*

reactive power injection due to the combined effects among the neighborhood homes. (Although cloud cover is highly spatially-correlated, it is certainly possible for one home to experience cloud cover while nearby homes are in full sun. Especially as clouds may tend to pass from home to home in rapid succession, this may be a very practical test case.)

2. A home agent controlling an associated PV may also have a personal goal or even a law policy (which may be implemented as a hard goal) to maintain voltage between 0.95 and 1.05. At the same time, the agent could get an updated goal from the local neighborhood organization to increase reactive power, but assisting with this assignment would cause the home voltage to go out of bounds.

3. A neighborhood transformer agent may supervise four homes, several of them with PV. The neighborhood transformer agent has issued the home reactive power setting goals that work best for the local group. As part of its higher-level lateral organization, the neighborhood transformer agent gets a goal to increase the neighborhood reactive power settings. Does this introduce a conflict? If so, how should it be managed?

Mechanisms for *detecting inconsistencies* and supporting the implementation of *stable distributed control algorithms* were needed, possibly with the later application of formal methods to ensure consistent behavior. Specifically, mechanisms for managing goal consistency in multigroup agents, were needed and future work is planned to test these mechanisms in IPDS agents operating in hierarchically- and holonically-structured organizations.

In each test case, the agent should be able to continuously *monitor* its goal assignments, *detect* the existence of possible conflicts, determine whether a conflict exists or not, and if a conflict is detected, enact procedures for appropriately *managing* the conflict[37].

In his 2008 paper on *Future Directions for Agent-Based Software Engineering*, M. Winikoff suggests three research areas are key to the future of the field[93]. The first of these is *goals*. Within this key area, one of the specific questions is:

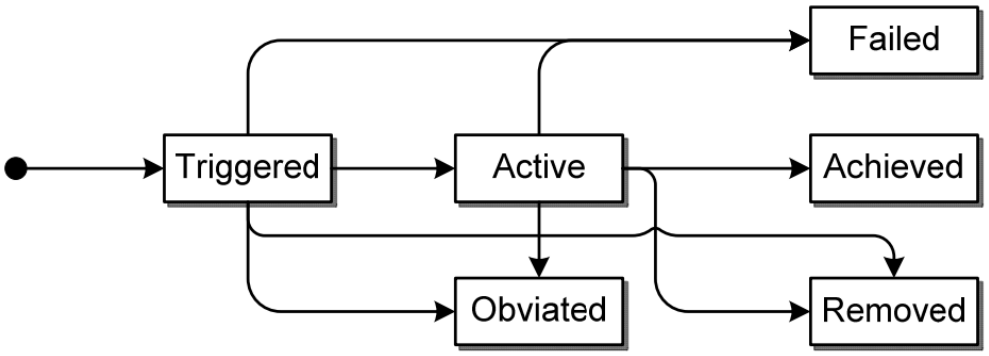


Figure 6.6: *Goal Model for Dynamic Systems (GMoDS) execution model*[14].

What (generic) mechanism can be added to agents to detect interactions between goals, in order to avoid conflict (e.g., resource contention) and exploit positive interactions ("synergy")?

In our applications, each organization – both *inner organizations* (the personal organizations maintained by each agent privately) and *affiliated organizations* (those that are created dynamically by the agents when working together) are driven by a defined set of specification goals. Using GMoDs the goal specification also becomes a working part of the runtime engine, instantiating goals for assignment to participating agent persona. Persona represent the agent in a dynamically-created, affiliated group. Persona are assigned to play a role to achieve a goal. Mechanisms supporting goal consistency are focused on enabling the persona to assess the goals they get from an organization in light of their current assignments and to detect direct conflicts and implement procedures for managing goals among themselves.

The AOSE process and tools used to design agent systems can be crucial. AO-MaSE provides recommendations for engineering complex systems.

There were several good options for selecting an approach. The current execution model in the Goal Model for Dynamic Systems (GMoDS)[91] is shown in Figure 6.6[14]. A triggered goal becomes active unless obviated due to no longer being necessary under current conditions (e.g., a rescue goal may be obviated if the search finds a non-resuscitable target.)

The recent work with Belief-Desire-Intention (BDI) agents proposes a goal life cycle that includes an initial pending state from which the agent may decide to either activate, suspend, or abort the goal. Suspended goals may be sent back to the pending state periodically for reevaluation. Goals in the active or monitoring states may also be sent to the suspended state and from there may return to the pending state. This life-cycle provides interesting insights to constructing a framework to support the monitoring, assessment, and resolution of potential goal conflicts[118].

Starting with an execution model implementation may be helpful, as it would provide a common structure regardless of the specific conflict reasoning and resolution approaches chosen. Triggered organization goals can be terminated before an agent begins by being removed due to negative triggers, obviation, or a guaranteed failure (such as the loss of a critical resource for achieving the goal). Triggered goals not removed may be assigned to an agent, but instead of a triggered goal going directly to the active state, there would be three assigned states: (1) a mandatory first stop for *assessing* and reviewing the goal for possible conflicts prior to adoption, which may include negotiation or returning a goal found to be inconsistent (2) *holding*, a temporary wait which could be revisited either due to a specific event or reviewed on a schedule and (3) *achieving*, the process of actively executing the associated plan. Should the running goal model for the organization decide that an assigned goal instance has been removed, obviated, or failed, it can notify the agent assigned, who will stop assessing, release the hold, or stop executing the plan, as appropriate.

At the highest level, there may be two opposing approaches to managing goal consistency:

- The first would be to provide a perfect life, to design the system so no inconsistencies can be introduced and to enforce rigorous evaluation prior to goal creation that would not allow the creation of goals that conflict.

- The second is to permit the creation of goals that may be incompatible and assume that detecting and resolving conflicts in cooperative systems is part of natural and artificial intelligence.

As organizations, multigroup agents can employ a variety of organizational mechanisms and supporting data structures including policies to guide the agent's monitoring, detection, and resolution of goal conflicts. Rules, however, may simultaneously both help to address organizational needs, but they may also inherently *contradict* organizational needs[160].

In summary, were are several possible ways to proceed:

1. Develop something similar to the Thangarajh and Winikoff method based on existing parametrized goals. This would provide an approach for non-BDI agents built on the existing GMoDS goal model.

2. Develop something similar to the Dufree method using a hierarchical behavior and space search algorithm. Goals from higher organizations would obviate any existing, related goals created by lower level organizations, and only law policies, reflecting required or inherent equipment limits, would limit the adoption of goals as directed down the hierarchy. This top-down command-and-control approach is possibly inherently more consistent, but also less flexible, and could negate some of the benefits of having intelligent, communicating, cooperative agents working out local optimal solutions.

3. Employ rigorous category theory to address the issue at a deeper level.

An approach similar to the first option was selected. The work focused on developing and testing mechanisms to address to the following specific cases of detection, consistency, and response.

- Detection of direct conflict.
- Detection of potentially antagonistic goals.
- Evaluation and internal management of antagonistic goals.
- Full or partial acceptance of non-conflicting, parametrized requests.

- Detection and selection of compound plans that support pursuit of one or more goals simultaneously.
- Identification of non-conflicting, non-opposing, unrelated goals and a mechanism for acceptance and execution via request.

6.13.1 Multigroup Agent Architecture

The simulation employed the multigroup agent architecture being developed. The multigroup agent architecture implements agents as groups of subagents and enabled standard mechanisms for reasoning about potential conflicts, establishing and maintaining connections with other agents in affiliated groups, and enabled agents to instantiate and administer affiliated organizations for achieving multiagent objectives.

Standard mechanisms can be incorporated into agent architectures for assessing goal consistency and managing the appropriate evaluation, negotiation, and acceptance of assigned goals before incorporating them into the agent workflow.

In their model[116], there is a resource summary associated with each plan. For us, when a plan gets assigned to a goal, the plan resource summary could effectively become the resource summary for the associated goal. This approach or something similar may be useful in the IPDS project when setting and managing goals associated with managing elastic and inelastic load demands for each home.

6.13.2 Goal Conflict Management

The selected approach to goal consistency primarily focuses on the area where an agent is participating in an affiliated group and gets a goal from the CC master issuing the organization assignments. The CC master assigns the goal to the affiliated CC slave. A persona in an affiliated organization typically provides communication and representation only; significant processing work is performed by a dedicated persona operating within an agent's

Conflict	Work	Study	Eat	Relate	Personal	Sleep	Accept Pay
Only Work		x	x	x	x	x	
Work		x		x	x	x	
Study	x			x	x	x	
Eat						x	
Relate	x	x				x	
Personal	x	x				x	
Sleep	x	x	x	x	x		
Accept Pay							

Figure 6.7: *Simplified goal direct-conflict-detection table.*

private domain. The following definitions may be helpful.

As described earlier, a *direct conflict* is a logical inconsistency. Two goals are in direct conflict when the goal *identifier*, the unparameterized goal type, of one is enough to imply the direct *negation* of another, e.g., when one of the two cases applies:

- A $\Rightarrow \neg$ B.
- B $\Rightarrow \neg$ A.

Given that personal inner organizations can maintain independent and private goal models, and that the identifiers used *between* goal models could possibly have inconstant usage, a direct means of maintaining a hash table of key-value pairs was selected. The key is the unique identifier of a goal that conflicts, and the value is a list of all goal identifiers that are in direct logical conflict with the key goal. For example, if a self agent goal model has a Pay goal and an affiliated group has a Freeze Expenditures goal, the list may be as follows: <Teller.Pay>, <Account.FreezeExpenditures>,
<Account.FreezeExpenditures>, <Teller.Pay>.

Entries are added to the list in both directions as appropriate. Currently, there is only one case where the direct conflicts involve parameters and that is essentially where an all or none are assumed. The current implementation of this mechanism assumes goal identifiers are consistent between all goal models, and the prepending of the optional organizational indicator was not needed. Some logical inconsistencies in the test simulation are noted in

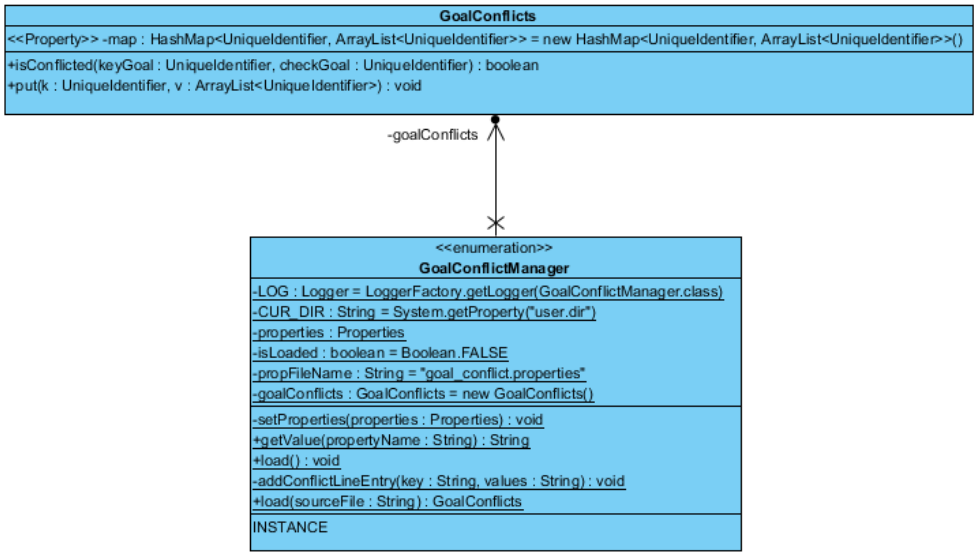


Figure 6.8: *Goal-conflict management support.*

Figure 6.7. Actual implementation prepended all entries with the goal model and used the actual goal type identifier. Some of the associated components related to to Goal Conflict Management are noted in Figure 6.8.

6.13.3 Community Bias Management

The selected approach for enabling reasoning based on the authority of natural hierarchies was generalized to include a configurable community bias. Each agent maintains a community bias indicator on a per goal basis. This bias indicator is currently configured as a simple multiplier, with 1.0 indicating the agent is completely community-focused and will strive to achieve the community goal in full. Similarly a ration of 0.0 would be completely selfish, indicating the agent will do nothing to support the associated goal when issued by an affiliated organization. Some of the components related to *community bias* and *selfishness management* are noted in Figure 6.9.

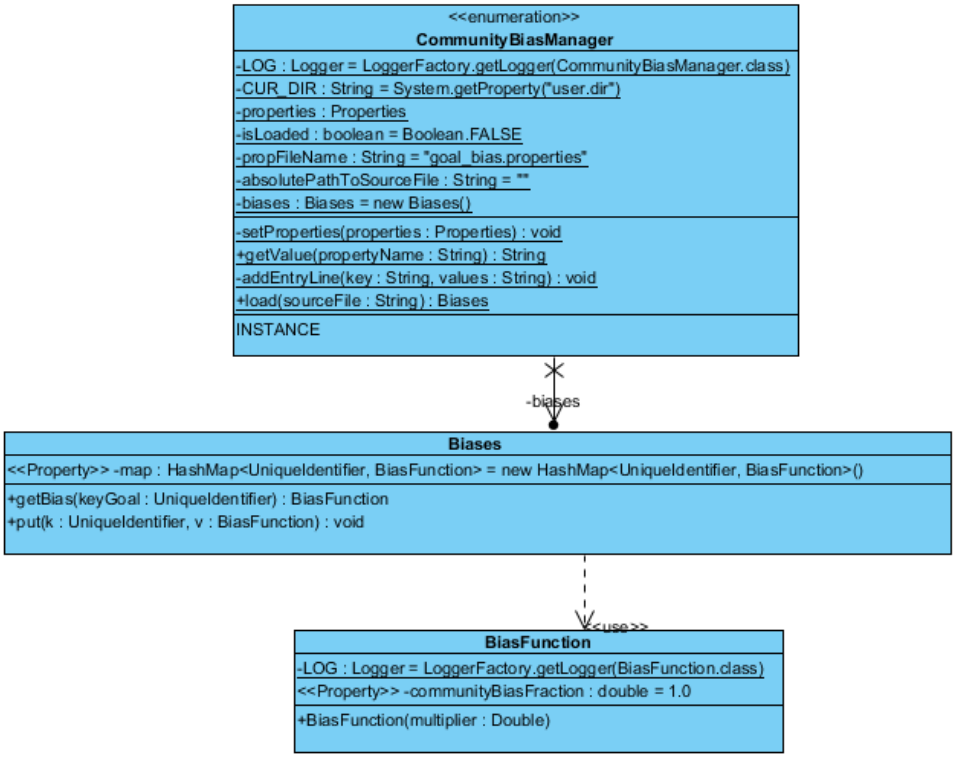


Figure 6.9: *Community bias versus selfishness support.*

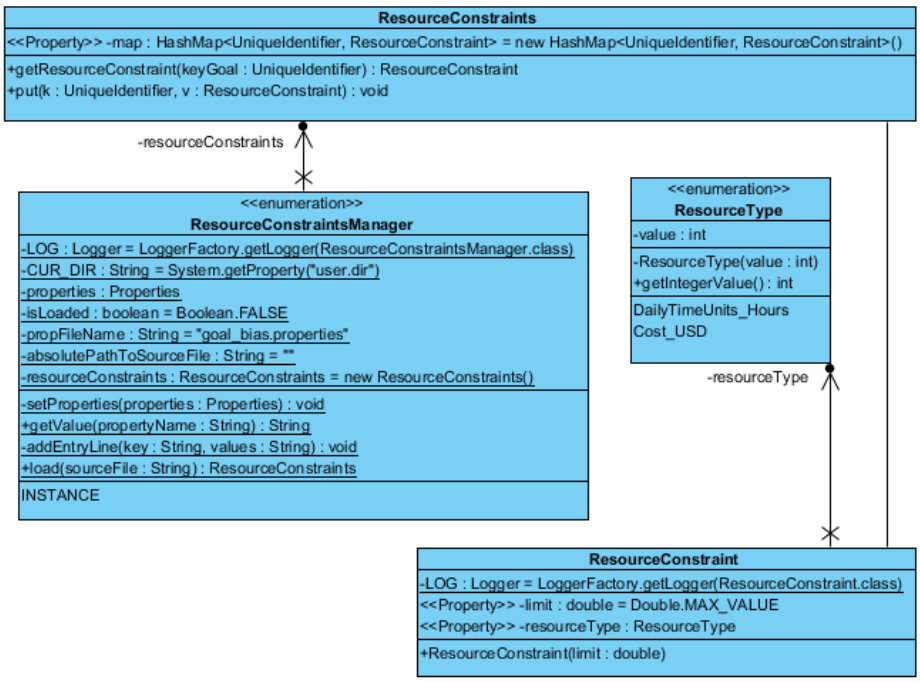


Figure 6.10: *Resource management support.*

6.13.4 Resource Management

Resource management is a critical component of real world reasoning and therefore, this effort included an implementation of resource management mechanisms that can be used to determine soft conflicts, e.g., when two goals each require 7 units of a resource from an agent that only has 5 available. Some of the components related to Resource Management are noted in Figure 6.10.

6.13.5 Reasoning with Utility Functions

Enabling resource management motivated an associated mechanism so that not all resource applications must be assumed to be equal. The project approach included the addition of utility functions to assist an agent that cannot meet the full demands of two assignments and must determine how much each goal should be partially satisfied to maximum the total possible utility. This mechanism will form the foundation from which a variety of distributed

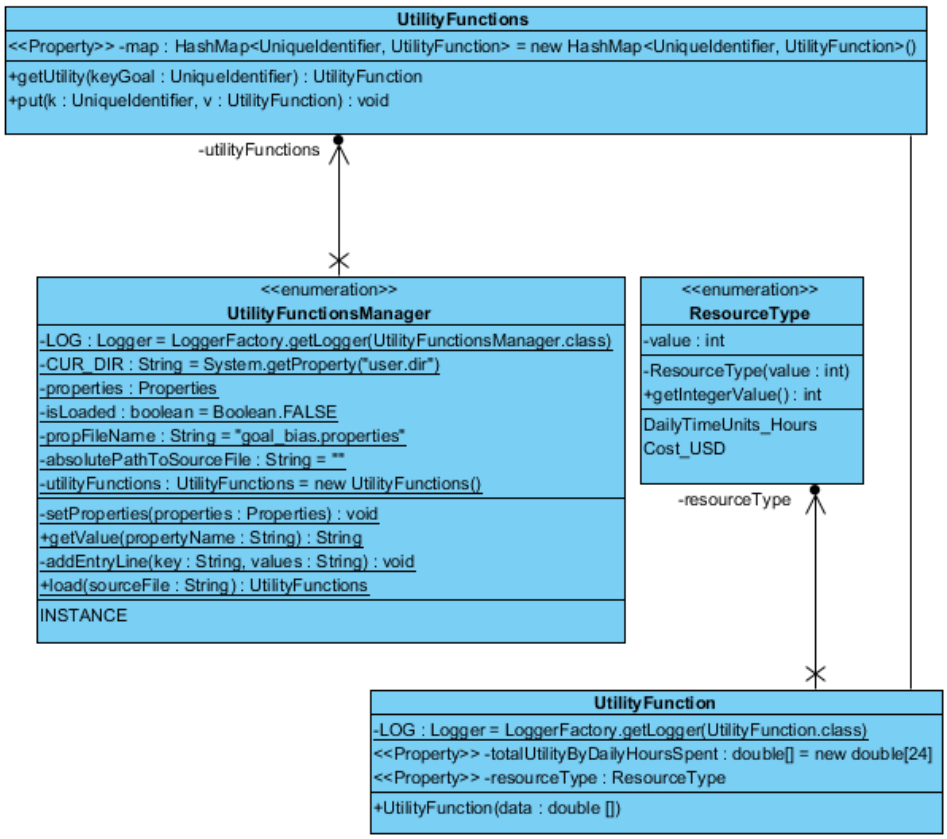


Figure 6.11: *Reasoning with utility functions support.*

constraint satisfaction problems can be resolved and begins to set a core layer from which a library of game theoretic algorithms could be employed and tested. Some of the associated components related to *reasoning with utility functions* are noted in Figure 6.11.

6.14 Summary

This chapter describes the way complex multiagent systems can be implemented in the OBAA^{++} multigroup agent architecture and illustrates how the architecture supports not only multigroup organizations such as holarchies, but how the system can support multiple organizations, operating under different goal-driven behavior specifications concurrently. This AASIS framework can be used to build complex systems by employing the software engineering process described in Chapter 7.

Chapter 7

AO-MaSE: Engineering Complex Systems

If you give someone a program, you will frustrate them for a day;
if you teach them how to program, you will frustrate them for a lifetime.
— *David Leinweber*[161]

This chapter describes the process developed for engineering multigroup agents for complex cooperative systems. Section 7.1 describes the motivation and challenges. Section 7.2 introduces the novel adaptive architecture. Section 7.3 covers the iterative, O-MaSE-compliant process used to build organizations, both inner organizations within a multigroup agent, and external, affiliate organizations the agents create and operate. Section 7.4 describes the process for beginning with the inner organizations of persona and Section 7.5 describes the process for affiliate organizations. Section 7.6 describes software quality benefits and Section 7.8 provides a chapter summary.

7.1 Challenges

Smart infrastructure optimization involves some of the most complex and critical systems in modern society[7]. Agent technology offers a way to manage the inherent complexity of such systems. Agents can be used to represent simple variables in a computer program as well as complex, distributed, intelligent objects involving potentially infinite numbers of states, decisions, and actions and reactions[141]. When modeling power systems, agent traits of particular interest include autonomy, heterogeneity, adaptivity, social ability, communicability, flexibility, and concurrence[162]. Agents implement goal-based behavior and *intelligent power distribution system* (IPDS) agents must demonstrate the ability to support the objectives of their respective owners while also acting cooperatively to achieve common objectives, such as maintaining critical loads and system efficiency.

Power flow, quality, and control lends itself to distributed, recursive optimization where possible. Some local optimization can be distributed and may not result in propagation throughout the hierarchy, while the system as a whole may be impacted by larger, more centralized control options such as load tap changes. Using a flexible, holonic architecture allows us to evaluate a variety of control algorithms and strategies.

7.2 AO-MaSE

The *Organization-based Multiagent Systems Engineering* (O-MaSE) framework provides a foundation supporting tailored software engineering implementations. A compliant process must meet the following requirements: (1) no new constraints may be placed on existing entities and relationships in the O-MaSE metamodel, (2) the method guideline pre-conditions must not become stronger or post-conditions made weaker, and (3) no existing metamodel entities, tasks, work products, or method-roles may be eliminated[9].

Multiagent systems (MAS) and holonic MAS (HMAS) may involve complex systems. Getting started with such frameworks can be challenging[163]. The *Adaptive O-MaSE software*

engineering process (AO-MaSE) provides a set of recommendations for dealing with that complexity by applying some of the principles commonly associated with agile processes[164]. Several of these agile principles are associated with MAS in general and include an ability to respond to changes, an ability to participate in ongoing collaboration, a recognition of the importance of interaction between autonomous participants, and a focus on goal-driven, executable components. In a similar way, the AO-MaSE approach focuses on adaptability and the structured evolution of a working system. Other systems such as PASSI have gone further to incorporate agile processes[165]. Agile PASSI research confirmed that agile processes tend to spend less time on design and correspondingly more in coding and testing and found that a quicker move to implementation was helpful when addressing high-risk areas[165]. In addition, research at the University of Vigo in Spain has adapted the INGENIAS methodology to follow the agile process SCRUM with promising results[166].

The process follows four key strategies:

- Start simply and add incrementally; in the AASIS framework, agents given even simple goals may require a significant infrastructure to execute.
- Apply recommended process conventions to enhance clarity and consistency.
- Follow models with code construction to get working systems early.
- Expand, enhance, and refactor as functionality evolves.

By following the AO-MaSE approach and detailed implementation guidelines, the full set of required components can be implemented early, and form a basis for expanding and enhancing the system. Completing the connections in a working system provides examples of how components connect the various models and drive the behavior of the system. For example, event triggers on the goal model may appear as transitions in the plan diagrams and domain objects may appear in method parameters in plan states and associated capabilities. A working version that connects the parts provides a concrete example for

software engineers and developers that have little experience in agent-oriented engineering. The method construction guidelines provide the ability for a team of software engineers and subject matter experts to work collaboratively to select key elements for implementation and to develop associated work products including requirements specifications, goal models, organization models, domain models, role models, role plans, plan states, capabilities, protocols, policies, and code.

7.3 Iterative Implementation

In AO-MaSE , the architect begins by creating a fully executable but limited-scope vertical spike through the system to create a working version early that offers a solid core from which increasingly complex analyses and behavior can evolve. AO-MaSE follows the O-MaSE compliant process and depicts three iterations that include the tasks and work products shown in Figure 7.1.

This iterative, O-MaSE-compliant process is used to build organizations, both the *inner organizations* within a multigroup agent, and the external, affiliate organizations that the agents create and operate while supporting different systems[39].

7.4 Internal Organizations

In AO-MaSE, the design process begins by defining the inner organizations of persona. For simplicity, each organization can begin with a supervisory level and a worker level that manages the various sensors and actuators the agent can access. This allocation of persona to different layers according to their purpose is somewhat similar in approach to the layered ADACOR implementation[17]. Different types of entities together form a process layer across the system.

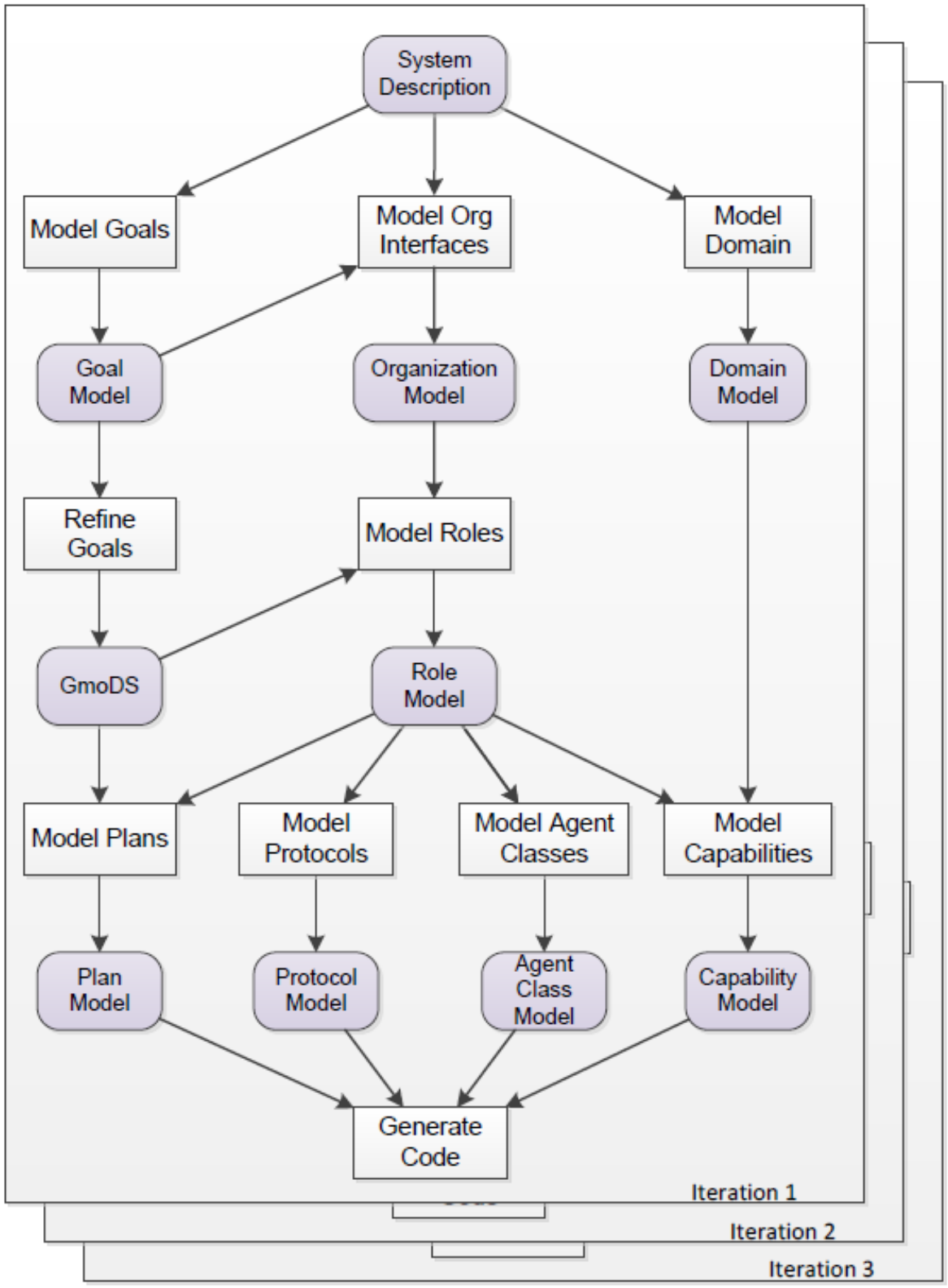


Figure 7.1: *In AO-MASE, each organization develops in a progressive, iterative process. Tasks are shown as rectangles, work products are shown as rounded rectangles.*

7.5 Affiliate Organizations

Independently, systems are designed by determining first, the organizational design for the system. If the system will employ multiple organizations, the system should be decomposed. In some designs, for example those employing hierarchic or holonic aspects, there may be multiple types of organizations. Organizations that differ in goals, roles, or behavior should be identified. In hierarchic structures, for example, there may be a top-level organization, a mid-level organization, and a lowest-level organization type. There may be one type of mid-level organization, regardless of the number of levels between the highest and lowest parts of the system.

The inner and affiliated organizations are integrated by implementing the communication layer with the additional goals, roles, capabilities, and persona to the inner organization models. Capabilities can be reused between independently-specified systems. This can be enhanced by defining good application programming interfaces (API) and following developing standards.

7.6 Software Quality

The project resulted in the development of a clear, repeatable process that sped the implementation of systems with no loss in functionality. The AO-MaSE process and the model-driven tools provide a complete path through the design, specification, implementation, and execution of a MAS. The process was successfully used to test implementation of the required goals, roles, capabilities, and plans required for the initial IEEE test case developed by the electrical engineering simulation team. The AO-MaSE process was employed during the development of an entirely new type of organization that will allow the implementation of additional reasoning for agents participating in multiple organizations. The process and tools provided a clear path that again allowed implementation of the first iteration of end-to-end functionality in a matter of days.

A comparison of O-MaSE compliant systems, some of which were developed using the AO-MaSE guidelines and some which were not, is shown in Table 7.1. For example, for the initial iteration of AO-MaSE, the options are constrained to create a quick implementation that cuts all the way through the process. When developing the initial set of roles that achieve each goal, the AO-MaSE process recommends starting with exactly one role capable of achieving each goal (see the first row entry in Table 7.1). In O-MaSE, no such correspondence is required and the association between goals and roles can be much more complex. This flexibility is available in later AO-MaSE iterations as well; however it is not recommended for the first iteration. Similar constraints are recommended for the initial implementation of AO-MaSE with regard to roles and plans. AO-MaSE recommends starting with one plan for each role. Additional recommendations for the initial AO-MaSE iteration are shown in the remaining rows along with their correspondence to the associated characteristics permitted in the original O-MaSE process (as shown in column 3).

7.7 Application

The AO-MaSE process was developed and implemented during the production of the initial IPDS architecture. AO-MaSE design conventions, recommended practices, and guidelines are described and illustrated.

7.7.1 Iteration 1: Getting Started

In addition to the infrastructure of the component parts, an OBAA-based system offers significant initial agent functionality, but introduces some additional embedded complexity. System behavior develops in response to a variety of events such as goal triggers, agent registrations, and organizational events. Early execution can help software engineers get a better understanding of the system. The first iteration results in a streamlined implementation that provides an early executable model of the system. The following summarizes the

Feature	**AO-MaSE initial iterations**	**O-MaSE final implementation**
Goal – role correspondence.	Direct 1-1 correspondence facilitates initial modeling and subsequent debugging.	No correspondence required; multiple roles may achieve a goal.
Roles – plan correspondence.	Direct 1-1 correspondence facilitates initial modeling and subsequent debugging.	No correspondence required.
Plans and plan state consistency.	Plans initially implemented using automated INIT-EXECUTE-STOP template.	Plans created and refined independently as flexible finite state models.
Post-fix object type naming standards.	Consistent application of suffix object type names (e.g., SmartMeterCapability, ManagePowerGoal) improves code readability and maintainability.	Suffix object type names not required. Less code clarity and increased need for commenting or familiarity for implementation and debugging.
Clearly-defined design process.	Yes. Application of process framework and agentTool3 modeling tools clearly described.	Yes. Application of process framework and agentTool3 modeling tools clearly described.
Clearly-defined implementation process.	Yes. Well-defined and structured implementation process and guidelines provided.	Flexible process for implementation; few direct guidelines.

Table 7.1: *Implementation of O-MaSE-compliant MAS with and without AO-MaSE.*

AO-MaSE recommended practices for an initial iteration.

1. Define one top-level goal to reflect the overall behavior desired by the system; add a small number of terminal goals (without subgoals) to represent core objectives.
2. Define the initial set of interfaces to the overall organization.
3. Define roles to achieve each terminal goal. Where possible, follow parallel, explicit naming conventions that differ only in type.
4. Define plans to perform each role.
5. Define the Plan Selection Algorithm (PSA).
6. Define capabilities specific to each plan; define a local domain-specific communication capability and an external controller communication capability.
7. Assign role requirements. Most roles require control communication (for OBAA agents), the local domain-specific communication, and at least one role-specific capability.
8. Define a limited number of plan states (e.g., INIT, EXECUTE, and STOP).
9. Define plan state transitions and state behaviors by defining and calling capability methods.
10. Define agent types based on the problem domain.
11. Define the Execution Component Execution Algorithms (ECEA).
12. Define the Control Component Execution Algorithms (CCEA).
13. Configure agent instances with associated capabilities and attributes.
14. Configure environment object instances such as sensors and actuators.

15. Configure, implement, debug, test, and execute the initial vertical spike.

The project began with the specification of requirements. From the set of requirements for the initial phase of the project an initial focus was selected that allowed testing core functionality – the distribution and management of goals within a local organization. Since achieving each goal requires substantial infrastructure, goals were limited to a small set of core objectives. The top goal of each recursive organization is Support IPDS, as shown in It will guide agent organizations while connected to the grid and while running in islanded mode.

The organization model was developed to define the boundaries and interfaces for the system. Each IPDS would seek an external controller that would both receive requests and send requests/guidelines down to the system, enabling centralized control and communications from the primary energy supplier. Inputs were provided to characterize the organization's goals. The goal model was drafted and then refined to show the supervisor triggering a manage instance goal for each participant. The domain model began to reflect the objects in the environment and included a smart meter object and a PV system, along with equipment attributes and unique identifiers.

Following the guidelines, a role was created for each terminal goal, a plan for each role, and gave each plan three initial states: (1) INIT for performing actions that will only need to be done once, (2) a role-specific state that captures the main work of the role, and (3) a STOP state consisting of behaviors to be executed when finishing the plan. The recommended capabilities were defined. As plan states were developed in the plan diagrams, the methods required of each capability were identified. Parallel naming conventions for goals, roles, plans, and role-specific default capabilities aided clarity and were used to employ additional code automation. Agent types did not parallel the goal or plan names. Instead, they reflected the physical installation or focus of the agent type. The first class was an Neighborhood Agent class, expected to run on or near a transformer serving 2-6 homes, and a Prosumer agent class, expected to be installed on or near a home-based smart meter.

As the OMACS components developed, they were implemented in the OBAA-based IPDS framework. Agent and Environment configuration files were used to instantiate specific agents and objects for a variety of test cases. The AASIS framework can be employed immediately if one control component master is declared for any local organization. The first iteration began with one supervisor neighborhood agent (the control component master) and two prosumer agents (both control component slaves) to test the ability of the system to solve adapt to changing local conditions.

7.7.2 Iteration 2 – Filling in the Framework

With a working simulation provided during Iteration 1, the focus in Iteration 2 shifted to adding functionality to address a variety of potential challenges. Development focused on enhancing the plan states and capability method calls. Additional capability types were added, providing additional differentiation and room for expanded functionality. Capabilities were implemented with simple algorithms that served to define the expected interfaces that would be required to support more complex optimization algorithms that were being developed in parallel research projects.

The goal model was enhanced to include parameterized goals with the external controller providing combined guidelines for the organization. Additional triggers were added to the refined goal model. The supervise goal, which had been distributing combined goals among participants during the INIT state, was enhanced to adapt participant goals during the SUPERVISE state in response to each participant's simulated history.

Organization guidelines were grouped into objects with defined purposes, making the system easier to expand as requirements were added. Three types of guidelines were given to each organization: combined load guidelines, combined power quality guidelines, and evaluation guidelines that reflected desired feedback intervals and forecast horizons. As a holarchy, the combined organization guidelines could be adapted in response to temporal conditions just as local participant guidelines were adapted. Plan states continued to evolve

to reflect more complex logic and additional actions and events were added to define the transitions between states. Objects and attributes were added to the domain model as more external devices were defined.

Capabilities grew in functionality as plan state logic developed. Capability methods were enhanced to include simulation interfaces and smart meter sensor capabilities began obtaining simulated device data from MATLAB. As capabilities became more complex, they were refactored into smaller, more specific capabilities that in turn began to grow in functionality. An IPDS Builder component was added to support the reliable generation of test cases.

7.7.3 Iteration 3 – Filling in the Framework

The third iteration focused on extending the refined goal model; introducing forecasting goals and adding supporting agent types. Although goal changes represent a relatively major change to the IPDS design, by following the guidelines and recommended process and code policies, new features were added relatively easily. The application, once established, was easy to extend. Additional goals brought additional triggering events and goal parameters.

As communications are added to plan diagrams, they include the specification of the performative, the type of message content, and the role of the agent with which the communication takes place. Message types and their associated message content types were implemented for each communication capability.

As an IPDS organization starts up, agents participating in the organization register with the control component master. The specification goal tree gets instantiated and activates the top level goal along with any non-triggered, non-preceded leaf goals. For example, as the goal plan for the Supervise Prosumers goal is executed, the Supervise Prosumers Plan INIT state triggers an instance of the Manage Prosumer goal for each participant. As each home agent gets assigned to a Manage Prosumer Role, it first enters the Manage Prosumer Plan INIT state, and then triggers a new instance of an associated Forecast Prosumer goal. The

home agent then transitions to the Manage Prosumer MANAGE state and begins sensing consumption and generation readings, which it reports back to the Supervisor, alerting the Supervisor if it detects an out-of-bounds condition. The supervisor optimizes combined local guidelines within the organization, adapting participant goals to maximize compliance. If guidelines cannot be met within the local organization, the supervisor will raise a request to the external controller who will, in turn, attempt to address the request from within the controller agent's local organization, recursively raising requests up the holarchy until a solution is available.

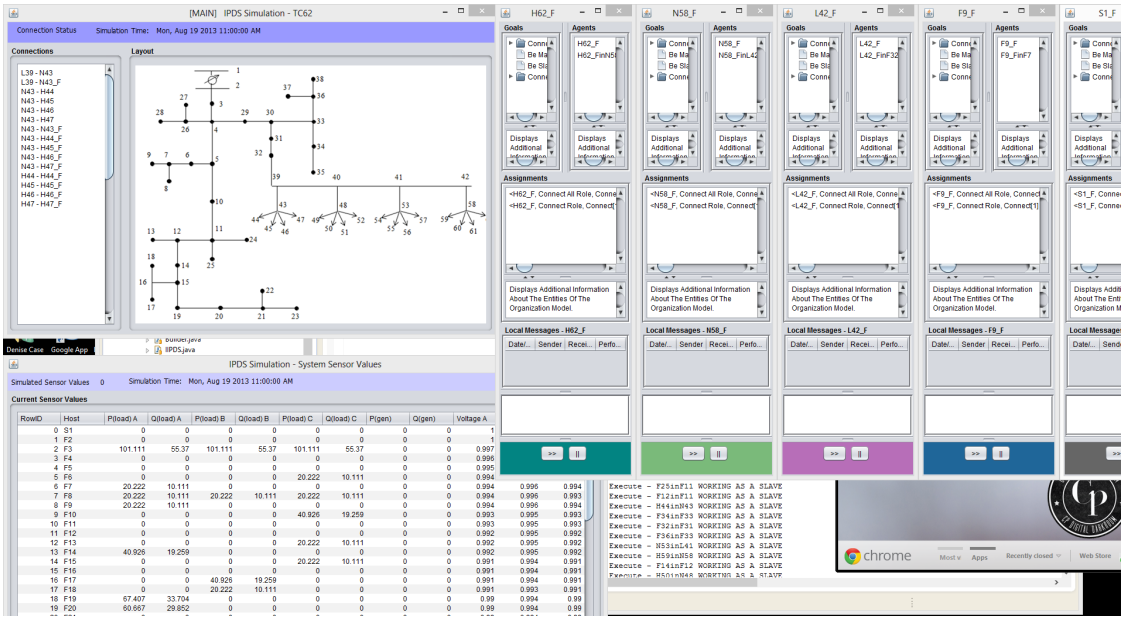


Figure 7.2: *Implementing complex MAS with AO-MaSE.*

An early cut all the way through a complex system supports early visual feedback. Figure 7.2 shows a view into a running IPDS test case. There is one organization window for each multigroup agent. Organization goals appear in each top left panel. Roles appear in the top center panel. Participating agents are displayed in the lower right panel. In the center bottom panel, assignments are displayed, indicating that each agent has been assigned to a specific instance goal based on their capabilities and attributes as defined in the agent configuration file. In this assignment panel, the current values of the agent's goal

parameters are shown as they are adjusted by their local group administrator. The prosumer goals are being distributed in accordance with each participant's demand. Maximum kW guidelines may be positive or negative. Negative upper boundaries are assigned to a participant generating more PV power than the participant is consuming. The extended system passed a variety of tests and demonstrated the operation of multiple goal models and assignments. During this iteration, the system was extended to additional levels of the holarchy in preparation for the evaluation of the test case involving 62 location-based hosts and approximately 46 local IPDS organizations. The test case is based on the IEEE Distribution System Analysis Subcommittee 37-Node Test Feeder case that begins with the substation node labeled 1. Electricity is distributed out along the 3-phase feeder lines. Extensions to the IEEE test case were made to test the system down to the home level. In this version, there were four nodes along a single-phase lateral line (39-42), four nodes corresponding to agents running on neighborhood transformers (43,48,53,58), and four homes being supplied by each transformer. The earliest trials assumed one of every four homes was equipped with roof-mounted solar PV panels. For example, Home 44 had PV distributed generation (DG) capabilities, but Homes 45, 46, and 47 under Neighborhood Transformer 43 did not.

7.7.4 Discussion

Each number on the test case diagram corresponded to a physical location or node that could host IPDS agents. These locations had sensors and/or actuators depending on the physical configuration being simulated. Generally, real power (P) and reactive power (Q) consumption (load) values were assumed to be available by phase at each of the nodes. In addition, homes equipped with PV had sensor readings available for the real power generated. Actuators or controllable equipment ranged from a single load tap changer at the top of the distribution network, down through capacitors on the three-phase feeders to smart inverters, which allowed for some moderation of reactive power at each PV-equipped home. Reactive power is typically "non-useful" power but can be used to help manage the

power quality characteristics during periods of drastic changes in generation associated with intermittent clouds. In addition, voltage readings may be used to help minimize losses and optimize efficiency. The simulation pulled simulated readings every second for each of the 62 nodes. Calculated control values were calculated in response to the readings. Calculated values consistently matched the updated control settings calculated in reference MATLAB test cases.

7.8 Summary

This chapter describes AO-MaSE, a customized O-MaSE-compliant process and suggests a novel approach based on early execution and iterative extension for engineering complex systems with multigroup agents. It describes a recommended software engineering process employing specific design conventions that begin simply and focus on moving sooner from initial concepts to code construction while creating an evolving, iterative framework suited to the development of complex, adaptive, intelligent, autonomic systems. The effort includes policy recommendations and detailed guidelines that produce a vertical slice of a complex system earlier in the process, forming a working core that enables quicker feedback into the behavior of a complex, recursive, hierarchical HMAS. The process is compliant with the proven O-MaSE process and enables the full functionality needed for complex control systems yet offers a structured path towards implementation that addresses several challenges encountered when developing complex MAS. The AO-MaSE process can be used to build systems of intelligent systems as described in Chapter 8.

Chapter 8

Evaluation

A complex system that works is invariably found
to have evolved from a simple system that worked.
A complex system designed from scratch never works
and cannot be patched up to make it work.
You have to start over, beginning with a working simple system.
— *John Gall*[167]

This chapter discusses the software experiments and the evaluation of the OBAA^{++} agent architecture and the AASIS framework for implementing complex systems. Section 8.1 describes a complex system, an *intelligent power distribution system* (IPDS), in which the agents operate a *complex multiagent system* (MAS) for a Grid Control System (GCS) for continuous volt-var control and a second a complex MAS for an Online Auction System (OAS) for conducting online auctions near sources of distributed generation (DG). Section 8.1.1 describes the experiments and evaluation with the GCS and Section 8.1.1 describes the experiments and evaluation with the OAS. Section 8.1.3 describes the evaluation with a *system of intelligent systems* that includes agents running both the GCS and

OAS. Section 8.2 describes the experiments and evaluation in a second application domain, a graduate school research lab (GSRL), and the test cases associated with implementing architectural components to support agents managing goal consistency when accepting assignments from different systems. Section 8.3 provides a chapter summary of the results of the evaluations.

8.1 Evaluation of AASIS on PDS

This chapter shows how AASIS and the OBAA^{++} agent architecture balance the dual objectives of flexibility and reusability, and demonstrate the architecture's efficacy by demonstrating how existing intelligent power distribution agents reapplied the same architecture and process to implement a second, goal-driven complex MAS operating concurrently and utilizing the same underlying physical equipment and power distribution system (PDS).

8.1.1 Grid Control System (GCS)

An example distribution system was developed based on an industry standard test feeder and was first used to evaluate power quality control algorithms to use smart inverters to assist with managing voltage[168,19].

Information flow in the new grid control holarchy is illustrated in Figure 8.1. Sensor information from home smart meters with power generation and consumption information as well as voltage information is sent up the power quality holarchy for aggregation at progressively higher levels. At the same time, local optimization algorithms are calculating new requests for reactive power assistance and distributing the requests as modified goal parameters back down the holarchy.

The intelligent power distribution system implemented includes a complex MAS, focused on volt-var control and overvoltage prevention. Implementation followed the AO-MaSE process described in Chapter 7.

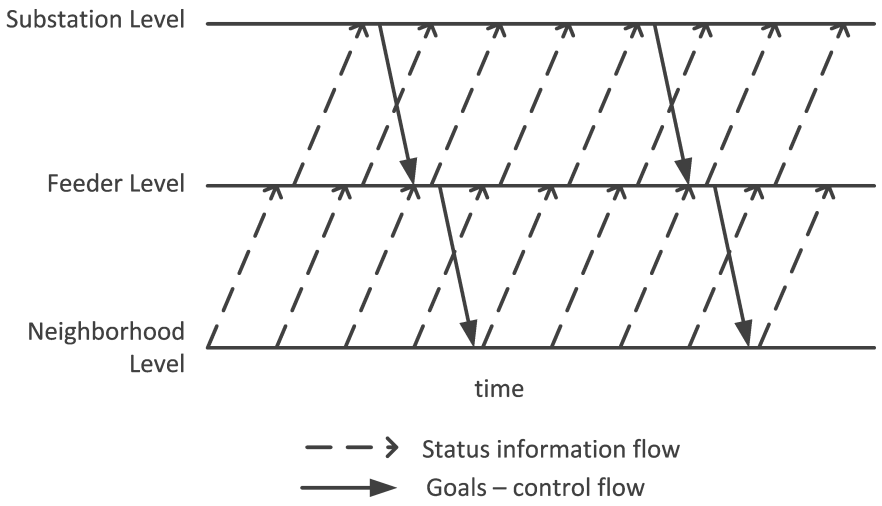


Figure 8.1: *Simplified information flow in the grid control holarchy.*

Two goal types were defined for the GCS:

- Be Super Holon. A goal to act as the head of the holon, accepting goals for the holon, registering and maintaining registration with, and creating and issuing assignments to, participants in the holon (the body) to maintain control objectives as defined in the goal.

- Be Holon. A goal to act as part of the body of the holon, registering with the head, accepting assignments, and doing its part to maintain grid control objectives as defined in its goals.

These goals worked hierarchically throughout the system.

Both goals are parameterized goals, accepting parameters that characterize the desired behavior. There are two parameterized goals, one related to the information needed to form the necessary connections and registration with the external, affiliated agents in each local group, and a second that contains the guidelines for the specific type of grid control desired. By parameterizing the goals consistently, when the super holon in a high-level group creates goals for the set of participating holons (the body), the body holons have the information needed to in turn, act as a super holon in a lower level group, passing down its portion of

the desired behavior into smaller sub problems to its participating body.

Following AO-MaSE, two roles were defined for the GCS:

- Be Super Holon Role. This role can be used to achieve the Be Super Holon goal.
- Be Holon Role. This role can be used to achieve the Be Holon goal.

Using roles reduces the coupling between agents the behavior specification for the organization. The *Reorganization Algorithm* (RA) is used by the agent playing the Super Holon Role to determine dynamically the best roles for participants in the body of the local group in response to the current guidelines and abilities of each participating agent.

Following AO-MaSE, two plans were defined for the GCS:

- Be Super Holon Plan. The Be Super Holon Plan includes plan steps for *Initializing* and *Administering* the local group. The Initializing step includes initializing the Grid Admin Capability based on the guidelines, and setting up the organization and goal reasoning needed to distribute assignments to itself and the participating agents. The main Administering step includes accepting and maintaining connections and registrations from the body so that participating agents can be included in goal reasoning and accept assignments in the local group. It also includes the work flow for managing grid control. Tests included aggregation of the power reports sent from the body and passing the information up the holarchy. A power report includes information about the current power readings, the last power readings (taken at the prior time step), the available margin, and assessments of the trends (based on the difference between the last reading and the current reading).
- Be Holon Plan. The Be Holon Plan includes plan steps for *Initializing*, *Registering*, *Confirming*, and *Participating* in the local group. The Initializing step includes information for connecting to the head, while registering and confirming are used to make sure the agent is registered with the head and ready to receive assignments in the local group. The main Participating step includes either getting simulated sensor

Capability	Description
Date Time	The ability to understand the simulation systems view of the current date and time and to derive information such as time of day, and time and to derive information such as time of day, weekday/weekend, associated event schedules, and apply time-dependent information such as forecasts.
Grid Admin	The ability to create a new grid control organization according to the model specifications provided and administer the organization, determining assignments and assigning them to appropriately-equipped agents.
Grid Control Holon	The ability to create and report voltage and VAR information.
Grid Participate	The ability to join and accept assignments in a grid control organization.
Grid Control Super Holon	The ability to gather, aggregate, and assess voltage var information for the local grid control group.
Power Communication	The ability send and receive messages related to grid control.
Smart Meter	The ability get a load, generation, and voltage (if available) reading from an authorized smart meter device.
Smart Inverter	The ability to set the reactive power setting on an authorized smart inverter device.

Table 8.1: *Key capabilities in the grid control system.*

readings, as the home prosumer agents do, or getting aggregated power messages, as the higher level agents do. PV-enabled home agents adjust their associated smart inverter settings to balance their immediate needs in response to swings in distributed generation (DG) (see Figure A.2) and also provide information about their margin in power messages to the head.

Agents were provided with capabilities that matched the sensors and actuators on their host (e.g., smart inverters on PV-enabled homes). Some key capabilities are listed in Table 8.1.

All agents were implemented in Java using OBAA^{++}. Each IPDS host was run in a

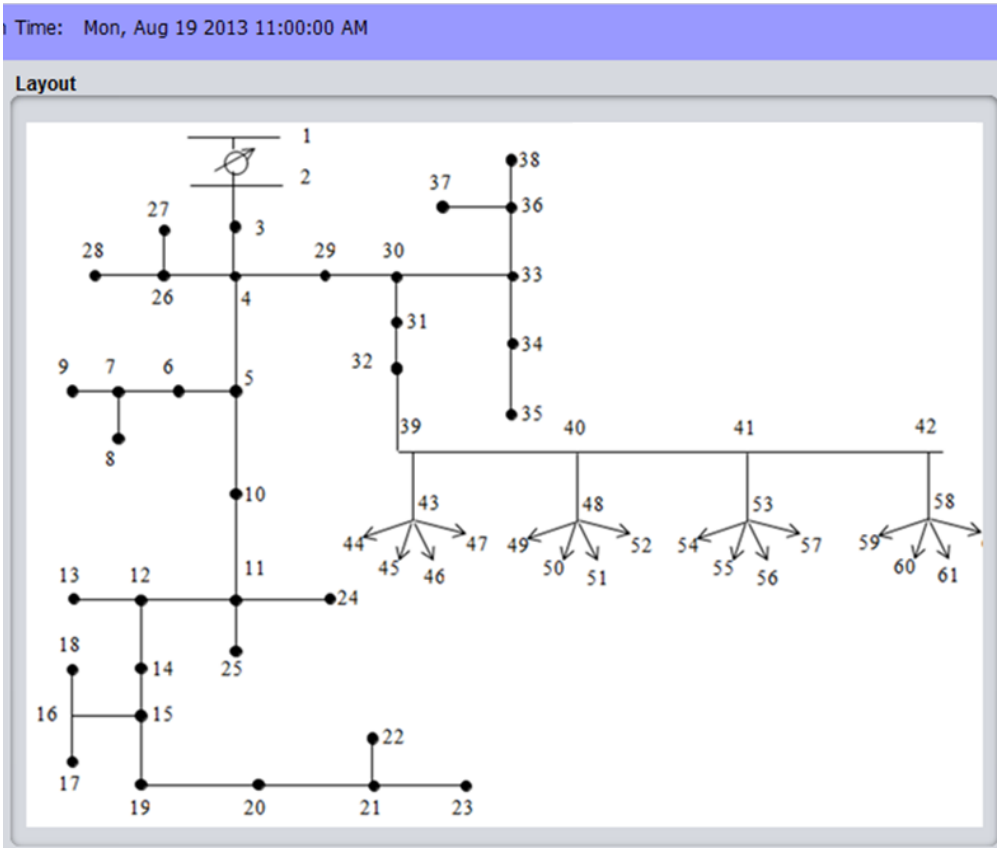


Figure 8.2: *Partial topology of an IPDS test case.*

separate Java virtual machine, with each host supporting one or more agents. Communication between agents was provided using RabbitMQ, with two separate communication channels used for each organization, one for CC-CC communication and one for EC-EC communication.

Each agent was configured with a predefined list of its affiliated organizations and whether it would be the initial master of those organizations. This list was generated from the physical layout of the IPDS test cases and is appropriate for this type of system. The initial master agent has the responsibility of initiating their appropriate organizations and providing the specified organizational knowledge (goals, roles, etc.) to the other agents.

The system worked in discrete time intervals where MATLAB computed voltage values within the PDS based on second by second power generation and power consumption values taken from historical data.

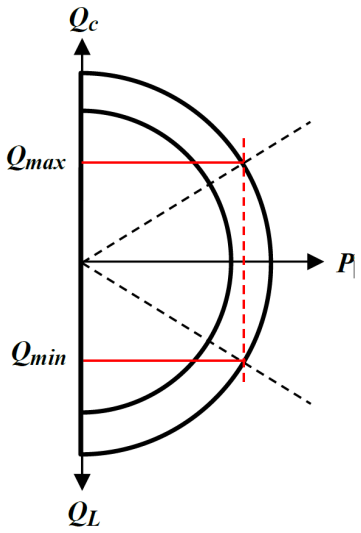


Figure 8.3: *Oversized smart inverters offer 20% more reactive power contribution for volt-var control*[19].

The first trials implemented the control algorithm using an oversized smart inverter[19]. Home agents obtained readings, homes with DG calculated their reactive power needs, and and sent power messages to the head with trends and available margins. The head aggregated the requests from all participants and updated the guidelines for the DG-enabled home to adjust reactive power settings to assist with the combined needs of the neighborhood. A series of tests were written using the Spock Specification Language, which were verified against both MATLAB calculations and manual computation. Tests were used to verify that the system produced correct results, specifying the desired setting when possible, or a maximum value when constrained by the available reactive power setting based on a 20% oversized smart inverter shown in Figure 8.3.

Details for some of the smart inverter tests are shown in Table 8.2. Reactive power is indicated with Q. Pgen refers to total distributed generation and PL refers to total consumption load. Current values are shown as well as values sensed at the last time slice.

Data from MATLAB was fed into the IPDS system where control values were computed and fed back into MATLAB for the next time period. Simulated sensor readings were pro-

Capability	Description						
Qsetting	Pgen	lastPgen	PL	lastPL	QL	lastQL	lastQqen
0.3962	0.4492	0.6001	5.0289	5.0289	3.7717	3.7717	0.4501
0.4486	0.6001	0.5993	0.5493	0.5493	0.4120	0.4120	0.4493
0.4485	0.5993	0.5989	2.0591	2.0591	1.5443	1.5443	0.4489
0.4450	0.5989	0.5970	0.7518	0.7518	0.5639	0.5639	0.4469
0.4501	0.6001	0.6001	5.0000	5.0000	3.0000	3.0000	0.4501
0.4407	0.7501	0.7501	5.0000	5.0000	3.0000	3.0000	0.4469
0.3080	0.3492	0.6001	5.0289	5.0289	3.7717	3.7717	0.4501
0.4000	0.5000	0.5000	2.0000	2.0000	3.0000	3.0000	0.4000
0.5732	0.6500	0.6487	0.4612	0.4612	0.3258	0.3258	0.7901
0.2222	0.7500	0.6842	0.1111	0.1111	0.2222	0.2222	0.2880

Table 8.2: *Tests on reactive power (Q) settings calculated given different sets of input power readings.*

vided by MATLAB and distributed to the appropriate agents as sensor readings, depending on their location and their available sensors. Typically, these sensor readings included the real and reactive power consumed, voltage levels, and the real and reactive power generated (for Home agents with PV panels).

Home agents with smart inverters then calculated the appropriate reactive power settings based on their sensor data and goals. The agents then sent their calculated settings back to MATLAB for recalculation of the new sensor readings.

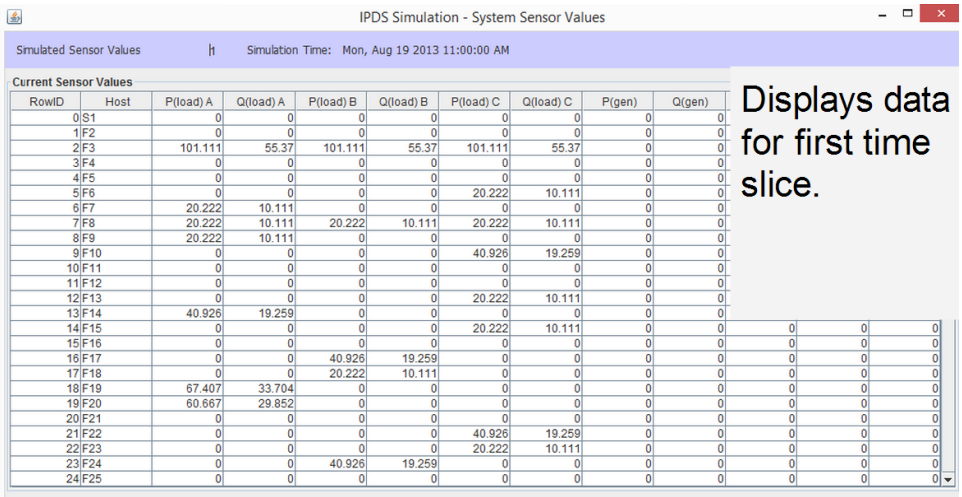
On startup, the simulation can be paused to verify the initial set of set simulated sensor data for all 560-devices is read correctly (see Figure 8.4).

A PDS can be decomposed into a three-level hierarchical ordered system based on natural physical topology (i.e., substation, feeder, and neighborhood levels). A neighborhood holon represents a single-phase transformer serving a group of residences or end-user prosumers. The neighborhood level encompasses all neighborhood holons. A feeder holon includes single-phase laterals with nested groups of neighborhood holons. The feeder level includes all feeder holons. Finally, feeder holons are nested in a substation holon, which includes the three-phase primary distribution lines and laterals connected to the distribution substation. In this view, the substation level includes a single substation holon.

The holonic partitioning scheme requires no physical change in system configuration or customer connection. Moreover, the proposed architecture relieves loading on communication and information processing while reducing the control, and energy management computational burden by enabling substation holon targets to be cascaded down to lower level holons of the holarchy.

The holonic control sequence is illustrated in Figure 8.5. The control algorithm begins with the lowest level home prosumer agents. Some specified fraction of the home agents are equipped with rooftop solar PV panels for distributed generation. The first iteration starts with all home agents reporting their real and reactive power demand and generation (if available) to their neighborhood transformer agent. The neighborhood transformer agent aggregates the sensor readings from all connected homes (typically 2-6) and forwards the aggregated values and additional supporting information up to their designated 3-phase feeder line agent, which in turn forwards their aggregated values to the substation agent.

The substation agent runs an optimal power flow (OPF) calculation to calculate new targets based on the responses received and passes the new targets back down the same hierarchy for progressive distribution. After the first new targets reach all the way down to the homes, the homes check to see if the new targets are within a given tolerance of their



RowID	Host	P(load) A	Q(load) A	P(load) B	Q(load) B	P(load) C	Q(load) C	P(gen)	Q(gen)			
0	S1	0	0	0	0	0	0	0	0			
1	F2	0	0	0	0	0	0	0	0			
2	F3	101.111	55.37	101.111	55.37	101.111	55.37	0	0			
3	F4	0	0	0	0	0	0	0	0			
4	F5	0	0	0	0	0	0	0	0			
5	F6	0	0	0	0	20.222	10.111	0	0			
6	F7	20.222	10.111	0	0	0	0	0	0			
7	F8	20.222	10.111	20.222	10.111	20.222	10.111	0	0			
8	F9	20.222	10.111	0	0	0	0	0	0			
9	F10	0	0	0	0	40.926	19.259	0	0			
10	F11	0	0	0	0	0	0	0	0			
11	F12	0	0	0	0	0	0	0	0			
12	F13	0	0	0	0	20.222	10.111	0	0			
13	F14	40.926	19.259	0	0	0	0	0	0			
14	F15	0	0	0	0	20.222	10.111	0	0	0	0	0
15	F16	0	0	0	0	0	0	0	0	0	0	0
16	F17	0	0	40.926	19.259	0	0	0	0	0	0	0
17	F18	0	0	20.222	10.111	0	0	0	0	0	0	0
18	F19	67.407	33.704	0	0	0	0	0	0	0	0	0
19	F20	60.667	29.852	0	0	0	0	0	0	0	0	0
20	F21	0	0	0	0	0	0	0	0	0	0	0
21	F22	0	0	0	0	40.926	19.259	0	0	0	0	0
22	F23	0	0	0	0	20.222	10.111	0	0	0	0	0
23	F24	0	0	40.926	19.259	0	0	0	0	0	0	0
24	F25	0	0	0	0	0	0	0	0	0	0	0

Figure 8.4: *Displaying simulated sensor data during initialization.*

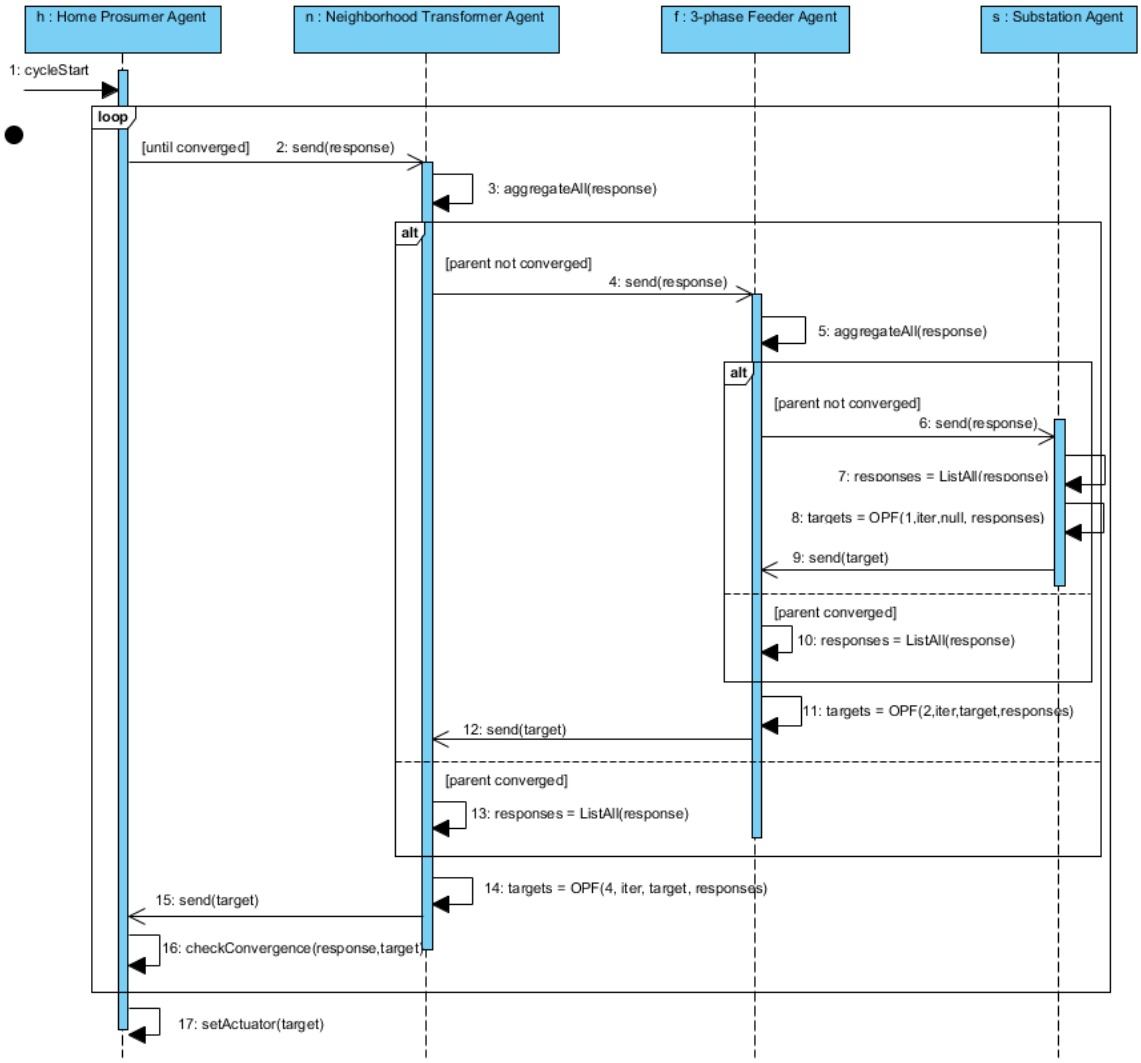


Figure 8.5: *Iterative grid control sequence diagram.*

response values and the desired operation of their associated smart inverter (if PV-enabled).

After the first iteration, they likely are not, so the PV-enabled homes send their proposed response actions based on their new targets, and these new response values are again aggregated and communicated up the hierarchy. The process continues until the substation targets are not significantly different. At this point the substation level is considered converged, and the process continues from the feeders on down. Within a couple iterations, the feeder level target values are considered converged and the process continues between the neighborhoods and the homes. When the neighborhoods don't change, the homes will

make any final adjustments to their actions based on their last set of new targets and cycle is complete. Each home executes their control actions and the cycle is ready to begin again.

Following the AO-MaSE recommended process, a simple initial test case was first prepared. Each smart device was assumed to provide a host for a set of intelligent agents. The first test case included one neighborhood organization, five hosts, ten agents, and eight inter-agent connections (communication connections between persona within a single agent). The test configuration was scaled to include 62 hosts, 124 agents, 46 organizations, and 110 inter-agent connections. The layout of this configuration is shown in Figure 8.2 where the lowest-level nodes (e.g, 44-47, 49-52, etc.) correspond to homes. Current trials involve cases with up to 400 hosts and 800 agents and are being used to develop two additional algorithms for GCS and further evaluate the scalability of the IPDS architecture.

All agents were implemented in Java using OBAA^{++}. Each IPDS host was run in a separate Java virtual machine, with each host supporting one or more agents. Communication between agents was provided using RabbitMQ, with two separate communication channels used for each organization, one for CC-CC communication and one for EC-EC communication.

Each agent was configured with an organization specification detailing the agent's affiliated organizations and the default master for each affiliated organization. Details for the organization specification was generated from the physical layout of the associated PDS driving the test case and the details of the grid control algorithm being tested. The default master agent has the responsibility of initiating their appropriate organizations and providing the specified organizational knowledge (goals, roles, etc.) to the other agents.

Experiments run in discrete time intervals, using simulated sensor readings directly from MATLAB-computed power and voltage values based on second-by-second power generation and power consumption values taken from historical data. Simulated sensor readings were provided by sensor adaptors to MATLAB and distributed to the appropriate agents as sensor readings, depending on their location and their available sensors. Sensor readings

included the real and reactive power consumed, voltage levels, and the real and reactive power generated (for Home agents with PV panels).

IPDS Home agents with smart inverters then calculated the appropriate reactive power settings based on their sensor data and parameterized goals.

In all test cases, the agents correctly aggregated real and reactive power values and assessed voltages correctly, matching the test case results provided from electrical engineers, and verifying that the system produced the same results as obtained during their MATLAB-only computations. Agent capabilities accessed simulated measurements and higher level agents correctly aggregated values and assessed them as in-bounds or out-of-bounds. Home agents operated their smart inverters in response to the simulated measurements. In addition, agents calculated the amount of reactive power margin based on an over-sized inverter and reported the margin up the hierarchy. All communication took place vertically. No algorithms tested thus far include negotiation among peers. Several options for smart inverter algorithms were tested. The algorithm to use is provided to the agents with their initial guidelines (parameterized goals).

Addition of new actuators, including capacitors on both the single-phase lateral lines and three-phase feeder lines is planned as described in Section 9.4.

8.1.2 Online Auction System (OAS)

In addition, a second multigroup, complex HHMAS was implemented in a *complex MAS* co-located and operating concurrently with the GCS.

In the associated new online auction holarchy, each *home agent* participates in a single local group at the lowest level of the auction holarchy. Each of these locals group includes a single neighborhood transformer agent assumed to be located on or near a nearby pole transformer that supplies a set of homes with electricity. Each neighborhood transformer agent supported exactly four homes, with one of the four having rooftop photovoltaic (PV) panels for generation.

Each neighborhood transformer agent executed (brokered) a first-tier auction, accepting bids from four participating homes for a given future time. Homes equipped with PV were assumed to have surplus power to sell that nearby homes (served by the same transformer) could bid for. The transformer agent and its homes autonomously create a small local market organization to execute the auction.

Each transformer agent also participated in a higher-level tier-2 auction. In these secondary auctions, each transformer agent served as an auction participant, while the associated lateral line agent accepted four transformer bids and brokered a second-tier double auction.

The two-tier double auction is implemented as a two-part linear programming problem; special computational capabilities were employed by the agents[40].

The volumes of energy bought and sold are determined through the auction by maximizing the total utility of all participating agents. The secondary auction requires the power requested from each neighborhood transformer agent to serve as the neighborhood bid quantity while the tier 1 clearing price serves as the new buy or sell price. The lateral line agent serves as the broker in the second-tier auction and determines the final clearing price at which the power trading occurs. There are various ways in which the clearing price can be determined (e.g., through negotiations with the utility company, to obtain budget balance, or by other means). The objective of the auction is to maximize the social welfare function (SWF), the aggregated utility of all winners.

Each auction was conducted asynchronously in accordance with the specific guidelines provided as goal parameters. Guidelines include those specified for the market organizations in which the online auctions will be conducted, as well as custom guidelines given to each multigroup agent that serve to direct the behavior of each agent in such a way that the agent could be customized to reflect the personal pricing strategies and comfort/profit motives of the owner. Some agents may be ultimately controlled by the homeowner, making the decision whether to sell power and at what future time and price. Some agents may be

wholly owned by the power company or market agency, for example, those running along the lateral lines. Simulated CPS agents communicate over RabbitMQ, a fast implementation of the Advanced Message Queuing Protocol (AMQP) standard.[169]. The Java-based simulation runs on Windows and iOS.

A smart system running on or near the smart meter may be a likely candidate to support the brokering of online auctions between homeowners and the grid. These test cases focused on extending the multigroup agents already running the GCS, a complex MAS for grid control to simultaneously support concurrent calculations for bidding and brokering online sales agreements among the PDS stakeholders.

The OAS employs a *two-tier double auction* scheme where home prosumer agents create bids to express their intentions and send them to an agent acting as the broker in a local market organization. The agent brokering the local auction determines the optimal resolution of the auction, and in the event of any unsatisfied amounts, participates as a bidder in a secondary, higher-level auction. The approach exploits the applicability of the double auction in the second-tier, where the auction takes place between the secondary participants representing their remaining community bids and shows the efficacy of the proposed hierarchical model as it further maximizes the overall social utility.

The motivation grew from research into several two-tier resource allocation techniques. Most specifically, that of spectral allocation such as Zhou's[170] where a two-tier resource allocation approach has been proposed that integrates a dispatcher-based node partitioning scheme with a server-based dynamic allocation scheme. Also, Abdelnasser[171] proposes a semi-distributed (hierarchical) interference management scheme based on resource allocation for femtocells[1]. In addition, several other market-based economic models have been proposed for the process of competitive buying and selling to solve for an optimal power flow in a smart grid. Local interactions[172] and decentralized resource scheduling[173] have been considered with better convergence under tight computational budget constraints. Auc-

[1]Femtocells are small, low-power cellular base stations

tions are an efficient mechanism, easily implemented in a grid structure, that allows buyers and sellers to compete for the resources to be auctioned to achieve an optimal resource flow in order to maximize the social benefits to the participants. *Double auctions* are auctions that involve both buyers and sellers. These auctions can be designed as an efficient, incentive-compatible mechanism where buyers and sellers participate without the risk of losing anything by choosing to participate. A recent study on auctions for spectrum allocation in wireless networks[174] has shown that a double auction can achieve a greater social welfare (benefit) compared to other auction mechanisms such as the well-known Vickrey-Clarke-Groves (VCG) mechanism[175]. An efficient double auction mechanism with uniform pricing has been proposed[176], that considers the dynamic, heterogeneous and autonomous characteristics of resources in a grid computing system. The double auction was developed and analyzed as a mechanism to characterize the trading price of the energy trading market that involves the storage units and the potential energy buyers in the grid[177]. Furthermore, several applications[178,179,180] have been proposed in a different field of study and have been shown as an effective mechanism when interest of both buyers and sellers are taken into consideration for a competitive market happening in a computational grid system. No existing literature was found where a double auction has been implemented in a hierarchical manner for electricity trading in isolated microgrids to achieve a greater social benefit in PDS. The implementation in a two-tiered structure, comprised of intelligent agents participating in the auction by sending messages to the auctioneer indicating an interest to buy or sell, demonstrates flexibility and reusability as follows. First, the proposed two-tier approach implements bids in two reusable stages. In the first tier, the auction involves individual homes within a neighborhood acting as buying and selling agents. In the second tier, a similar auction between multiple neighborhoods is conducted. This arrangement follows the spatial topology of the PDS, where feeders deliver power via several transformers to the neighborhoods. The approach is flexible in that it could be implemented in existing distribution systems – or added to existing intelligent distribution systems – without needing

separate systems for information exchange, with both agents at the transformers and on feeders reusing the capabilities and plans to broker auctions.

Second, the double auction mechanism proposed is formulated as a linear programming problem known to be of exponential complexity. The tiered-approach can be perceived as a divide-and- conquer scheme that divides the larger auction problem at the feeder level into several smaller, more tractable sub- problems, one corresponding to each neighborhood, that are solved in a parallel fashion.

Lastly, the constraints imposed upon the auctions taking place at the first tier and second tier are different. It can be assumed that across individual homes within a small geographical neighborhood would entail an underlying well-connected social group. Hence, the demands or supplies of electricity of individual homes at any given instance can be gleaned either from historical data or from prediction algorithms. These can serve as bids for the first tier auction, obviating the need for direct human intervention. Furthermore, the energy pricing must be uniform across the entire neighborhood. These requirements need not hold at the feeder level, where each neighborhood may be priced differently. Furthermore, due to larger geographic distances between neighborhoods, the auction may have to take into account additional factors such as I^2R loss, local cloud conditions, etc.[181]. Some of these issues, not currently taken into account, could be readily incorporated with minor modifications.

Agents were equipped with the key capabilities shown in Table 8.3.

The holonic power auction organizations were all fed from a single power power line, with a single lateral feeder agent, L39. The power line was assumed to supply four neighborhood transformers, each hosting one of the neighborhood transformer agents, N43, N48, N53, and N58. Each transformer supplied four homes, with one of the four homes providing mid-day power from rooftop PV panels. Each home was assumed to host an agent. The four agents associated with PV-enabled homes generated offers to sell power at a given future time to the other three homes in their neighborhood in the first-tier auction. The four neighborhood agents all received four bids from the supplying homes—one to sell power,

Capability	Description
Auction Communication	The ability send and receive messages related to online auctions.
Auction	The ability to create bids to buy or sell power to be exchanged at a specific future time.
Broker	The ability to gather bids and broker a specific auction for a local online market organization.
Date Time	The ability to understand the simulation systems view of the current date and time and to derive information such as time of day, and time and to derive information such as time of day, weekday/weekend, associated event schedules, and apply time-dependent information such as forecasts.
Market Admin	The ability to create a new online market organization according to the model specifications provided and administer the organization, determining assignments, and assigning them to appropriately-equipped agents.
Market Participate	The ability to join and accept assignments in an online market organization.

Table 8.3: *Key capabilities in the Online Auction System (OAS).*

and three offers to buy power. Upon receiving the bid messages from the four home agents, each neighborhood agent acted as a broker to execute the local auction. After executing the first-tier auctions, some bids were not completely fulfilled. The brokering agent determined the remaining quantity and forwarded the offer to the lateral agent for the second tier auction to be brokered by the lateral agent.

We can look inside the transformer agent participating in a 2-tier double auction and see how the process flows. This agent is concurrently supporting grid control communications as well, in a highly analogous but more complex process. The steps are shown in the Neighborhood Transformer agent type beginning with the self persona and the two affiliate persona participating in the Auction Holarchy as shown in Figure 8.6.

The numbered steps indicating persona interactions during a two-tier auction are summarized below. See also Figure 8.6.

1. Self initializes a (Tier 1) T1 broker persona.

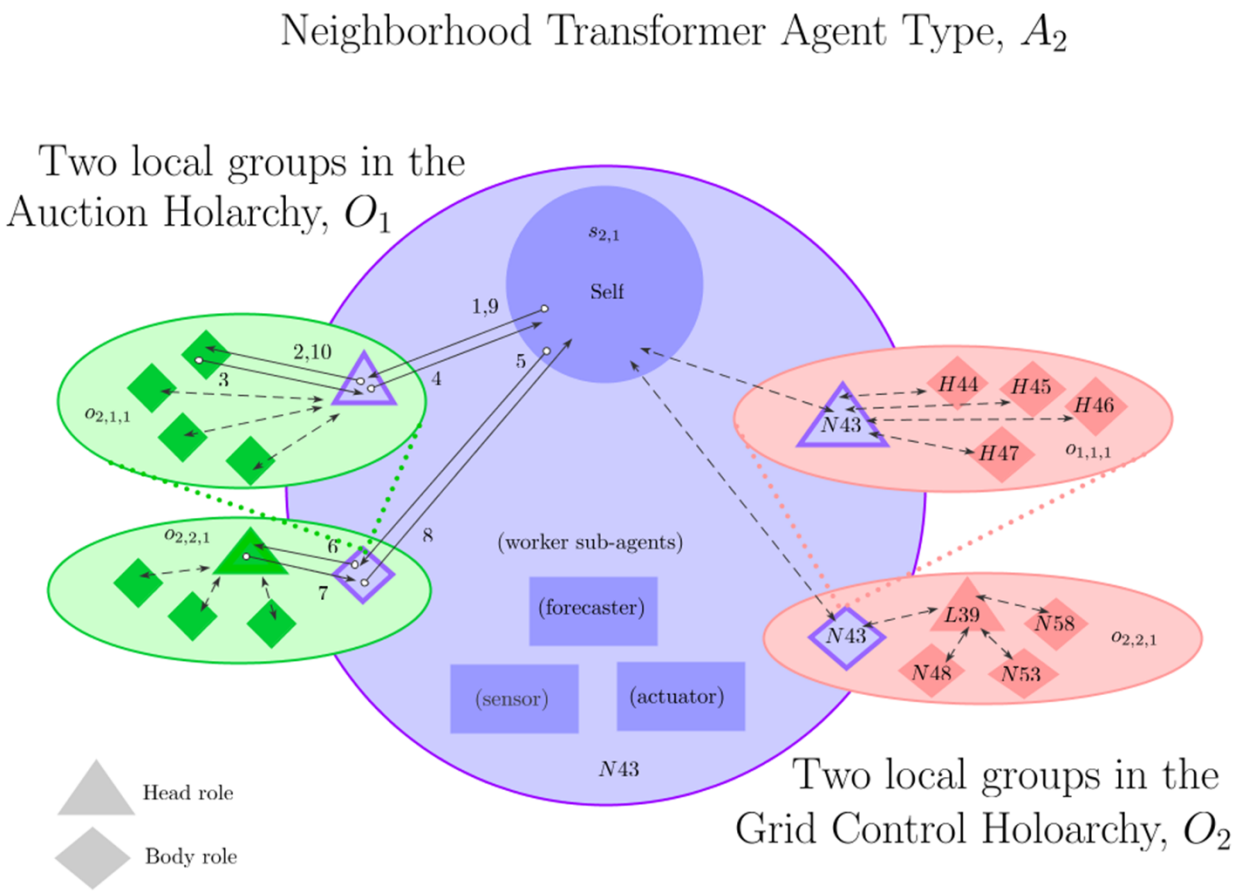


Figure 8.6: *A multigroup agent with multiple affiliated groups. Numbered steps indicating persona interactions during a two-tier auction.*

2. T1 broker connects and registers each external T1 bidder.

3. T1 bidders submit bids to T1 broker.

4. T1 broker forwards initial results to Self for review.

5. Self reviews, forwards to T2 bidder.

6. T2 bidder submits bid to T2 broker.

7. T2 broker issues T2 results to T2 bidder.

8. T2 bidder forwards to Self for review.

9. Self reviews, forwards to T1 broker.

10. T1 broker issues T1 auction results to T1 bidders.

The bid information and auction clearing results for four first-tier auctions is shown in Figure 8.7. In each first-tier auction, there were three bids to buy power as indicated by a non-zero buy bid price, b_i, shown in yellow. In each first-tier auction, there was one bid to sell power, indicated by a non-zero sell bid price, s_j, shown in green. The agent name, A_i, and the bid quantity for all bids, $q_{i,j}$, are also shown in the four first-tier tables. The associated home agents send their bid message to their associated neighborhood broker. The neighborhood brokers translated the message content information from each participant into a array of bids and bid information and used a double-auction computational capability to execute the auction. In each first-tier auction, at least one bid could not be satisfied fully within the first-tier auction. Unsatisfied quantities were used to create bids at the first-tier clearing price in a second-tier auction.

In addition to serving as brokers in the lower organizations, the neighborhood transformer agents also serve as auction participants in the higher-level second tier auction organization brokered by the agent running on the power line. The lateral power line broker agent then brokers a second-tier auction with the new secondary bids by again translating

the message information into an input array and executing the secondary auction computational capability. The inputs and results of a second-tier auction are also presented in Figure 8.7; there is one second-tier bid resulting from each of the four first-tier auctions. These are shown in the black text between the first-tier tables. N43 and N48 provided a single buy bid each for their unsatisfied quantities at the clearing price of their respective first-tier auctions. N53 and N58 provided a single sell bid each for their excess quantities at the clearing price of their respective first-tier auctions. All four neighborhood bids were forwarded to the broker of the second-tier auction and the results of the second-tier auction are shown in Figure 8.7 in red. The second-tier results where passed down to the first-tier participants, with the final reconciled quantities, $r_{i,j}$ shown beside the corresponding original bid quantities, $q_{i,j}$, provided in the four first-tier tables.

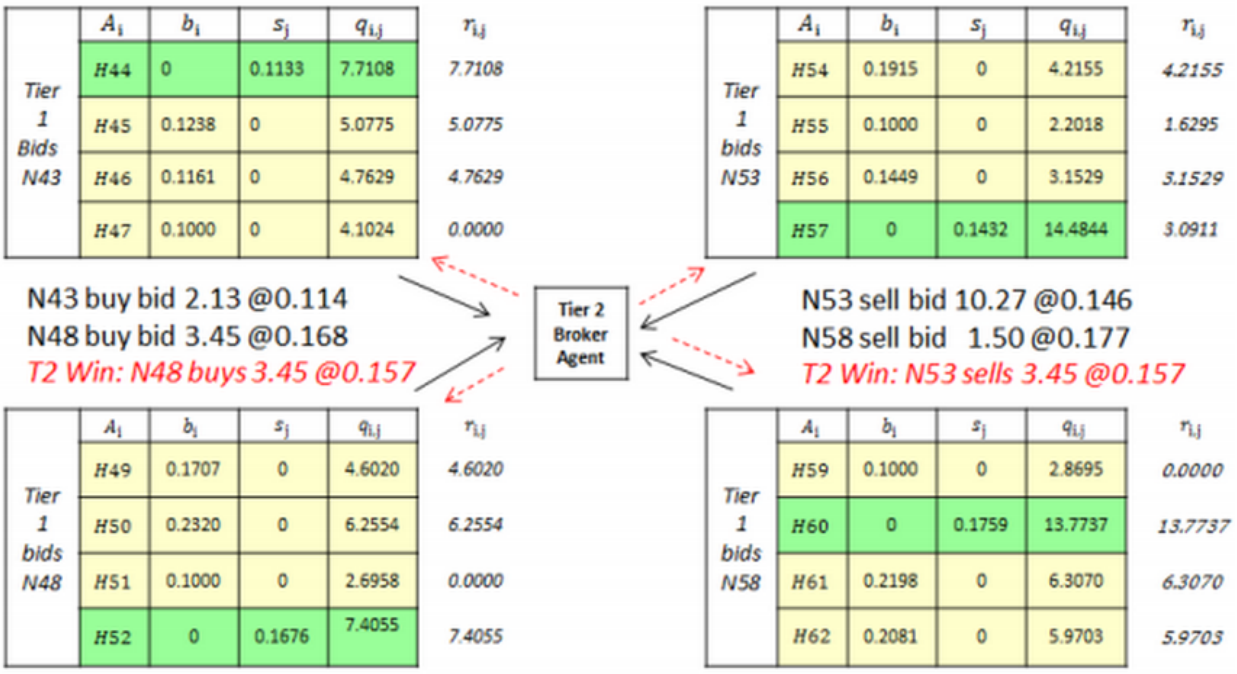


Figure 8.7: *Results from a two-tier auction trial.*

The additional test cases involved the new online auction behavior running concurrently with the continuous power quality control process. Example results are provided to show

the details of a single future power auction running on a single lateral line. The line feeds four neighborhoods, each with four homes. The process flow can be illustrated by looking at one of the neighborhoods, N48. In the first tier auction, the 4 homes send bids to the broker agent running on the transformer (as shown in the upper group). Any additional unsatisfied quantities are then used in a second tier auction, where N48 and the other three neighborhoods send bids to the agent running on the lateral line (as shown in the lower group).

In the first tier auction, the first three agents want to buy, and the forth, PV-enabled home wants to sell. The first two are willing to pay more than the selling price of 16.76 cents. But together they need 4.6+6.3 or 10.9 units of power and the seller only has 7.4. The broker satisfies each of them based on the quantities and price they bid at the first tier clearing price of 16.84 cents and will try to buy the remaining 3.5 units in the second tier auction.

In the second-tier auction, the price is good—another neighborhood, N53, is wanting to sell 10.3 units for just 14.62 cents each. The second tier broker will announce the clearing price and let N48 know that it was able to satisfy all 3.5 units requested. Back in the tier 1 auction, N48 will distribute this accordingly, with the requests from both H49 and H50 fully satisfied. Only the third tier 1 agent, H51 who hoped to buy at a too-low price of 10 cents was not successful.

The online auction simulations demonstrated the ability to add new organizations and new behavior by extending the capabilities of an existing set of distributed intelligent agents simulating implementation in a distributed CPS. Further, the behaviors and communications and messaging protocols were independently configured in a repeatable, extensible manner. Changes to the desired market behaviors have minimal impacts on the previously existing functionality, and specifications for the behavior of the market behaviors remain unaffected by the modifications to the prior functionality (related to managing voltage fluctuations). Therefore, in addition to testing the distributed implementation of a double auction, the

results showed the ability of the multigroup agents to successfully create and participate in new organizations, implement new and independent goal-driven behavior specifications, and successfully manage the addition of new capabilities to support the new objectives.

Additional information about the OAS and the auction trials is provided in Appendix B.

8.1.3 Complex System: Two HHMAS

Attempting to illustrate the participation domains, capabilities, and component types that support a single agent requires some detail. The diagram shown in Figure 8.6 has been developed based on the biological model shown in Figure 3.6[16] to illustrate internal aspects along with relationships to affiliated organizations for an agent acting in two complex systems: a Grid Control System (GCS) for volt-var control and an Online Auction System (OAS) for conducting discrete auctions.

The agent is designed as an organization containing sub-agents as shown in Figure 8.6 shown earlier in this chapter. This figure depicts a neighborhood transformer agent type, A_2, shown in purple.

Neighborhood transformer agents, or neighborhood agents for short, support two different multigroup organizations, the first, O_2 supporting the OAS as shown on the left in green, and a second multigroup organizations, O_1 supporting the GCS as shown on the right in pink. An example of this type of agent was Neighborhood 43 or N43 from the test cases.

The agent has a single self persona shown in the inner purple circle, and three worker persona, shown as inner purple rectangles: one for forecasting, one for getting sensor readings from the smart meter, and one for controlling the smart inverter actuator. The self persona, $s_{2,1}$, will handle the set of parametrized goals that will be used to initialize the agent.

In our experimental systems, neighborhood agents operate in *middle layers* of their holarchies. Neighborhoods support and manage a lowest-level set of homes below them, and neighborhoods operate as parts of local organizations under higher-level single-phase lateral lines or three-phase feeder lines above them.

Thus, the agent has a corresponding affiliation with *two* local groups in each system. The two local groups for the auction system are shown as the two green ovals on the left, and the two local groups for the grid control system are shown as the two ovals on the right. In each system, the two local groups are related.

Therefore, following the AASIS design process, in addition to the self persona and the worker persona, agent $a_{2,1}$ has four affiliate persona, one for each local group in which the agent acts. Each affiliate persona plays one role in the affiliated organization, and this has been depicted by the role the affiliate persona is currently playing in each affiliated organization.

For a more detailed explanation, look first at the upper local group on the left, $o_{2,1,1}$. This local organization, $o_{2,1,1}$, is named o_2 because it is a group in O_2, and $o_{2,1}$ because it is part of the first, or lowest level of organization in the holarchy (that is, it includes homes and neighborhoods). The final subscript indicates that the organization shown is the first group in the lowest level of the auction organization.

The five darkly colored elements represent roles in the local organization, $o_{2,1,1}$. There are four body roles, shown as diamonds, and one head role shown as a triangle in this local organization. The head of this organization is colored purple, like the neighborhood agent, and indicates that this neighborhood agent is currently playing the administrator (head) role in that local organization. The other participant roles in this organization, depicted with green diamonds, are played by the homes supported by the associated transformer.

Similarly, in the green organization shown in the lower left, $o_{2,2,1}$, there are four body roles, shown as diamonds, and one head role shown as a triangle. One of the participants (body) in this organization is colored purple, like the neighborhood agent, and indicates that this neighborhood agent is currently playing a participant (body) role in that local organization.

Because the organizations are related holonically and hierarchically, we can also see that the entire organization $o_{2,1,1}$, is being represented in the higher level organization, $o_{2,2,1}$, by

the transformer agent, as shown with the dotted green lines.

An analogous pair of organizations is shown on the right, representing the agent's affiliation with two local groups in the grid control system. Again, the agent heads one organization while representing that organization in a second, higher-level organization. The grid control roles are shown with the abbreviations for the agents playing them in the test case used for the complex IPDS running both a GCS and OAS.

AASIS was used to implement multiple complex MAS with a set of multigroup agents concurrently operating organizations for both power quality control and online auctions. The trials indicate that the AASIS mechanisms for decomposing systems and designing and implementing agents were *reusable*. This was demonstrated by reusing the standard approach, capability types, and connection, registration, and initialization processes to experiment with a second complex MAS. In addition, the *flexibility* of AASIS to handle two different types of complex MAS was demonstrated. In addition to handling the differences listed in Table 6.1, the AASIS implementation provides for different communication exchanges and protocols for handling registration and inter-agent communication as may be expected when managing systems operated under the direction of different agencies (such as the power company and the associated market organizations).

The $OBAA^{++}$ agent architecture employs the concept of agent *persona* to separate the behavior of the agents in each of its affiliated group while providing a mechanism for coordinating those behaviors internally. Persona provide a way to reduce unnecessary coupling of behavioral reasoning between groups while increasing the cohesion of the behavior reasoning for an agent within a specific group. The use of the $OBAA^{++}$ agent architecture is demonstrated by its use in a holonic MAS designed to provide distributed, intelligent control for PDS.

The IPDS experiment demonstrated the efficacy of the multigroup agents to provide continuous operation for voltage control.

Addition of the online auctioning experiment demonstrated that multigroup agents could

be extended to initiate and conduct discrete online auctions in new multigroup holarchies, extending their functionality without impacting prior, externally-governed tasks from independent multigroup organizations.

The research demonstrated a possible mechanism for enhancing distributed intelligent agents supporting IPDS to include the capabilities and behaviors necessary to create, broker, and bid in online auctions using a two-tier double auction mechanism. Additional work is planned to develop and test the addition of various alternative online power auction algorithms in existing agents running in simulated distributed cyber-physical PDS. Plans for evaluating additional iterative auction solutions may be investigated, along with additional mechanisms for adapting the behavior due to communication delays, agents entering and leaving the local auctions, and responding to potential attempts to manipulate the market based on known (or learned) effects of agent-assisted pricing mechanisms.

8.2 Evaluation of AASIS for Goal Consistency

The complex IPDS test cases demonstrated the flexibility and reusability of the OBAA^{++} architecture and the AASIS framework. Additional tests to evaluate the extensibility to new application domains was desired, along with development and evaluation of architectural aspects and processing algorithms for managing goal consistency among multigroup agents. Thus, a new application domain was used to develop a new set of test cases in a very different problem domain, a university Graduate School Research Lab (GSRL). This domain includes a *professor* with goals to run a research lab and advise students, and a set of *graduate students* who get goals from multiple sources, including goals to assist in the lab, but also from family, friends, and of course, also subject to personal goals for learning and maintaining basic health and quality of life as shown in Figure 8.8[2].

The test cases included two agent types with the following goals (simplified somewhat

[2]http://www.deathbymovies.com/wp-content/uploads/2012/08/agent-smith.jpg

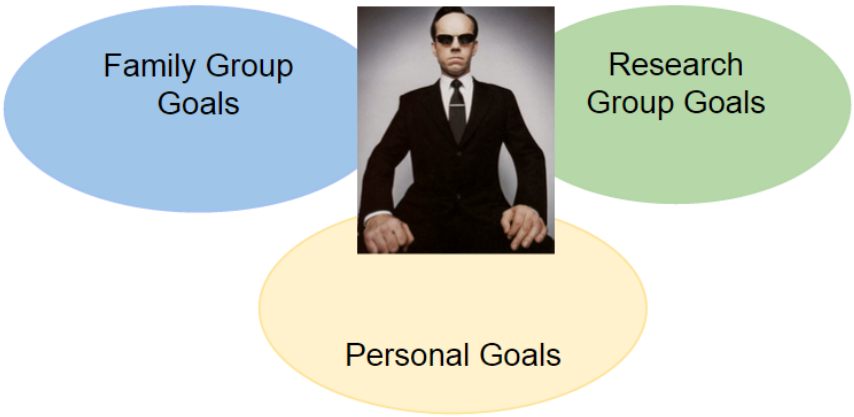


Figure 8.8: *Applying AASIS to a new problem domain: a graduate student getting goal assignments from multiple, independent sources.*

to test the architectural components being developed):

- **Grad Student Agent.** with goals: Work, Study, Eat, Relate, Personal/Play, Sleep.
- **Professor Agent.** with goals: Work, Run Research Lab.

A simple method was used to detect conflicts between various goal types (as characterized by the name or type of goal, rather than the specific parameters, weightings, and temporal aspects as the system runs). Conflicts were defined as shown in Figure 8.9.

Graduate students have critical needs, for example, sleep > 0 and eat > 0. Additional goals were designed such that some could be compatible and performed at the same time (by combining plans), such as eating and studying, some where in conflict (sleeping and

Conflict	Work	Study	Eat	Relate	Personal	Sleep	Accept Pay
Only Work		x	x	x	x	x	
Work		x		x	x	x	
Study	x			x	x	x	
Eat						x	
Relate	x	x				x	
Personal	x	x				x	
Sleep	x	x	x	x	x		
Accept Pay							

Figure 8.9: *Conflict detection table for entities and organizations associated with a graduate school research lab.*

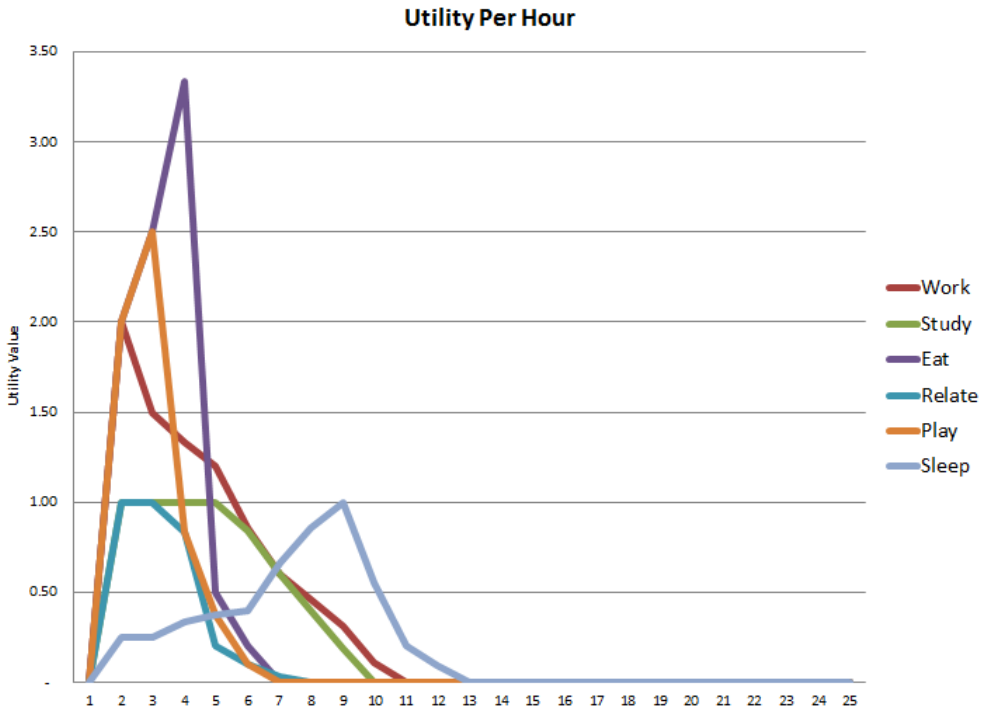


Figure 8.10: *Total utility by hours invested.*

working). Utility functions were developed to characterize the optimal utility per daily hour invested in the pursuit of a particular goal.

For testing, the utility functions shown by total utility over the selected hours and by the average utility per hour are shown in Figure 8.11 and Figure 8.10, respectively. These were used to manage the agent's response to increasing demands from affiliated groups such as research labs or relationships. Additional content can be found in Appendix C.

The following test cases were used to evaluate the goal consistency and biasing architectural supports described in Section 6.13:

- Direct conflict; request rejected.
- Potentially antagonistic goals; complete request accommodated.
- Antagonistic goals; partial request accommodated.
- Non-conflicting, non-antagonistic request accommodated.

In all cases the system performed as expected. The implemented new components provided useful features when designing and implementing complex MAS and provided demonstrable capabilities for detecting conflicts, rejecting or accommodating requests as appropriate and provided some means just as the application of utility functions for meeting or partially meeting requests. Additional work was investigated in the areas of overlapping task assignments and combining plans and is planned for additional research.

The proposed approach was implemented and provides a start on architectural and calculational methods for managing goal consistency in multigroup agents. The work is original and provides a flexible, reusable approach to assist with managing goal consistency. The work is motivated by some of the theoretical research being done around the management of goal consistency. Additional work is needed to implement support for promising theoretical foundations and value and cost-based reasoning, especially in the area of agent learning and the application of stable game theory algorithms for distributed satisfaction problems.

Goal management among multigroup agents is an active area of research and is receiving

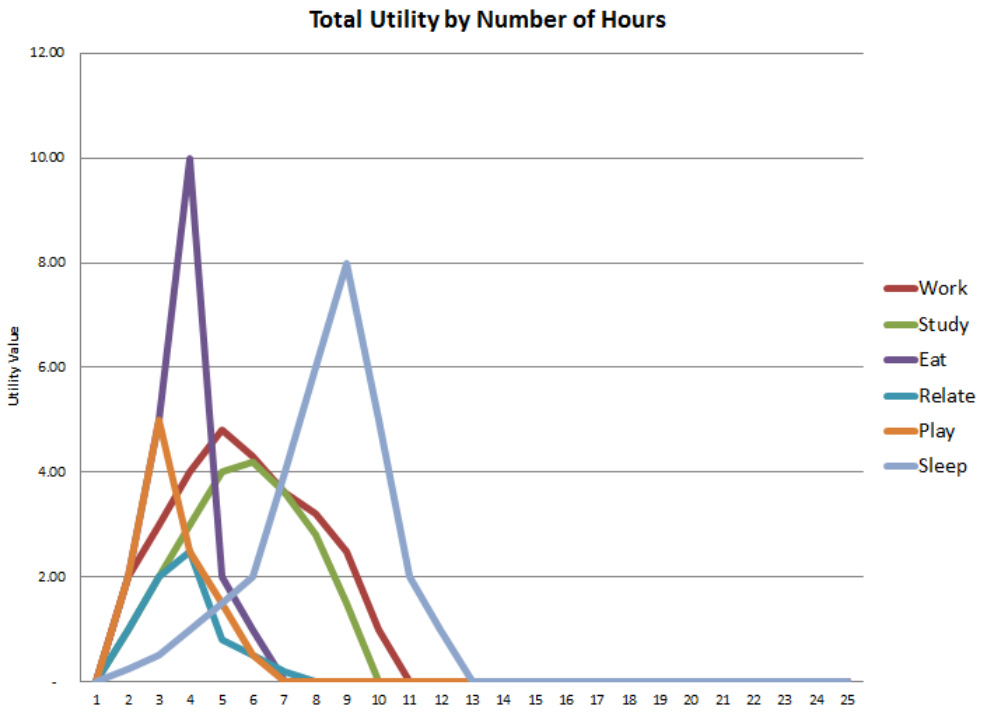


Figure 8.11: *Average utility per hour based on task duration.*

considerable theoretical attention. This work provides a small start on implementing standard architectural mechanisms for supporting goal management. The work lays a promising foundation for implementing and evaluating a variety of game theory algorithms in a cyber-physical multigroup architecture. Agents as personal organizations of subagents that are able to create and administer affiliated organizations appears to hold promise for robust, scaleable, extensible, flexible systems of multigroup agents.

A promising result is the initial quantifiable approach to managing potentially antagonistic goals through the application of utility cost and benefit functions. Correctly designed, the application of community utility can help control systems find stable optimums.

For the test cases described, simple utility functions provided a mechanism for managing potentially antagonistic goals. Figure 8.11 illustrates the total utility by number of resource units (daily hours) spent and Figure 8.10 shows the average utility or value per hour. Biasing mechanisms allow agents to tailor their response to particular goals. Additional work on biasing and utility functions should support the application of temporal considerations, as an agent may be willing to forgo selfishness for community benefit when requested to assist with some goals by not others, and the relative weight of personal versus community benefit may change depending on such as factors as time of day, day of week, season, weather, related pricing structures or other dynamic indicators that may be of interest only to the participating agent rather than to the community as a whole.

8.3 Summary

This chapter discussed complex systems and demonstrated how OBAA^{++} multigroup agents implementing a system with a complex MAS can be further extended to implement *systems of complex MAS*, or as they are called in the framework, *systems of intelligent systems*.

The framework employs intelligent multigroup agents, and each agent is an inner MAS sub-agents called persona, enabling the introduction of special self control capabilities that

can assist with managing goal assignments from multiple groups. The architecture's ability to balance the dual objectives of flexibility and reusability was demonstrated by showing how existing intelligent power distribution agents were implemented with the same architecture, framework, and process to implement a second, goal-driven complex MAS operating concurrently and utilizing the same underlying physical equipment and power distribution system (PDS).

Results showed the multigroup agents were used to simulate and evaluate complex systems, obtaining the same results obtained with centralized implementations of the grid control algorithms and the two-tier double auction algorithms in the MATLAB environment. Together, the GCS and OAS demonstrated a complex IPDS, or a system of intelligent systems, with multigroup agents operating both a complex MAS for online grid control and concurrently, a complex MAS for online auctions.

In addition, testing in a second problem domain, a graduate school research lab (GSRL) was introduced for demonstrating architectural features for goal customization, consistency management, and agent biasing between the demands of potentially competing systems. The conclusions of this work and recommendations for continuing the research are summarized in Chapter 9.

Chapter 9

Conclusions and Recommendations

All Nature bristles with the marks of interrogation –
among the grass and the petals of flowers,
amidst the feathers of birds and the hairs of mammals,
on mountain and moorland, in sea and sky – everywhere.
It is one of the joys of life to discover those marks of interrogation,
these unsolved and half-solved problems and try to answer their questions.
— Sir John Arthur Thomson

This chapter concludes this thesis by summarizing the current state and contributions of the work. Section 9.2 summarizes the current state and open issues. Section 9.3 addresses limitations and Section 9.4 highlights some areas proposed for future work and Section 9.5 acknowledges the funding that helped support the implementation and and evaluation of the architecture. Section 9.6 provides a chapter summary.

9.1 Summary of Contributions

The purpose of this research was to develop a flexible, reusable approach to implementing complex intelligent systems. Specifically, an architecture was developed that allow agents to accept assignments from different groups and manage their multiple associations and assignments in a consistent and unambiguous way.

The work includes the construction of a set of definitions and conceptual model, and the definition of an intelligent system as a goal-driven, reasoning entity. Intelligent systems may be implemented as either a single system with a single set of goals, or may be designed as complex, multigroup systems, where each group functions as a goal-driven entity in support of the system goals. We further defined complex intelligent systems, as systems that integrate two or more systems, each focused on a different problem-solving domain, with goal sets that may compete for resources and processing.

Contributions include the new OBAA^{++} agent architecture for multigroup MAS. OBAA^{++} provides core reusable functionality by designing agents with an inner MAS of sub-agents called persona. A single *self persona* provides advanced reasoning for each multigroup agent, while domain-specific *worker persona* manage sensors, actuators, and associated computations. Flexible *affiliate persona* provide the architectural mechanism for multigroup agents to create, administer, and participate in multiple local groups for a single system, and to create, administer, and participate in forming multiple, multigroup systems concurrently.

OBAA^{++} agents were employed in AASIS, a novel framework for implementing systems of intelligent systems with multigroup agents. AASIS provides an approach to implementing and operating complex systems that is both flexible and reusable. It is reusable enough to work for different types of complex systems, yet flexible enough to allow the necessary customizations required to handle different goals, organizational designs, and organizational decision-making styles.

This work demonstrates that AASIS and the OBAA^{++} architecture balance the dual objectives of flexibility and reusability, and shows the architecture's efficacy by extending

existing intelligent power distribution agents to initiate and operate additional complex organizational structures to support a second, goal-driven HMAS operating concurrently and utilizing the same underlying physical equipment and distribution system.

The IPDS experiment demonstrated the effectiveness of the multigroup agents to operate a multigroup MAS for volt-var control. Addition of the online auctioning functionality in the same agents, demonstrated that multigroup agents could be extended to initiate and conduct discrete online power auctions in new multigroup holarchies, extending their functionality without impacting prior, externally-governed tasks from independent multigroup organizations.

The research demonstrated a possible mechanism for enhancing distributed intelligent agents supporting future power distribution systems to include the capabilities and behaviors necessary to create, broker, and bid in online power auctions using a two-tier double auction mechanism. Additional work is planned to develop and test the addition of various alternative online power auction algorithms in existing agents running in simulated distributed cyber-physical PDS. Plans for evaluating additional iterative auction solutions may be investigated, along with additional mechanisms for adapting the behavior due to communication delays, agents entering and leaving the local auctions, and responding to potential attempts to manipulate the market based on known (or learned) effects of agent-assisted pricing mechanisms.

The work included the development of architectural supports and components for managing goal consistency. In addition to developing and testing the additions for conflict detection and management, the effort provided a second simulation test for the new AASIS framework. The architecture supported the new research lab experiments, and resulted in several enhancements beyond the original system.

The following architectural supports were designed and tested.

- Goal Conflict Management
- Community Bias Management

- Resource Management
- Reasoning with Utility Functions

Application of the AO-MaSE process helped with the potentially pain-staking implementation of complex adaptive systems designed as highly independent, yet interwoven components. Agent-oriented engineering and implementation of goal-driven, organization-based, complex control systems remains challenging, but standard architectures, frameworks, tools, and processes provide valuable assistance for exploring their potential in our evolving smart infrastructure and other intelligent distributed applications.

In the new architecture, each agent implements an *internal organization of sub-agents*. Like other MAS, this internal organization of sub-agents is goal-driven and offers the same architectural support to multigroup agents that have been tested in MAS. This concept of an *agent as an organization* provides the foundation for our approach to managing the complexity of multigroup agents through implementing a reusable, goal-driven agent architecture for building complex systems.

A *reusable multigroup agent architecture* has been developed, along with a framework and recommended software engineering practices, to support the implementation of complex multigroup applications. The architecture has been used to implement multiple, independently-governed organizations within an integrated collection of agents simulating an intelligent power distribution system. The simulation employs agents on a common physical electrical power distribution system (PDS) that operate multiple, multigroup systems concurrently: one for power quality control one for online power auctions.

9.2 Current State

The Adaptive Architecture for Systems of Intelligent Systems (AASIS) was used to implement multiple complex multiagent systems (MAS) in a set of intelligent power distribution

system (IPDS) agents concurrently operating organizations for both power quality control while conducting future power auctions. The trials indicate that the AASIS mechanisms for decomposing systems and designing and implementing agents were *reusable*. This was demonstrated by reusing the standard approach, capability types, and connection, registration, and initialization processes to implement a second complex MAS. In addition, the *flexibility* of AASIS to handle two different types of complex MAS was demonstrated. AASIS handled the differences listed in Table 6.1, and the AASIS implementation provides for different communication exchanges and protocols for handling registration and inter-agent communication as may be expected when managing systems operated under the direction of different agencies (such as the power company and the market organizations).

9.3 Limitations

Managing goal consistency remains a challenging problem. AASIS endows each intelligent agent with advanced organizational mechanisms such as policies and norms; future work includes evaluating these as possible mechanisms to provide greater customization for agents balancing the demands of multiple affiliations. Currently, all multigroup MAS implemented in AASIS have been holarchies. The architecture will be tested with different types of multigroup organizations, including the flat, overlapping groups employed in the state estimation MAS[182,159], and is being considered for application in different problem domains.

9.4 Future Work

The system will be used to test new iterative grid control algorithms including integrated optimal power flow calculations and advanced iterative auction mechanisms. Additional trials will evaluate the impact of message unreliability and delays on the robustness of the new control algorithms. Currently, all multigroup MAS implemented in AASIS have been holarchies. The architecture will be tested with different types of multigroup organizations,

including the flat, overlapping groups employed in the state estimation MAS[182,159], and is being considered for application in different problem domains.

Additional testing of the conflict resolution method could be applied when implementing a holonic multiagent system to assist with demand side management in electrical power distribution systems in the Intelligent Power Distribution System project[7]. Ramchurn, et al., have shown peak demand can be reduced by employing cooperative agent-based control systems in demand side management[183].

Additional application experiments with additional control algorithms proposed for agent-based intelligent power distribution systems are possible[184,185,186] and will include the addition of new actuators, including capacitors on both the single-phase lateral lines and three-phase feeder lines.

Additional work is planned around merged plans that can be conducted simultaneously. For example, the test implementations include goal models focused on the number of daily hours spent. The current limit is 24 daily hour units. The conflict table indicates that the goal *Eat* is in direct conflict with some goals, such as *Sleep*. But does not conflict with goals such as *Study* or *Relate*. In each of these cases of potential overlap, combined plans such as *Study and Eat* were developed. Future work should include a mechanism to extend the total number of daily work units possible by overlapping these plans.

Although OBAA^{++} agents do not use a BDI approach for determining goal assignments, the *CAN* language developed for describing plan bodies[118] provided interesting options for reasoning about goal consistency from our goal names and evaluating whether a similar application of the CAN language for reasoning in multigroup agents is recommended.

AASIS endows each intelligent agent with advanced organizational mechanisms such as policies and norms; proposed future work includes evaluating these as possible mechanisms to provide greater customization for agents balancing the demands of multiple affiliated groups.

Extension of the utility functions is being discussed with project team members from

the Electrical and Computer Engineering department at Kansas State University[1] interested in employing game theoretic algorithms and agent learning. Current application of utility functions is limited to an agents personal goal reasoning. However by combining the total utility for the complex MAS in a manner such as the following, the system's ability to seek and maintain stable community-based optimums could be enhanced as proposed below.

$$U_t = \sum_{i=1}^{n} u_i + \sum_{j=1}^{g} v_j \sum_{k=1}^{p} u_{kj}$$

where U_t = Total Utility

n = number of individual agents

u_i = utility of individual agent i

g = number of affiliated groups

p = number of participants in group j

u_{kj} = utility of individual participant k in group j

v_j = the relative value of group j's success

Reducing unneeded duplication in the specifications is likely to reduce implementation time and have a beneficial impact on system and test case maintenance.

The addition of *actors* within persona may provide benefits. *Actors are essentially well encapsulated active objects, which can only communicate by sending one another immutable messages asynchronously. Whatever state an actor holds internally, it cannot be accessed from outside the actor except by sending a message to the actor and receiving its reply*[2]. The actor model was proposed by Hewitt in 1973 as a mathematical model of concurrent computation with actors as the universal primitives[187]. When an actor receives one of its

[1] http://ece.k-state.edu/

[2] http://www.drdobbs.com/parallel/jvm-concurrency-and-actors-with-gpars/229402193

finite message types, the actor can make decisions, create more actors, send messages, and plan its next response[3].

For example, actors may hold promise for assisting with agent-to-agent negotiations within the context of centrally-managed hierarchical organization. Agent-to-agent negotiations may be helpful for managing goal consistency but may require a broader effort than this initial focus. Some concepts and ideas included in this section are not being suggested for immediate application. However, they may be of interest to later work and are included in an effort to maintain a slightly more complete vision of potentially-beneficial research contributions.

Related work in the development of intelligent cyber-physical systems (CPS) may also provide insights that could be used to enhance goal reasoning. Crucial cyber-physical systems may require significant verification and validation and provide ideas for assessing potential conflicts in distributed artificial intelligence[188,1,189].Testing the architecture for application in other complex, critical, and/or computationally-demanding problem areas is also planned.

Future work is planned for developing additional architectural aspects and processing algorithms for managing goal consistency among multigroup agents supporting the objectives of multiple affiliated groups while operating in concert with differing owner-specified degrees of selfishness and cooperation.

9.5 Special Acknowledgment

The project providing testing and validation of the engineering approach described here was supported by the United States National Science Foundation (NSF) via Award No. CNS-1136040. The NSF provides "*fundamental scientific research that has had a profound impact on our nation's innovation ecosystem and kept our nation at the forefront of the*

[3]http://www.drdobbs.com/parallel/jvm-concurrency-and-actors-with-gpars/229402193

world's science-and-engineering enterprise" [190]; I thank all those who invest and participate in their efforts.

9.6 Summary

This chapter describes the current state of the research with limitations and plans for future work. It summarizes the major contributions, including OBAA^{++}, the new agent architecture for complex MAS, the AASIS framework and system architecture for multigroup agents, and AO-MaSE, the O-MASE compliant-process for designing and implementing complex MAS and systems of systems with multigroup agents.

Reusability was demonstrated by providing:

1. A standard goal-driven multigroup agent architecture.
2. Standard practices for defining desired behavior of complex organizations.
3. Standard practices for implementing agents capable of operating the organizations specified.

Flexibility was demonstrated by:

1. Using the recommended practices and mechanisms to specify the desired behavior for a complex multigroup MAS organization.
2. Using the same practices to specify the desired behavior for a second, independent complex MAS organization operating in a closely-related problem domain.
3. Using the recommended practices and mechanisms to *implement intelligent agents* that operate both multigroup organizations concurrently.
4. Presenting results from experiments implementing the organizations specified with multigroup agents.

Bibliography

[1] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.

[2] Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

[3] Janos Sztipanovits, Xenofon Koutsoukos, Gabor Karsai, Nicholas Kottenstette, Panos Antsaklis, Vijay Gupta, Bill Goodwine, John Baras, and Shige Wang. Toward a science of cyber–physical system integration. *Proceedings of the IEEE*, 100(1):29–44, 2012.

[4] Michael Wooldridgey and Paolo Ciancarini. Agent-oriented software engineering: The state of the art. In *Agent-oriented software engineering*, pages 1–28. Springer, 2001.

[5] Hoon Ko, Goreti Marreiros, Hugo Morais, Zita Vale, and Carlos Ramos. Intelligent supervisory control system for home devices using a cyber physical approach. *Integrated Computer-Aided Engineering*, 19(1):67–79, 2012.

[6] Stefan Grobbelaar and Mihaela Ulieru. Complex networks as control paradigm for complex systems. In *SMC*, pages 4069–4074, 2007.

[7] Anil Pahwa, Scott A Deloach, Sanjoy Das, Bala Natarajan, Xinming Ou, Daniel Andresen, Noel Schulz, Gurdip Singh, Sanjoy Das Bala Natarajan Xinming Ou Daniel Andresen Noel Schulz Anil Pahwa Scott A. DeLoach, and Gurdip Singh. Holonic Multi-agent Control of Power Distribution Systems of the Future. In *CIGRE Grid of the Future Symposium*, October 2012.

[8] Ebisa Negeri and Nico Baken. Architecting the smart grid as a holarchy. In *Proceedings of the 1st International Conference on Smart Grids and Green IT Systems, 19-20 Apr 2012, Porto, Portugal*. SciTePress, 2012.

[9] Massimo Cossentino, Nicolas Gaud, Stéphane Galland, Vincent Hilaire, and Abderrafiâa Koukam. A holonic metamodel for agent-oriented analysis and design. In *Holonic and Multi-Agent Systems for Manufacturing*, pages 237–246. Springer, 2007.

[10] Marco Calabrese, Alberto Amato, Vincenzo Di Lecce, and Vincenzo Piuri. Hierarchical-granularity holonic modelling. *Journal of Ambient Intelligence and Humanized Computing*, 1(3):199–209, 2010.

[11] Danny Weyns and Michael Georgeff. Self-adaptation using multiagent systems. *Software, IEEE*, 27(1):86–91, 2010.

[12] Scott A DeLoach and Juan Carlos Garcia-Ojeda. O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 4(3):244–280, 2010.

[13] Scott DeLoach. OMACS: A framework for adaptive, complex systems. *Handbook of research on multi-agent systems: Semantics and dynamics of organizational models*, pages 76–98, 2009.

[14] Christopher Zhong and Scott A DeLoach. Runtime Models for Automatic Reorganization of Multi-Robot Systems. In *6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011).*, May 2011.

[15] Peter Smith Ring and Andrew H de Ven. Structuring cooperative relationships between organizations. *Strategic management journal*, 13(7):483–498, 1992.

[16] John J Kineman. Relational science: a synthesis. *Axiomathes*, 21(3):393–437, 2011.

[17] Paulo Leitão and Francisco Restivo. Adacor: A holonic architecture for agile and adaptive manufacturing control. *Computers in industry*, 57(2):121–130, 2006.

[18] Massimo Cossentino, Nicolas Gaud, Vincent Hilaire, Stéphane Galland, and Abderrafiâa Koukam. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, 2010.

[19] Ahmad Reza Malekpour, Anil Pahwa, and Sanjoy Das. Inverter-based var control in low voltage distribution systems with rooftop solar pv. In *North American Power Symposium (NAPS), 2013*, pages 1–5. IEEE, 2013.

[20] Julius Sumner Miller. What Science Teaching Needs. *Junior college journal*, 38, 1967.

[21] Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

[22] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, pages 161–166, 2011.

[23] A. Pahwa, S. A. DeLoach, B. Natarajan, S. Das, A. R. Malekpour, S. M. Alam, and D. M. Case. Goal-based holonic multi-agent system for operation of power distribution systems. *IEEE Transactions on Smart Grid*, 2015.

[24] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316, 2004.

[25] Manuel Kolp, Paolo Giorgini, and John Mylopoulos. Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1):3–25, 2006.

[26] Olivier Gutknecht and Jacques Ferber. MadKit: Organizing heterogeneity with groups

in a platform for multiple multi-agent systems. *Rapport Interne LIRMM*, 97188:1997, 1997.

[27] Sam Devlin, Daniel Kudenko, and Marek Grześ. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(02):251–278, 2011.

[28] Zhenhua Jiang. Agent-based control framework for distributed energy resources microgrids. In *Intelligent Agent Technology, 2006. IAT'06. IEEE/WIC/ACM International Conference On*, pages 646–652. IEEE, 2006.

[29] H F Wang. Multi-agent co-ordination for the secondary voltage control in power system contingencies. In *In Proc. IEE Generation, Transmission and Distribution*, pages 61–66. IEEE, January 2001.

[30] Fabrice Lauri, Gillian Basso, Jiawei Zhu, Robin Roche, Vincent Hilaire, and Abderrafiâa Koukam. Managing Power Flows in Microgrids Using Multi-Agent Reinforcement Learning. *Agent Technologies for Energy Systems (ATES)*, 2013.

[31] Manisa Pipattanasomporn, Hassan Feroze, and S Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE, 2009.

[32] Zhe Xiao, Tinghua Li, Ming Huang, Jihong Shi, Jingjing Yang, Jiang Yu, and Wei Wu. Hierarchical MAS based control strategy for microgrid. *Energies*, 3(9):1622–1638, 2010.

[33] Tobias Linnenberg, Ireneus Wior, Sebastian Schreiber, and Alexander Fay. A market-based multi-agent-system for decentralized power and grid control. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8. IEEE, 2011.

[34] Gillian Basso, Vincent Hilaire, Fabrice Lauri, Robin Roche, and Massimo Cossentino. A mas-based simulator for the prototyping of smart grids. In *Proceedings of the 9th European Workshop on Multi-Agent Systems (EUMAS 2011), Maastricht, Netherlands*, pages 1–15, 2011.

[35] Bart De Win, Frank Piessens, Wouter Joosen, and Tine Verhanneman. On the importance of the separation-of-concerns principle in secure software engineering. In *Workshop on the Application of Engineering Principles to System Security Design, WAEPSSD, Boston, MA, USA*, 2002.

[36] Denise M Case and Scott A DeLoach. OBAA++: An Agent Architecture for Participating in Multiple Groups. In *Proceedings of the 13th Intl Conference on Autonomous Agents and Multiagent Systems*, pages 1367–1368. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[37] Denise M Case. Distributed Intelligent Systems: Agent Self Control and Consistency Management in Complex Adaptive Systems. 2014.

[38] Denise M Case and Scott A DeLoach. Applying an O-MaSE Compliant Process to Develop a Holonic Multiagent System for the Evaluation of Intelligent Power Distribution Systems. In *Engineering Multi-Agent Systems*, pages 78–96. Springer, 2013.

[39] Denise M Case. Engineering multigroup agents for complex cooperative systems. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1707–1708. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[40] Denise M Case, M Nazif Faqiry, Bodhisattwa P Majumder, Sanjoy Das, and Scott A DeLoach. Implementation of a Two-tier Double Auction for on-line Power Purchasing in the Simulation of a Distributed Intelligent Cyber-Physical System. In *Research in Computing Science*, 2014.

[41] Alexander Backlund. The definition of system. *Kybernetes*, 29(4):444–451, 2000.

[42] Merriam-Webster Dictionary. System, definition of. http://www.merriam-webster.com/dictionary/system, 2015.

[43] Jacques Ferber. *Multi-Agent Systems: an Introduction to Distributed Artificial Intelligence*, volume 1. Addison-Wesley Reading, 1999.

[44] Adriana Giret and Vicente Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15(5):645–659, 2004.

[45] Intelligence. Intelligence – Wikipedia, the free encyclopedia, 2015. URL http://en.wikipedia.org/wiki/Intelligence. [Online; accessed 17-March-2015].

[46] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25, 1995.

[47] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. From agents to organizations: an organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV*, pages 214–230. Springer, 2004.

[48] Franco Zambonelli, Nicholas R Jennings, and Michael Wooldridge. Developing Multiagent Systems: the Gaia Methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, 2003. ISSN 1049-331X. doi: http://doi.acm.org/10.1145/958961.958963.

[49] Walamitien H Oyenan, Scott A DeLoach, and Gurdip Singh. Exploiting reusable organizations to reduce complexity in multiagent system design. In *Agent-Oriented Software Engineering X*, pages 3–17. Springer, 2011.

[50] Jomi Fred Hübner, Jaime Simao Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Advances in artificial intelligence*, pages 118–128. Springer, 2002.

[51] Herbert A Simon. The Architecture of Complexity. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482, 1962.

[52] Arthur Koestler. *The Ghost in the Machine*. Macmillan, 1968.

[53] Franz Pichler. Modeling complex systems by multi-agent holarchies. In *Computer Aided Systems Theory-EUROCAST'99*, pages 154–168. Springer, 2000.

[54] Bodhisattwa P Majumder, M Nazif Faqiry, Sanjoy Das, and Anil Pahwa. An efficient iterative double auction for energy trading in microgrids. In *Computational Intelligence Applications in Smart Grid (CIASG), 2014 IEEE Symposium on*, pages 1–7. IEEE, 2014.

[55] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12, 2006.

[56] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, and Others. Building Watson: An overview of the DeepQA project. *AI magazine*, 31 (3):59–79, 2010.

[57] Joseph L Jones. Robots at the tipping point: the road to iRobot Roomba. *Robotics & Automation Magazine, IEEE*, 13(1):76–78, 2006.

[58] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

[59] UNSW CSE. 2014 RoboCup SPL Grand Final: rUNSWift Vs HTWK. https://www.youtube.com/watch?v=dhooVgC_0eY, 2014.

[60] John Markoff. Virtual and Artificial, but 58,000 Want Course. http://www.nytimes.com/2011/08/16/science/16stanford.html?_r=0, 2011.

[61] Udacity. Intro to Artificial Intelligence. https://www.udacity.com/course/cs271, 2014.

[62] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1): 139–159, 1991.

[63] Alan H Bond and Les Gasser. A Survey of Distributed Artificial Intelligence, 1988.

[64] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[65] Serena Chan. Complex adaptive systems. In *ESD. 83 Research Seminar in Engineering Systems*, 2001.

[66] John H Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314, 1962.

[67] Eve Mitleton-Kelly. Organisation as co-evolving complex adaptive systems. 1997.

[68] John H Holland. Complex adaptive systems. *Daedalus*, pages 17–30, 1992.

[69] Festo. Technology Overview. BionicANTs: Cooperative behaviour based on natural model . http://www.festo.com/cms/en_corp/14252.htm, 2015.

[70] Festo. BionicANTs: Cooperative behaviour based on a natural model . http://www.festo.com/net/SupportPortal/Files/367917/Festo_BionicAnts_en.pdf, 2015.

[71] New England Complex Systems Institute. Concepts: Adaptive. http://www.necsi.edu/guide/concepts/adaptive.html, 2014.

[72] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2): 14, 2009.

[73] Robert Laddaga. Guest Editor's Introduction: Creating Robust Software through Self-Adaptation. *IEEE Intelligent Systems*, 14(3):26–29, 1999.

[74] Peyman Oreizy, Michael M Gorlick, Richard N Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S Rosenblum, and Alexander L Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent systems*, 14(3):54–62, 1999.

[75] Jeffrey O Kephart and David M Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, 2003. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/MC.2003.1160055.

[76] Ozalp Babaoglu. *Self-star properties in complex information systems: conceptual and practical foundations*, volume 3460. Springer, 2005.

[77] Gerard J Foschini and Zoran Miljanic. A simple distributed autonomous power control algorithm and its convergence. *Vehicular Technology, IEEE Transactions on*, 42(4): 641–646, 1993.

[78] Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou, and Gauthier Picard. Adelfe: A methodology for adaptive multi-agent systems engineering. In *Engineering Societies in the Agents World III*, pages 156–169. Springer, 2003.

[79] Estefanía Argente, Vicent Botti, and Vicente Julian. Gormas: An organizational-oriented methodological guideline for open mas. In *Agent-Oriented Software Engineering X*, pages 32–47. Springer, 2011.

[80] Juan Pavón and Jorge Gómez-Sanz. Agent oriented software engineering with INGENIAS. In *Multi-Agent Systems and Applications III*, pages 394–403. Springer, 2003.

[81] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. Moise+: towards a structural, functional, and deontic model for mas organization. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 501–502. ACM, 2002.

[82] Virginia Dignum, Javier Vázquez-Salceda, and Frank Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems*, pages 181–198. Springer, 2005.

[83] Virginia Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, University of Utrecht, Utrecht, The Netherlands, 2003.

[84] Huib Aldewereld and Virginia Dignum. OperettA: Organization-oriented development environment. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, pages 1–18. Springer, 2011.

[85] Massimo Cossentino. From requirements to code with the PASSI methodology. *Agent-oriented methodologies*, 4:79–106, 2005.

[86] Lin Padgham and Michael Winikoff. *Developing intelligent agent systems: A practical guide*, volume 13. Wiley. com, 2005.

[87] Emilia Garcia, Adriana Giret, and Vicente J Botti. Developing Regulated Open Multi-agent Systems. In *AT*, pages 12–26, 2012.

[88] Ambra Molesini, Andrea Omicini, Enrico Denti, and Alessandro Ricci. SODA: A roadmap to artefacts. In *Engineering societies in the agents world VI*, pages 49–62. Springer, 2006.

[89] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. TROPOS: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8:203–236, May 2004.

[90] Scott A DeLoach, Lin Padgham, Anna Perini, and Angelo Susi. Using three AOSE toolkits to develop a sample design. *International Journal of Agent-Oriented Software Engineering*, 3(4):416–476, 2009.

[91] Scott A Deloach and Matthew Miller. A Goal Model for Adaptive Complex Systems. 5(2), 2010.

[92] John Thangarajah, Lin Padgham, and Michael Winikoff. Detecting and Avoiding Interference Between Goals in Intelligent Agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.

[93] Michael Winikoff. Future directions for agent-based software engineering. *International Journal of Agent-Oriented Software Engineering*, 3(4):402–410, 2009.

[94] David Isern, David Sánchez, and Antonio Moreno. Organizational structures supported by agent-oriented methodologies. *Journal of Systems and Software*, 84(2): 169–184, 2011.

[95] Juan C Garcia-Ojeda, Scott A DeLoach, Walamitien H Oyenan, Jorge Valenzuela, et al. *O-MaSE: a customizable approach to developing multiagent development processes*. Springer, 2008.

[96] Scott J Harmon, Scott A DeLoach, et al. Guidance and law policies in multiagent systems. Technical report, DTIC Document, 2007.

[97] RafałLeszczyna. Evaluation of agent platforms. 2004.

[98] AgentBuilder. AgentBuilder Website. https://www.agentbuilder.com, 2014.

[99] Multiagent and Cooperative Robotics (MACR) Laboratory at Kansas State University. agentTool3 Website. https://http://agenttool.cis.ksu.edu/, 2014.

[100] Aaron Helsinger, Michael Thome, and Todd Wright. Cougaar: a scalable, distributed multi-agent architecture. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1910–1917. IEEE, 2004.

[101] Massimo Cossentino, Daniele Dalle Nogare, Raffaele Giancarlo, Carmelo Lodato, Salvatore Lopes, Patrizia Ribino, Luca Sabatucci, and Valeria Seidita. GIMT: A tool for ontology and goal modeling in BDI Multi-Agent Design. 2014.

[102] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. JACK intelligent agents-summary of an agent infrastructure. In *5th International conference on autonomous agents*, 2001.

[103] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with jade. In *Intelligent Agents VII Agent Theories Architectures and Languages*, pages 89–103. Springer, 2001.

[104] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.

[105] Vladimir Gorodetsky, Oleg Karsaev, Victor Konushy, and Vladimir Samoilov. Model-Driven Engineering of Multi Agent Systems. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 2, pages 83–86. IEEE, 2008.

[106] Piermarco Burrafato and Massimo Cossentino. Designing a multi-agent solution for a bookstore with the passi methodology. In *AOIS CAiSE*, 2002.

[107] David Morley and Karen Myers. The SPARK agent framework. In *Proceedings of the*

Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 714–721. IEEE Computer Society, 2004.

[108] Fausto Giunchiglia, John Mylopoulos, and Anna Perini. The tropos software development methodology: processes, models and diagrams. In *Agent-Oriented Software Engineering III*, pages 162–173. Springer, 2003.

[109] Hyacinth S Nwana, Divine T Ndumu, Lyndon C Lee, and Jaron C Collis. Zeus: a toolkit for building distributed multiagent systems. *Applied Artificial Intelligence*, 13 (1-2):129–185, 1999.

[110] Sylvanus Jenkins. Comparative Study of Multi-agent Development Toolkits. Master's thesis, University of Greenwich, 2011.

[111] Rafael H Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah-Seghrouchni, Jorge J Gomez-Sanz, Joao Leite, Gregory M P O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)*, 30(1):33–44, 2006.

[112] Haysam Alsayed, Lucian Aron, Tatyana Rabinovitch, Brooks Riley, and Michel Fattouche. MAS-UML-MODELER GROUP 5. 2007.

[113] Lin Padgham, Michael Winikoff, Scott DeLoach, and Massimo Cossentino. A unified graphical notation for AOSE. In *Agent-Oriented Software Engineering IX*, pages 116–130. Springer, 2009.

[114] Michael Winikoff, Lin Padgham, James Harland, and John Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, 2002.

[115] Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric Yu. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25(8):841–877, 2010.

[116] John Thangarajah, Michael Winikoff, Lin Padgham, and Klaus Fischer. Avoiding resource conflicts in Intelligent agents. In F van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, 2002.

[117] Mark Klein. Supporting conflict resolution in cooperative design systems. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(6):1379–1390, 1991.

[118] James Harland, David N Morley, John Thangarajah, and Neil Yorke-Smith. An operational semantics for the goal life-cycle in BDI agents. *Autonomous Agents and Multi-Agent Systems*, pages 1–38, 2013.

[119] John Thangarajah, James Harland, David Morley, and Neil Yorke-Smith. Operational behaviour for executing, suspending, and aborting goals in BDI agent systems. In *Declarative Agent Languages and Technologies VIII*, pages 1–21. Springer, 2011.

[120] Katie Atkinson, Trevor Bench-Capon, and Peter Mcburney. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems*, 11(2):153–171, 2005.

[121] Scott J Harmon, Scott A DeLoach, Robby, and Doina Caragea. Leveraging Organizational Guidance Policies with Learning to Self-Tune Multiagent Systems. In *The Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, October 2008.

[122] Scott J Harmon, Scott A DeLoach, and Others. Trace-Based Specification of Law and Guidance Policies for Multi-Agent Systems. In *Engineering Societies in the Agents World VIII*, pages 333–349. Springer, 2008.

[123] Carl Folke, Thomas Hahn, Per Olsson, and Jon Norberg. Adaptive governance of social-ecological systems. *Annu. Rev. Environ. Resour.*, 30:441–473, 2005.

[124] Ross Anderson. *Security engineering*. John Wiley & Sons, 2008.

[125] Adam Langley. Maintaining digital certificate security. http://googleonlinesecurity.blogspot.com/2015/03/maintaining-digital-certificate-security.html, 2015.

[126] Vijay Prakash and Manuj Darbari. A new framework of distributed system security using dna cryptography and trust based approach. 2014.

[127] Eric Chan-Tin, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. The frog-boiling attack: Limitations of secure network coordinate systems. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):27, 2011.

[128] H Lee Willis. *Distributed power generation: planning and evaluation*. CRC Press, 2000.

[129] Hassan Feroze. *Multi-agent systems in microgrids: Design and implementation*. PhD thesis, Virginia Polytechnic Institute and State University, 2009.

[130] Silviu Ionita. Multi Agent Holonic Based Architecture for Communication and Learning about Power Demand in Residential Areas. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 644–649. IEEE, 2009.

[131] Zhenhua Jiang. Agent-based power sharing scheme for active hybrid power sources. *Journal of power sources*, 177(1):231–238, 2008.

[132] H Asano. Holonic Energy Systems: Coevolution of Distributed Energy Resources and Existing Network Energy. In *International Symposium on Distributed Energy Systems and Micro Grids*, 2005.

[133] Morgan Bazilian, Ijeoma Onyeji, Michael Liebreich, Ian MacGill, Jennifer Chase, Jigar Shah, Dolf Gielen, Doug Arent, Doug Landfear, and Shi Zhengrong. Re-considering the economics of photovoltaic power. *Renewable Energy*, 53:329–338, 2013.

[134] L K Wiginton, H T Nguyen, and Joshua M Pearce. Quantifying rooftop solar photovoltaic potential for regional renewable energy policy. *Computers, Environment and Urban Systems*, 34(4):345–357, 2010.

[135] Robert Wilson. Architecture of power markets. *Econometrica*, 70(4):1299–1340, 2002.

[136] Pattie Maes, Robert H Guttman, and Alexandros G Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, 1999.

[137] Ioannis S Baxevanos and Dimitris P Labridis. Implementing multiagent systems technology for power distribution network control and protection management. *Power Delivery, IEEE Transactions on*, 22(1):433–443, 2007.

[138] Kai Huang, David A Cartes, and Sanjeev K Srivastava. A multiagent-based algorithm for ring-structured shipboard power system reconfiguration. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(5):1016–1021, 2007.

[139] Hak-Man Kim and T Kinoshita. Multiagent system for Microgrid operation based on power market environment. In *Telecommunications Energy Conference, 2009. INTELEC 2009. 31st International*, pages 1–5. IEEE, 2009.

[140] Ahmad Reza Malekpour and Anil Pahwa. Reactive power and voltage control in distribution systems with photovoltaic generation. In *North American Power Symposium (NAPS), 2012*, pages 1–6. IEEE, 2012.

[141] V Vishwanathan, J McCalley, and V Honavar. A multiagent system infrastructure

and negotiation framework for electric power systems. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 1, pages 6—-pp. IEEE, 2001.

[142] I Zabet and M Montazeri. Decentralized control and management systems for power industry via multiagent systems technology. In *Power Engineering and Optimization Conference (PEOCO), 2010 4th International*, pages 549–556. IEEE, 2010.

[143] Zhong Zhang, James D McCalley, Vijay Vishwanathan, and Vasant Honavar. Multiagent system solutions for distributed computing, communications, and data integration needs in the power industry. In *Power Engineering Society General Meeting, 2004. IEEE*, pages 45–49. IEEE, 2004.

[144] Stuart Russell. Artificial Intelligence: A Modern Approach Author: Stuart Russell, Peter Norvig, Publisher: Prentice Hall Pa. 2009.

[145] Massimo Cossentino, Stéphane Galland, Nicolas Gaud, Vincent Hilaire, and Abderrafiâa Koukam. How to control emergence of behaviours in a holarchy. In *Self-Adaptive and Self-Organizing Systems Workshops. 2nd IEEE Intl Conf on*, pages 180–185. IEEE, 2008.

[146] Sei Ping Lau, Geoff V Merrett, and Neil M White. Energy-efficient street lighting through embedded adaptive intelligence. In *Advanced Logistics and Transport (ICALT), 2013 International Conference on*, pages 53–58. IEEE, 2013.

[147] Hendrik Van Brussel, Jo Wyns, Paul Valckenaers, Luc Bongaerts, and Patrick Peeters. Reference architecture for holonic manufacturing systems: Prosa. *Computers in industry*, 37(3):255–274, 1998.

[148] Mahshid Helali Moghadam and Nasser Mozayani. A street lighting control system based on holonic structures and traffic system. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, volume 1, pages 92–96. IEEE, 2011.

[149] Caroline E Harriott, Rui Zhuang, Julie A Adams, and Scott A DeLoach. Towards using human performance moderator functions in human-robot teams. In *First International Workshop on Human-Agent Interaction Design and Models*, 2012.

[150] Scott A DeLoach. Engineering organization-based multiagent systems. In *Software Engineering for Multi-Agent Systems IV*, pages 109–125. Springer, 2006.

[151] Jorge Valenzuela. *DTAACS: Distributed Task Allocation for Adaptive Computational Systems*. PhD thesis, Kansas State University, Kansas, United States of America, 2014.

[152] Walt Whitman. *Leaves of grass*. GP Putnam's sons, 1907.

[153] Jaime Simão Sichman, Virginia Dignum, and Cristiano Castelfranchi. Agents' organizations: a concise overview. *Journal of the Brazilian Computer Society*, 11(1):3–8, 2005.

[154] N R Jennings, K P Sycara, and M Wooldridge. A Roadmap of Agent Research and Development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–36, 1998. URL Intro/jaamas98.pdf.

[155] Katia P Sycara. Multiagent systems. *AI magazine*, 19(2):79, 1998.

[156] Kansas State University Computing and Information Sciences Department. Holonic Multi-Agent Control of Intelligent Power Distribution Systems Project Site. http://ipds.cis.ksu.edu/, 2015.

[157] Nadeem Aslam. *The wasted vigil*. Anchor Canada, 2009.

[158] Arthur Koestler. The ghost in the machine. 1989.

[159] Anil S M Shafiul Alam, Balasubramaniam Natarajan. Distribution grid state estimation from compressed measurements. *IEEE Transactions on Smart Grid*, 5(4): 1631–1642, 2014.

[160] Rodrigo Canales. Weaving Straw into Gold: Rule Bending, Localism, and Managing Inconsistencies in Organizational Rules. *Localism, and Managing Inconsistencies in Organizational Rules (June 10, 2010)*, 2010.

[161] David J Leinweber. *Nerds on Wall Street: Math, machines and wired markets*. John Wiley and Sons, 2009.

[162] Zhi Zhou, Wai Kin Victor Chan, and Joe H Chow. Agent-based simulation of electricity markets: a survey of tools. *Artificial Intelligence Review*, 28(4):305–342, 2007.

[163] Nicholas R Jennings. On agent-based software engineering. *Artificial intelligence*, 117 (2):277–296, 2000.

[164] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[165] Antonio Chella, Massimo Cossentino, Luca Sabatucci, and Valeria Seidita. Agile PASSI: An agile process for designing agents. *International Journal of Computer Systems Science & Engineering*, 21(2):133–144, 2006.

[166] Alma M Gómez-Rodríguez and Juan C González-Moreno. Comparing agile processes for agent oriented software engineering. In *Product-Focused Software Process Improvement*, pages 206–219. Springer, 2010.

[167] John Gall and Robert O Blechman. *Systemantics: how systems work and especially how they fail*. Quadrangle, 1977.

[168] IEEE PES. IEEE PES Distribution Test Feeders. http://ewh.ieee.org/soc/pes/dsacom/testfeeders/, 2014.

[169] John O'Hara. Toward a commodity enterprise middleware. *Queue*, 5(4):48–55, 2007.

[170] Xiaobo Zhou, Yu Cai, C Edward Chow, and Marijke Augusteijn. Two-tier resource allocation for slowdown differentiation on server clusters. In *Parallel Processing, 2005. ICPP 2005. International Conference on*, pages 31–38. IEEE, 2005.

[171] Amr Abdelnasser, Ekram Hossain, and D Kim. Clustering and Resource Allocation for Dense Femtocells in a Two-Tier Cellular OFDMA Network. 2014.

[172] Baisravan HomChaudhuri, Manish Kumar, and Vijay Devabhaktuni. Market based approach for solving optimal power flow problem in smart grid. In *American Control Conference (ACC), 2012*, pages 3095–3100. IEEE, 2012.

[173] HuaXing Chen and Hoong Chuin Lau. Decentralized resource allocation and scheduling via walrasian auctions with negotiable agents. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 2, pages 356–360. IEEE, 2010.

[174] ShiGuang Wang, Ping Xu, XiaoHua Xu, ShaoJie Tang, XiangYang Li, and Xin Liu. TODA: truthful online double auction for spectrum allocation in wireless networks. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–10. IEEE, 2010.

[175] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[176] Chuliang Weng, Xinda Lu, Guangtao Xue, Qianni Deng, and Minglu Li. A double auction mechanism for resource allocation on grid computing systems. In *Grid and Cooperative Computing-GCC 2004*, pages 269–276. Springer, 2004.

[177] Yunpeng Wang, Walid Saad, Zhu Han, H Vincent Poor, and Tamer Basar. A game-theoretic approach to energy trading in the smart grid. *Smart Grid, IEEE Transactions on*, 5(3):1439–1450, 2014.

[178] Xiaojun Feng, Yanjiao Chen, Jin Zhang, Qian Zhang, and Bo Li. Tahes: A truthful double auction mechanism for heterogeneous spectrums. *Wireless Communications, IEEE Transactions on*, 11(11):4038–4047, 2012.

[179] Tan Le, Mihaela Beluri, Martino Freda, Jean-Louis Gauvreau, Scott Laughlin, and Pekka Ojanen. On a new incentive and market based framework for multi-tier shared spectrum access systems. In *Dynamic Spectrum Access Networks (DYSPAN), 2014 IEEE International Symposium on*, pages 477–488. IEEE, 2014.

[180] Pu Huang, Alan Scheller-Wolf, and Katia Sycara. Design of a Multi–Unit Double Auction E–Market. *Computational Intelligence*, 18(4):596–617, 2002.

[181] M Faqiry, Mukherjeeand Kunduand, Sanjoy Das, and Anil Pahwa. Game Theoretic Model of Energy Trading Strategies at Equilibrium in Microgrids. *NAPS 2014*, 29, 2014.

[182] Vassilis Kekatos and Georgios B Giannakis. Distributed robust power system state estimation. *Power Systems, IEEE Transactions on*, 28(2):1617–1626, 2013.

[183] Sarvapali D Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nick Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[184] H E Z Farag, Ehab F El-Saadany, and Ravi Seethapathy. A two ways communication-based distributed control for voltage regulation in smart distribution feeders. *Smart Grid, IEEE Transactions on*, 3(1):271–281, 2012.

[185] Hany E Z Farag and Ehab F El-Saadany. A Novel Cooperative Protocol for Distributed Voltage Control in Active Distribution Systems. 2013.

[186] Fenghui Ren, Minjie Zhang, and Danny Sutanto. A Multi-Agent Solution to Distribution System Management by Considering Distributed Generators. 2012.

[187] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd international joint conference on Artificial intelligence*, pages 235–245. Morgan Kaufmann Publishers Inc., 1973.

[188] Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alex Roederer, and Others. Challenges and Research Directions in Medical Cyber–Physical Systems. *Proceedings of the IEEE*, 100(1):75–90, 2012.

[189] John Hatcliff, Eugene Vasserman, Sandy Weininger, and Julian Goldman. An overview of regulatory and trust issues for the integrated clinical environment. *Proceedings of HCMDSS 2011*, 2011.

[190] Dr. France A. Cordova. National Science Foundation, Where Discoveries Begin. http://www.nsf.gov/about/congress/reports/where_discoveries.pdf, 2014.

[191] Bill Bryson. *At home: A short history of private life*. Random House LLC, 2010.

[192] Siddharth Deshmukh, Balasubramaniam Natarajan, and Anil Pahwa. Voltage/VAR control in distribution networks via reactive power injection through distributed generators. *Smart Grid, IEEE Transactions on*, 3(3):1226–1234, 2012.

[193] Purdue University. How To Write A Dissertation or Bedtime Reading For People Who Do Not Have Time To Sleep. https://www.cs.purdue.edu/homes/dec/essay.dissertation.html, 2015.

[194] Juan C Garcia-Ojeda, Scott A DeLoach, and Others. agentTool process editor: supporting the design of tailored agent-based processes. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 707–714. ACM, 2009.

[195] Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46, Dec 2011. doi: 10.1109/IEEESTD.2011.6129467.

[196] Hwa-Young Jeong, Mohammad S Obaidat, Neil Y Yen, and James J Jong Hyuk Park. *Advances in Computer Science and its Applications: CSA 2013*, volume 279. Springer Science & Business Media, 2013.

[197] Federal Energy Regulatory Commission et al. Assessment of demand response and advanced metering staff report. 2010. URL http://www.ferc.gov/legal/staff-reports/2010-dr-report.pdf.

[198] US National Science Foundation. US National Science Foundation. http://www.nsf.gov/, 2015.

[199] US DOE. US DOE Smart Grid System Report. https://www.smartgrid.gov/sites/default/files/pdfs/sgsr_main.pdf, 2009.

[200] EPRI. EPRI IntelliGrid Initiative. http://intelligrid.epri.com/, 2015.

[201] Piero Mella. The holonic revolution holons, holarchies and holonic networks, the ghost in the production machine, 2009.

[202] Juan Manuel Carrasco, Leopoldo Garcia Franquelo, Jan T Bialasiewicz, Eduardo Galván, RC Portillo Guisado, Ma AM Prats, José Ignacio León, and Narciso Moreno-Alfonso. Power-electronic systems for the grid integration of renewable energy sources: A survey. *Industrial Electronics, IEEE Transactions on*, 53(4):1002–1016, 2006.

[203] Richard Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[204] Ronald A Howard. Dynamic programming and markov processes. 1960.

[205] Markov decision process. Markov decision process – Wikipedia, the free encyclopedia, 2015. URL http://en.wikipedia.org/wiki/Markov_decision_process. [Online; accessed 22-March-2015].

[206] Microgrid Institute. Microgrid Institute: About Microgrids. http://www.microgridinstitute.org/about-microgrids.html, 2015.

[207] PA Morris, R Cedolin, and CD Feinstein. Reliability of electric utility distribution systems. *EPRI White Paper*, 1000424, 2000.

[208] Len Bass. *Software architecture in practice*. Pearson Education India, 2007.

[209] Paul Clements, Rick Kazman, and Mark Klein. *Evaluating software architectures*. 2003.

[210] Glenn Vanderburg. Qcon keynote: Real software engineering, 2012. URL http://www.infoq.com/presentations/Software-Engineering. "[Online; accessed 28-March-2015]".

[211] US DOE. US DOE Smart Grid. http://www.smartgrid.gov/, 2015.

[212] Smart Meter. Smart meter – Wikipedia, the free encyclopedia, 2015. URL http://en.wikipedia.org/wiki/Smart_meter. [Online; accessed 22-March-2015].

[213] Smart Distribution Wiki. Smart Distribution Wiki: Volt and Var Control and Optimisation. http://wiki.powerdistributionresearch.com/index.php?title=Volt_and_Var_Control_and_Optimisation, 2015.

Appendix A

IPDS Grid Control System (GCS)

The world is not composed of atoms or symbols or cells or concepts.
It is composed of holons.
— Ken Wilber

The $OBAA^{++}$ agent architecture and the integrated multigroup AASIS framework was used as the basis for implementing an intelligent power distribution system (IPDS) project that aims to evaluate distributed control algorithms for electrical power distribution systems. While the project simulates the PDS using MATLAB, our objective in designing the cyber architecture is to make it realistic so that could be used in a true cyber-physical deployment where the computation is distributed within the system.

The IPDS assumes there are a certain number of homes in the system with rooftop solar PV panels. In general, at midday with full sun, PV panels can provide enough power to supply multiple homes. While PV power provides important socio-economic benefits, at high penetrations, it can be problematic in terms of power quality and voltage.

When the solar generation rises and falls as during a day with full sun, no power quality issues are introduced and the electricity flows from the substation, and then increasingly

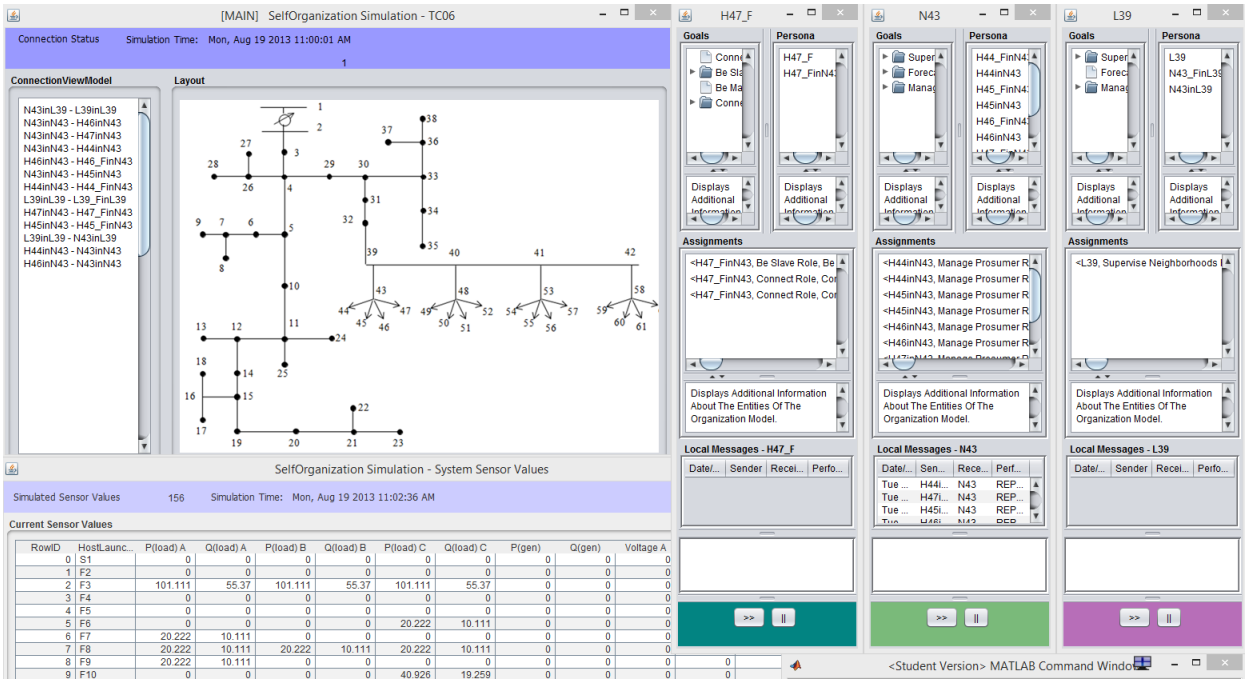


Figure A.1: *Recursively-optimized IPDS simulation.*

from the PV-enabled homes as the sunlight increases. Similarly towards evening, the flow of electricity again comes increasingly from the substation as the sun sets. These sources of distributed renewable generation can bring significant socioeconomic benefits, and smaller PV penetrations (the percent of homes with PV panels) can be handled with existing controls. However, the flow of electricity from PV panels is subject to significant intermittency when, for example, fast moving clouds introduce rapid variations in the amount of generation produced as shown in Figure A.2[1]. When power changes rapidly, voltage variations can be introduced that cause power quality issues—lights can flicker and sensitive electronics could be damaged. At higher penetration levels, this creates a significant problem for the power companies.

A notional example of the IPDS architecture is shown in Figure A.3 (self persona are omitted due to limited space). In the bottom right of the figure, a neighborhood organization with eight different agents is shown. There is one Transformer agent, four Home agents, and three Forecaster agents. Typically, there is a Home agent for each home in the

[1]https://www.egauge.net/

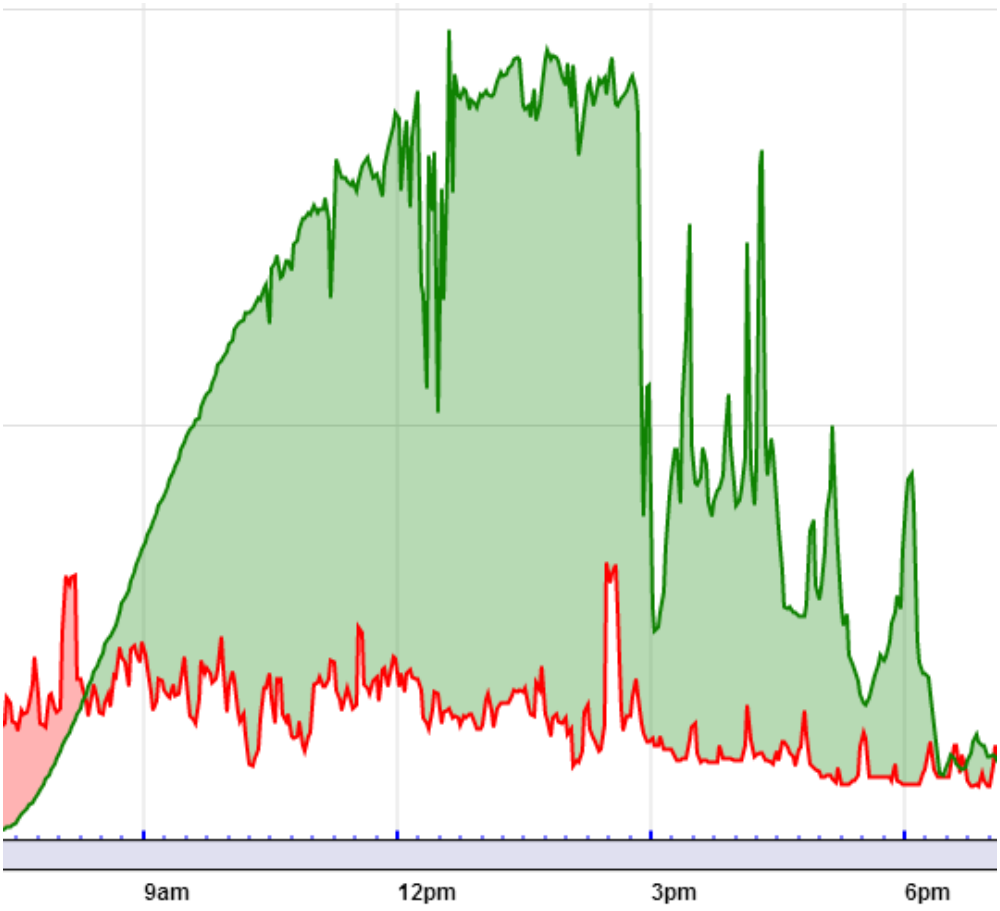


Figure A.2: *Variability in distributed generation from a rooftop solar photovoltaic (PV) installation.*

neighborhood, with a Forecaster agent for each home with a PV panel. There is also a single Transformer agent, which is typically co-located with its own Forecaster agent. Notice that the Transformer agent has two persona, one that is the master of its Neighborhood organization and one that represents its neighborhood in its parent Lateral organization. In total, the Lateral organization has four Transformer agents in it that represent neighborhoods, along with a Forecaster agent and the Lateral agent, which acts as the master. Notice that the Lateral agent has two persona, one of which represents the Lateral organization in the parent Feeder organization.

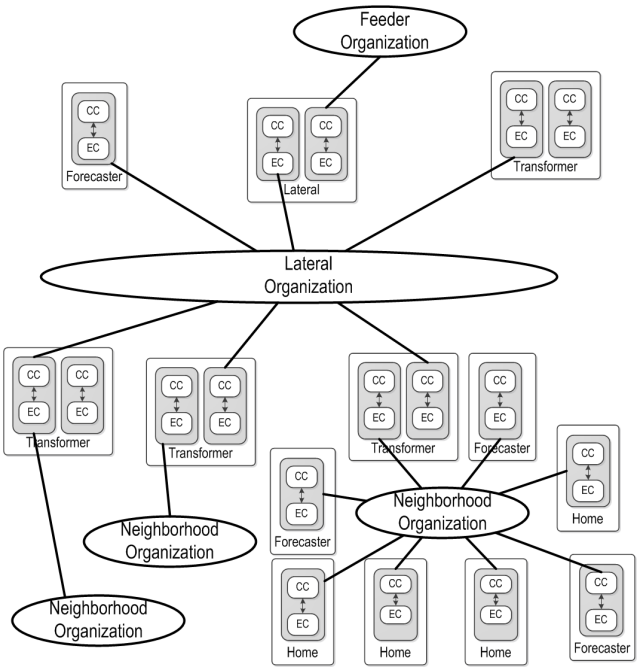


Figure A.3: *Partial architecture for an IPDS test case (not all agents are shown for each organization).*

A.1 HMAS for PDS Grid Control Requirements

Our approach uses holonic design principles to formulate the control problem and develop a computational architecture appropriate for intelligent power distribution systems. Power distribution systems, and cyberphysical systems in general, can use this holonic approach when the associated physical system can be recursively decomposed from the top-level (a super-holon reflecting the entire system) into set of sub-systems, eventually resulting in the lowest level sub-systems that consist of the low-level physical devices. In a power distribution system, the top level is a substation, while the lowest level devices (agents) represent the individual consumers, or homes.

An example of modeling a power distribution system in this way is shown in the three-level holarchy of Figure 2. Each white oval encapsulates an organization, or group of agents working together, while the grey ovals encapsulate a level in the system, i.e., the substation, feeder, and neighborhood levels. Each node labeled with a number represents an agent in that particular organization. Each agent at one level may actually be composed of several

agents at the next lower level, which may again be composed of agents at next lower level. Atomic (non-decomposed) agents may exist at any level.

As illustrated in Figure 3.7, there are four levels in the IPDS: Substation, Feeder, Lateral, and Neighborhood. Each level consists of a set of organizations designed for that level. The PDS control system is designed to support multiple objectives depending on the state of the environment. These include (1) improving efficiency during normal operation, (2) managing power quality during intermittent cloud cover, or (3) supplying local power to critical loads during periods when disconnected from the rest of the PDS. During typical operation the overall system goal is to improve efficiency, but during periods of intermittent cloud cover, the rapid rise and fall of PV generation can result in power quality issues and thus the overall goal switches to maintaining power quality. These objectives are communicated between layers through a set of goals and the parameters of those goals.

For the IPDS, each organization at the same level is the same *type* of organization, each populated with different agents based on the physical configuration of the PDS. As the organizations cooperate towards the achievement of their goals, these goals become the chief control and feedback mechanism within the system. For instance, at level n, the system may only have access to p kW of power and thus it would have the goal of efficiently distributing p kW of power. Instead of dividing p evenly among the agents (the sub-systems) for distribution, the agents can negotiate amongst themselves to determine exactly how best to distribute the power based on the needs of the agents (sub-systems). Thus each agent at level n would be assigned the goal of efficiently distributing its negotiated amount p_i of power where $p = \sum p_i$. Each organization attempts to achieve its overall goal by decomposing its goal into individual goals that are assigned to agents in the organization.

In the IPDS, one agent is assigned to the *master* role in each organization. The master receives the organization's goals from its parent organization and decomposes them and assigns them to other local agents. In addition, the master represents its local organization in its parent organization, where it can negotiate with other agents for the redistribution of

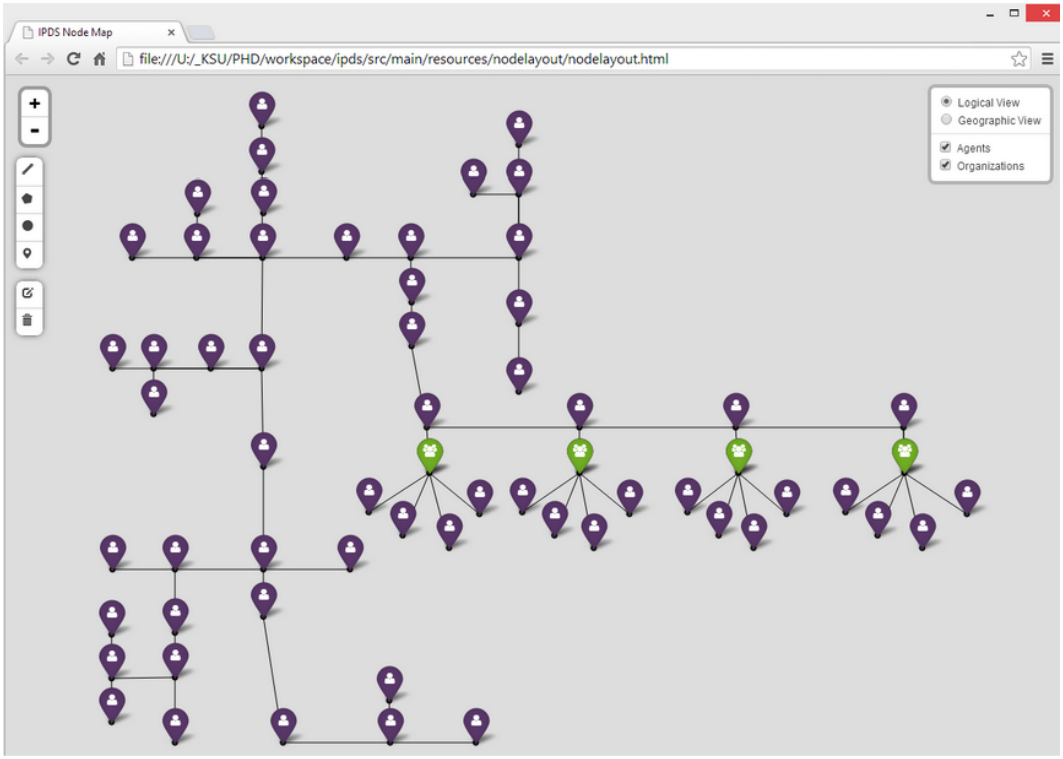


Figure A.4: *62-node test case logical topology (4 neighborhood transformers, 16 homes).*

power. If the master becomes disconnected or disabled, an existing agent in the organization is elected to take its place.

The topology is shown logically in Figure A.4 and with a sample geo-spatial representation in Figure A.5.

A.2 Behavior Specification and Models

The desired behavior for grid voltage control organizations and the agents capable of creating and managing holonic organizations for grid control was defined in models according to the AO-MaSE process.

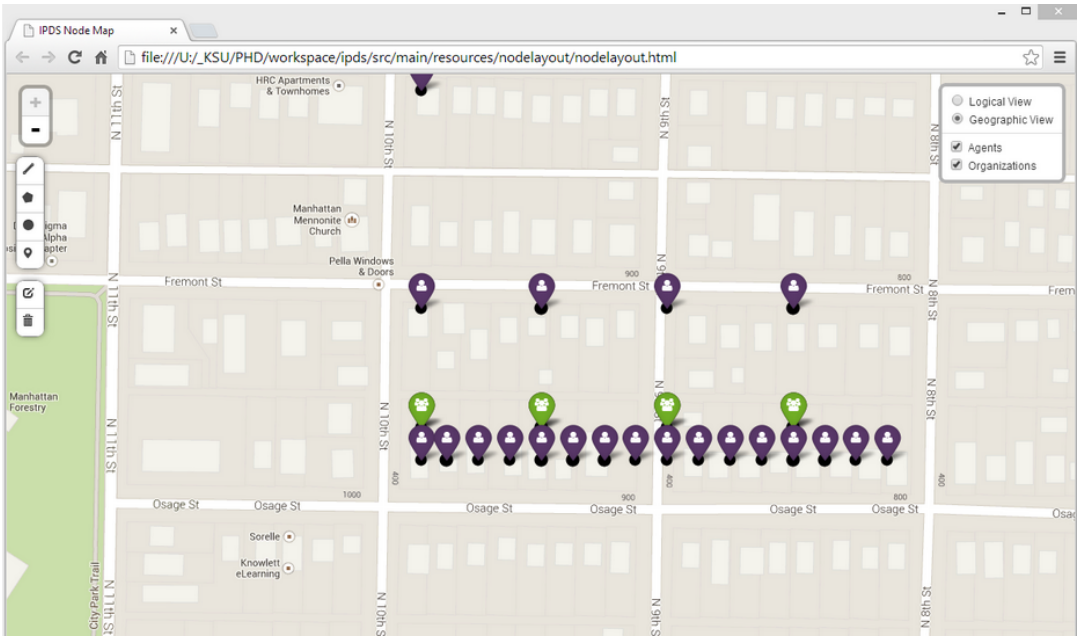


Figure A.5: *62-node test case sample geospatial topology (4 neighborhood transformers, 16 homes).*

A.2.1 Specifications for Grid Control Organizations

The refined goal models for grid control organizations begins with a single substation at the top, branching into a possibly nested series of three-phase feeder lines (feeders may branch into other feeders), down through single-phase lateral lines, into neighborhood transformers, and into individual homes, some of which may be equipped with rooftop solar PV panels and smart inverters. The associated goal models are shown in Fig A.6. The goals and goal parameters are applied recursively between levels - the models are easily applied to any tree-based network configuration, the only requirement being that all participants have exactly one parent, except the top-most substation. In our case, the only level that could have its own level as a parent node was the feeder level, but they could be adjusted as needed if for example, a single-phase lateral line agent had another single-phase lateral line agent as its parent.

The holonic role models are highly similar between roles, reusing a large fraction of the content between levels, but device-specific goals and roles are required to manage the

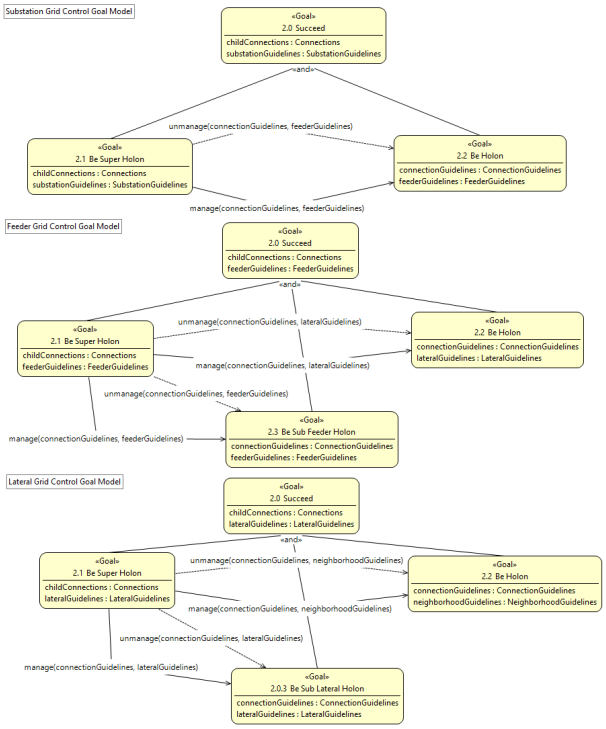


Figure A.6: *Grid control organization goal models.*

different types of sensors and actuators available at different levels of the power distribution system. Therefore, role models for grid control are customized by the types of levels. The goals and goal parameters are applied recursively between levels; the models are easily extensible from homes to neighborhood transformers, to radial or branching single-phase lateral lines, to feeders, to substations – or to a series of aggregated feeders (feeders under feeders) to substations.

A.3 Grid Control Agents

The types of agents in each organization are based on the level where they appear. Most agents have sensors to measure the voltage and power quality at their locations, while some are co-located with actuators that provide control actions to the PDS. At the substation level, the agent types include a single Substation agent and a set of Feeder agents, which

represent their sub-organizations from the feeder level. At the feeder level, the agent types include a single Feeder agent (who also represents its organization at the substation level) and a set of Lateral agents. Each Lateral agent represents a lateral organization, which also includes a set of Transformer agents who are representing their neighborhood organizations. At the neighborhood level, the Transformer agents are joined by a set of Home agents that reside on individual homes, which may or may not have rooftop photovoltaic (PV) solar panels. Additional supporting agents may also be resident in each organization. For example, in the neighborhood level, Forecaster agents may be co-located with Transformer and Home agents to help forecast the local load, temperature, and cloud cover.

A.4 Equipping Agents with Capabilities: Sensors, Actuators, Processing

At the neighborhood level, Home agents equipped with PV generation can be outfitted with *smart inverter* actuators. In alternating current systems, electrical power has two components, *real power, P*, and *reactive power, Q*. P reflects the power available to do useful work and Q can be adjusted to help maintain power quality. Smart inverters can be set to introduce more or less Q to help balance the voltage during sharp swings in PV-enabled generation. Reactive power can be combined up the hierarchy, enabling distributed, cooperative solutions. In the event the smart inverters cannot vary Q enough, smart capacitors on the lateral lines can be actuated to provide additional voltage relief for all downstream agents. When the combination of smart inverters and capacitors is not enough, the Substation agent can alter the setting on its associated load tap changers and provide voltage relief affecting the entire PDS.

A.5 Control Flow

A typical control cycle begins when agents get a new set of sensor data for current generation, consumption, and voltages. If significant changes have occurred since the last set of sensor values, a Home agent equipped with PV-generation may update its smart inverter setting to optimize its performance in relation to its assigned goals. All Home agents then report their data values to their supervising neighborhood Transformer agent, who takes the readings from all the Home agents in its organization, and then calculates and sends a set of smart inverter settings to the PV-enabled Home agents optimized for the neighborhood goals. The Transformer agent, which is the neighborhood's representative in its parent Lateral organization, then reports its current status and margin information to the Lateral level. This process of local aggregation, optimization, and reporting continues up through Lateral and Feeder lines to the Substation. Higher-level organizations have additional actions they can take to support larger changes using equipment such as capacitors and load tap changers.

Under the original approach, once all homes converge, all homes must send a new message all the way up the holarchy to the substation and wait for the message to get back down the holarchy from the substation before they can execute their inverter settings.

The algorithm was enhanced by recognizing once a level has converged, it will always remain converged as shown in Fig A.7.

- First, responses from the smart meter are aggregated up the holarchy to the substation.
- Then, OPF is used to calculate targets for each subholon feeder. They pass these targets all the way down to the homes and new responses come back up.
- Once the targets and responses for all 38 feeders get within a given tolerance, the feeders have converged. *Once converged, always converged*, so the algorithm won't go as high as the substation again during this cycle of iterations; the feeder targets will not continue to change.

- Each feeder continues as the top of new smaller holarchy, continuing iterations under any feeder whose has subholon targets that are still outside the tolerance.
- Subholons that are not converged continue their target is within the tolerance. As the converge in any area of the distribution system reaches all the way down to the home, the home sets the smart inverter actuator reactive power setting to the target value.

In this way, only unconverged areas continue the iteration. Eventually, the convergence spreads out to all distributed areas; first, the 3-phase feeders converge, and then, in a distributed fashion, the convergence spreads out and down until the last neighborhood (the one with 287) reaches convergence.

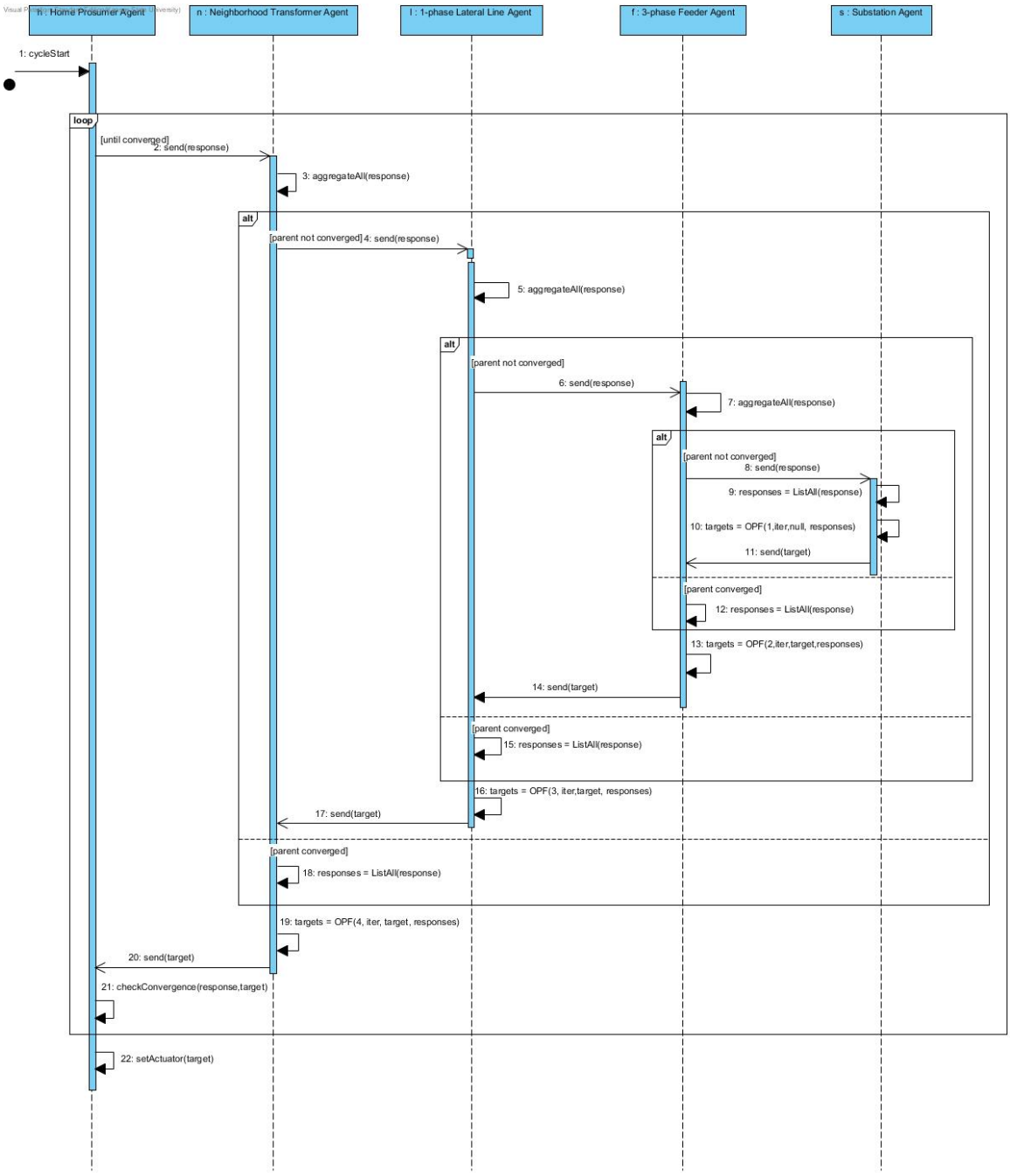


Figure A.7: *Iterative grid control algorithm.*

Appendix B

IPDS Online Auction System (OAS)

We forget just how painfully
dim the world was before electricity.
— *Bill Bryson*[191]

The application employs a two-tier double auction scheme where home prosumer agents create bids to express their intentions and send them to an agent acting as the broker in a local market organization. The agent brokering the local auction determines the optimal resolution of the auction, and in the event of any unsatisfied amounts, participates as a bidder in a secondary, higher-level auction. The approach exploits the applicability of the double auction in the second-tier, where the auction takes place between the secondary participants representing their remaining community bids and shows the efficacy of the proposed hierarchical model as it further maximizes the overall social utility[40].

The project demonstrates an architecture for multigroup agents that provides a modular, extensible approach for supporting agents participating in multiple affiliated and independent groups, each with their own behavior specification, while providing a means to customize the intelligent agents based on homeowner preferences and personal market

strategies.

The remainder of this appendix is organized as follows. The two-tier double auction algorithm is defined in Section B.1. The behavior specification and associated models are described in Section B.2. Auction agents are defined in Section B.3. Agent capabilities are described in Section B.4. A description of the auction process is provided in Section B.5.

B.1 Two-tier Double Auction Requirements

In the two-tier double auction, each *home prosumer agent* participates in a single holon at the lowest level of the holarchy. Each of these lowest level organizations includes a neighborhood transformer agent that may be situated on or near the pole transformer that supplies a small set of homes with power. For testing, each neighborhood transformer agent supported four homes supplied by the associated transformer, one of which has rooftop photovoltaic (PV) panels for generation.

Each neighborhood transformer agent was equipped to broker a local auction, accepting bids from the four participating homes for a given future time period. Homes equipped with rooftop solar panels were assumed to have surplus distributed generation (DG) to sell that nearby homes (those served by the same transformer) could bid on. The neighborhood transformer agent and and the homes supplied by the transformer would autonomously create a small local market organization and execute (or *broker*) the auction.

Each neighborhood transformer agent also further equipped to participate in an auction at a higher level. In these secondary auctions, the neighborhood transformer agents served in a different role. In the higher organization, each neighborhood agent served as an auction participant, while the single lateral power line agent, supplying electricity to several neighborhoods, was equipped to accept their bids and serve as broker in the second-tier double auction.

The holonic nature of these local market organizations is illustrated in Figure B.1. Home

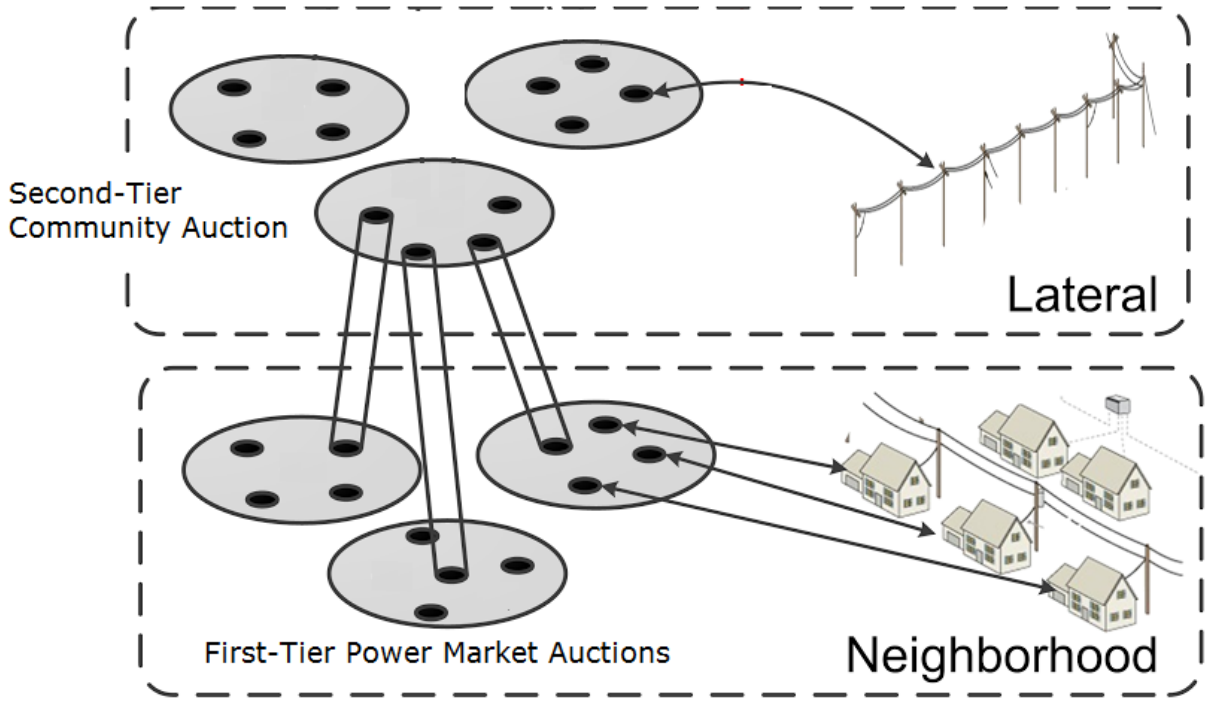


Figure B.1: *Holonic market organizational structure for the two-tier, distributed double-auction simulation.*

prosumer agents bid in first-tier auctions brokered by agents running on neighborhood transformers. Neighborhood transformer agents then bid in second-tier auctions brokered by an agent running on their supplying lateral power line.

B.1.1 First-Tier Auction

At the first tier of the proposed scheme, each of the neighborhood transformer agents (indexed $k \in 1, 2, ...N$) conducts an independent auction from the bids provided by the home prosumer agents supplied by the associated transformer. In each local first-tier auction, let N_B^k and N_S^k be the number of potential buyers and sellers with indexes i and j, respectively, their bid prices per unit of energy be $c_{b,i}$ and $c_{s,j}$, and their maximum demands and available supplies (in energy units) be d_i and s_j. With denoting c_0^k the clearing price per unit of

power, the agent utilities can be defined as follows. For buyers:

$$u_{b,i} = \left\{ \begin{array}{ll} (c_0^k - c_{b,i})q_{b,i}, & c_{b,i} \geq c_0^k \\ 0, & \text{otherwise} \end{array} \right\}. \tag{B.1}$$

and for sellers:

$$u_{s,j} = \left\{ \begin{array}{ll} (c_0^k - c_{s,j})q_{s,j}, & c_{s,j} \geq c_0^k \\ 0, & \text{otherwise} \end{array} \right\}. \tag{B.2}$$

Here, the volumes of energy $q_{b,i}$ and $q_{s,j}$ bought and sold are determined through the auction by maximizing the total utility of all participating agents, U^k. With p^k being the assigned energy volume imported (exported when positive) to neighborhood k, the underlying auction is formulated as the following linear programming problem. Maximize:

$$U^k = \sum_{i \in W_B^k} u_{b,i} + \sum_{j \in W_S^k} u_{s,j} \ . \tag{B.3}$$

Subject to:

$$0 \leq q_{b,i} \leq d_i. \ . \tag{B.4}$$

$$0 \leq q_{s,j} \leq s_j. \ . \tag{B.5}$$

$$\sum_{j \in W_S^k} q_{s,j} - \sum_{i \in W_B^k} q_{b,j} = b^k \ . \tag{B.6}$$

The neighborhood transformer agent, serving as the broker, places the quantities b^k and c_0^k as the bid volume and price, respectively.

B.1.2 Second-Tier Auction

This secondary auction requires the power requested from each neighborhood transformer agent k, to serve as the neighborhood bid volume b^k and clearing price c_0^k. The lateral feeder line agent serves as the broker in the second-tier auction and determines the final clearing

price c_0 at which subsequent power trading occurs and the power flow from each power-exporting neighborhood l to every power-importing neighborhood k. There are various ways in which the clearing price may be determined (e.g., through negotiations with the utility company, to obtain budget balance, or by other means). These issues are not addressed here, and a price c_0 is determined somewhat arbitrarily, to lie within the range of prices in the neighborhood bids. The clearing price determines the *winner sets*—the set of neighbors that ultimately participate in the auction—either as buyers or sellers as defined below.

$$W_l = \{k |\, b^k \leq 0, c_0^k \geq c_0\} \; . \tag{B.7}$$

$$W_E = \{k |\, b^k \leq 0, c_0^k \geq c_0\} \; . \tag{B.8}$$

The objective of the auction is to maximize the social welfare function (SWF), the aggregated utility of all winners, as provided in the following equation.

$$SWF = \sum_{k=1}^{N} U^k \; . \tag{B.9}$$

The neighborhood utilities as seen by the broker in this tier are now determined as follows.

$$U^k = \left\{ \begin{array}{ll} (c_0^k - c_0)p^k, & k \in W_l \\ (c_0 - c_0^k)p^k, & k \in W_E \\ 0, & \text{otherwise} \end{array} \right\} . \tag{B.10}$$

This allows the SWF to be expressed directly in terms of the bids in the following linear programming formulation to obtain the power flows $P^{k,l}$. Maximize:

$$SWF = \sum_{k \in W_l} \sum_{i \in W_E} (c_0^k - c_0^l) p^{k,i} \; . \tag{B.11}$$

Subject to:

$$p^k = \sum_{l=1}^{N} p^{k,l} \ . \tag{B.12}$$

$$\left\{ \begin{array}{ll} 0 \leq p^k \leq b^k, & k \in W_l \\ p^k \leq b^k \leq 0, & k \in W_E \end{array} \right\} . \tag{B.13}$$

$$\sum_{k \in W_l} p^k + \sum_{l \in W_E} p^l = 0. \quad \text{(power balance)} \ . \tag{B.14}$$

The power balance constraint above assumes an isolated microgrid that does not transfer power from external sources. A single clearing price was assumed in the experiments, and the approach is strongly budget balanced. However, the above problem can be reformulated in various ways, in which case a strong budget balance requirement may be added as another constraint.

B.2 Behavior Specification and Models

The desired behavior for online market organizations and the agents capable of creating and managing these organizations was defined in models according to the AO-MaSE process.

B.2.1 Specifications for Market Organizations

The refined goal models for the two-tier double auction includes home agents in the lowest level and transformer agents running on the poles in the first-tier auctions, and single-phase lateral line agents working with several neighborhood transformer agents to conduct the second-tier auctions. The associated goal models are shown in Fig B.2. The goals and goal parameters are applied recursively between levels; the model could be easily extended to conduct additional, higher-level auctions as circumstances allow. The differences in the goal models are unique to the highest and lowest tiers. The lowest level will not have any (lower) auction connections or need guidelines to broker an auction. The highest level will not have

any (higher) broker connections or need auction guidelines to participate in a higher-level auction.

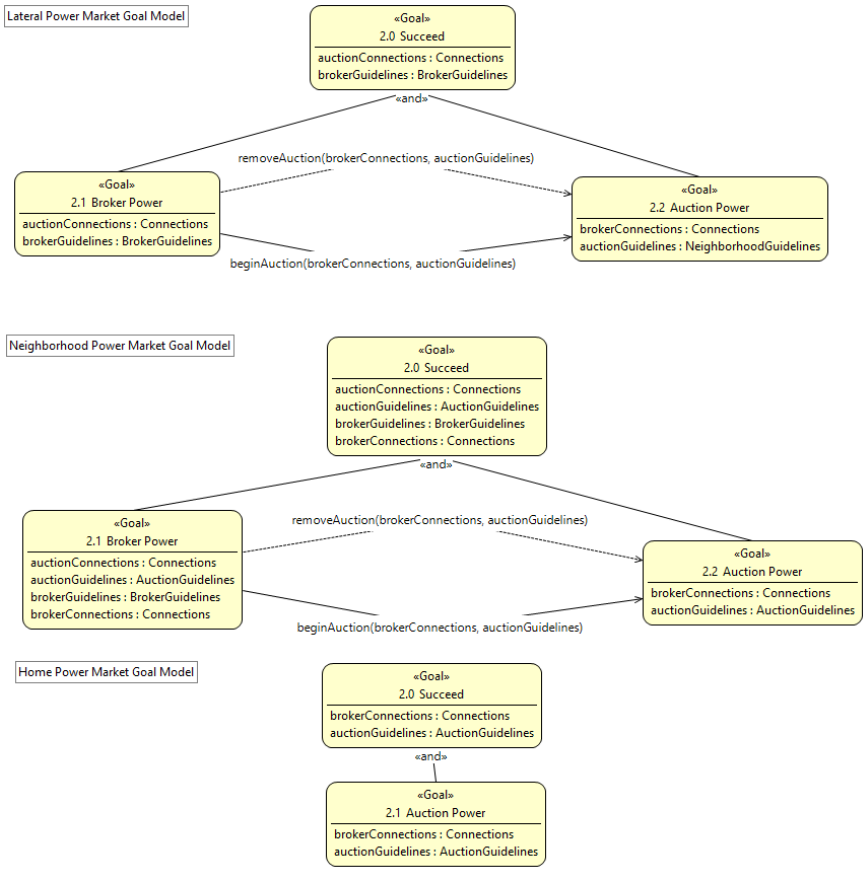


Figure B.2: *Two-tier market organization goal models.*

A single market role model can be used for all markets. The role model is shown in Fig B.3. The goals and goal parameters are applied recursively between levels; the model could be easily extended to conduct additional, higher-level auctions as circumstances allow.

B.3 Online Auction Agents

In addition to the computational approach for the auctions, the ability to extend an existing intelligent power distribution system to support future online auctions was evaluated.

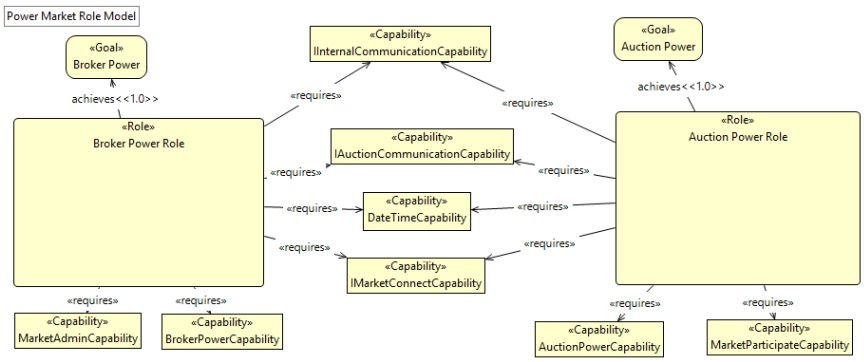


Figure B.3: *Two-tier market organization role model.*

For example, future power distribution systems may include distributed intelligent agents supporting advanced capabilities such as reactive and proactive power quality control for voltage regulation and control[192]. Mechanisms were explored for enhancing intelligent agents by adding capabilities to autonomously create and conduct online auctions. This required agents to operate under the external guidance of multiple affiliated organizations and adapt their behavior to provide the additional functionality without compromising or impacting prior agent behaviors.

B.3.1 Equipping Agents to Conduct On-line Auctions

To implement the on-line double auctions, the existing hierarchic holonic MAS (HHMAS) was used to evaluate power quality control algorithms for future intelligent power distribution systems. The topology, shown in Figure B.4 is based on the IEEE 37-bus feeder test case, with a sample data for a community of four neighborhoods, with four homes each with one of the four having distributed generation that could be made available for sale. The market organizations were arranged in a holonic manner, similar to the grid control options, but are be subject to different behavior specifications and external stakeholders. A smaller, but highly parallel second hierarchical holarchy was implemented to support holonic on-line auction experiments. Agents were built using AASIS and the OBAA^{++}[36] architecture specifically designed for *multigroup agents* participating in multiple independently-controlled

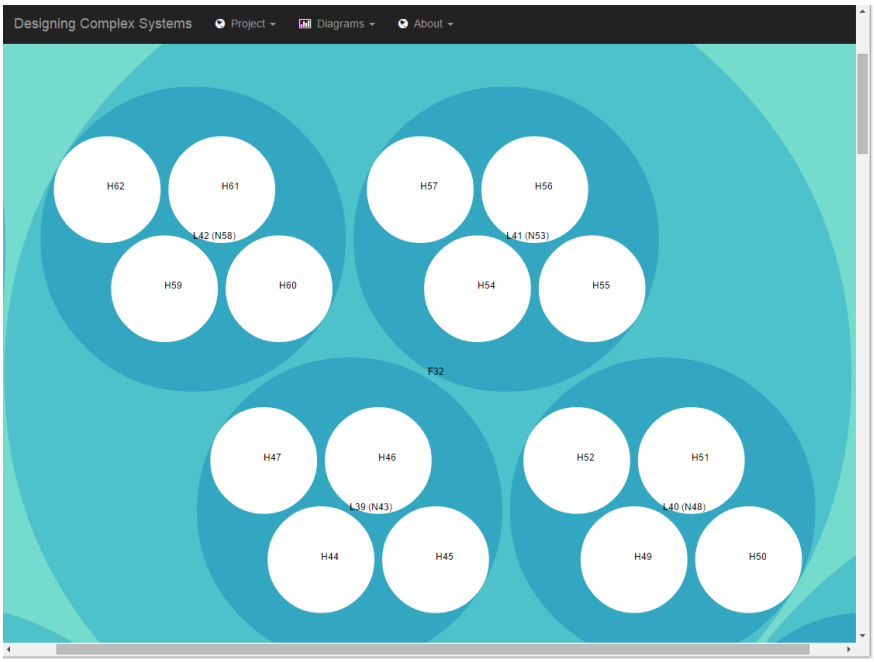


Figure B.4: *Power distribution network topology for the double-auction test case.*

organizations.

OBAA^{++} agents are equipped with *capabilities* that provide specific functionality. The architecture includes an executable goal model for specifying organizational behaviors and defining the behavior goals for each of the local market organizations. During execution of the system, suitably-equipped agents are dynamically assigned to specific roles that can achieve a particular organizational goal. Agents in market organizations can be assigned to only one of two roles. They either act as an *auction participant*, to achieve the goal we called *Auction*, or they act as the *auction broker*, accepting bid messages and executing the double auction for the participants to achieve the goal we called *Broker*. Agents are never assigned to do both in the same local organization, but some mid-level agents may *broker auctions* in a lower-level organization, and then *bid in auctions* in a higher-level one, as Neighborhood transformer agents do.

The necessary capabilities include typical group formation and administration abilities such as the ability to create authorized connections to affiliated agents (for example, an

auction participant must be able to establish a secure line of communication with the local market broker) and to register with the organization, essentially presenting the participants capabilities to the broker so it can get assigned roles to achieve the goals defined for the local market organization. Additional online auction related capabilities focus on the ability to prepare bids, send bid messages to the broker, or call the necessary analytical capabilities to execute or broker the auction and determine the degree to which each bid is satisfied.

A list of the capabilities required for each role is shown in Fig B.5 along with the goal that role can achieve to meet the overall objectives of the organization.

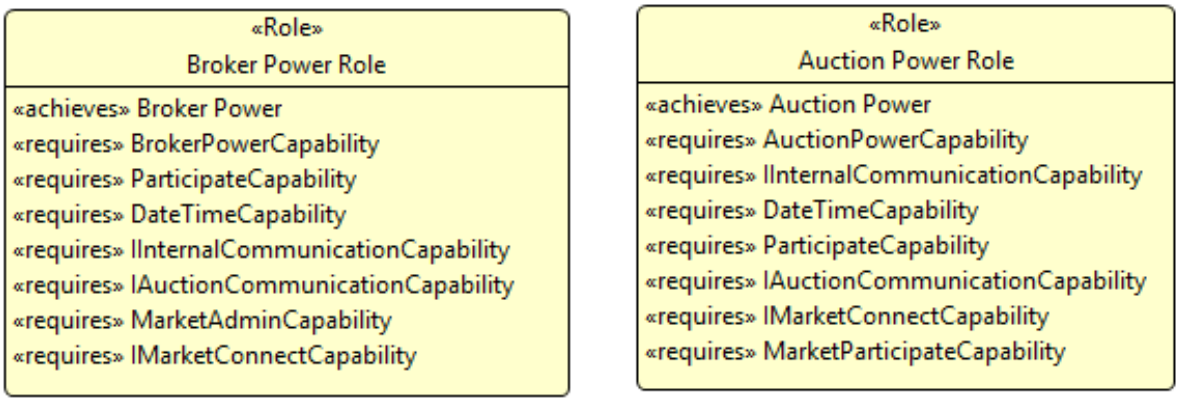


Figure B.5: *Agents operating in the online auction organizations may be assigned to either Auction or to Broker.*

B.4 Equipping Agents with Capabilities: Processing

At the neighborhood level, Home agents equipped with PV generation can be sources of distributed generation (DG). These agents can participate in forward online auctions to sell power or energy at some future time. Agents can be customized to reflect the pricing preferences of their owner and when they detect they will have additional DG beyond the immediate needs of their owner, they can make a bid to sell future generation in online auctions.

B.5 Exchanging Market Messages and Brokering Auctions

Each auction is conducted asynchronously in accordance with the specific guidelines provided. Guidelines include those specified for the market organizations in which the online auctions will be conducted, as well as custom guidelines given to each multigroup agent that serve to direct the behavior of each agent in such a way that the agent could be customized to reflect the personal pricing strategies and comfort/profit motives of the owner. We expect some agents may be ultimately controlled by the homeowner, who makes the decision to sell or not - and some agents may be wholly owned by the power company or market agency, for example, those running along the lateral lines. Communication between agents was simulated using RabbitMQ[1], a fast implementation of the Advanced Message Queuing Protocol (AMQP) standard.[169].

[1] http://www.rabbitmq.com/

Appendix C

Graduate School Research Lab (GSRL)

After great pain, a formal feeling comes.

— *Emily Dickinson*[1]

Additional tests to evaluate the flexibility and reusability were desired, along with development and evaluation of architectural aspects and processing algorithms for managing goal consistency among multigroup agents. Thus, a new application domain was used to create a new set of test cases.

This application domain centers on a university research lab. It includes a professor with goals to run a research lab and advise students, and a set of graduate students who get goals from multiple sources, including assisting in the lab, but also from family, friends, and of course, also maintain personal goals for learning and maintaining basic health and quality of life as shown earlier in Figure 8.8.

[1] As quoted in *How to Write a Dissertation* or *Bedtime Reading for People Who Do Not Have Time To Sleep* [193].

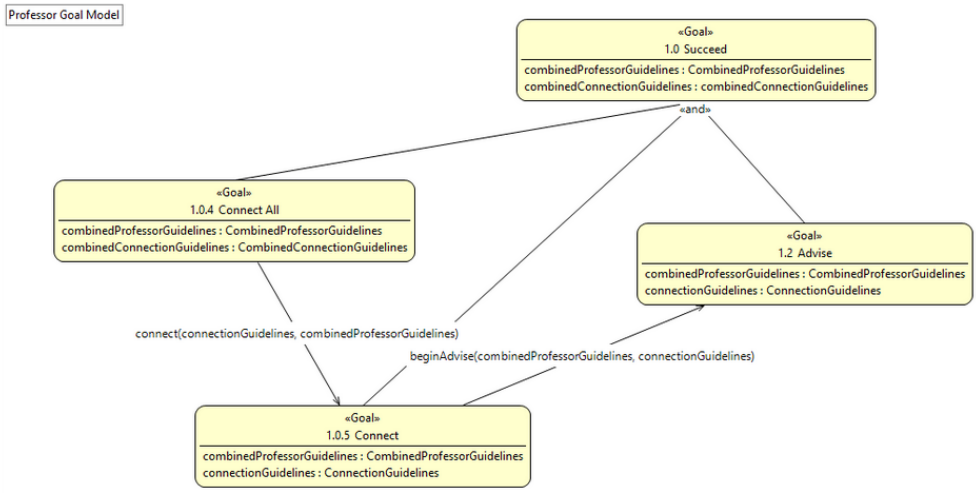


Figure C.1: *Professor agent goal model.*

Goal models, role models and the other specifications for the test cases provided verification that the architecture supports the creation of affiliated organizations between affiliated agents.

The goal specification for a research professor agent is shown in Figure C.1 and the goal-based specification for a dynamically created affiliated organization for a graduate research lab is shown in Figure C.2.

A similar goal specification for a graduate student is shown in Figure C.3

While the test cases were created to provide a simple way to motivate the desired architectural features, the work also be applied to the development of a holonic MAS for

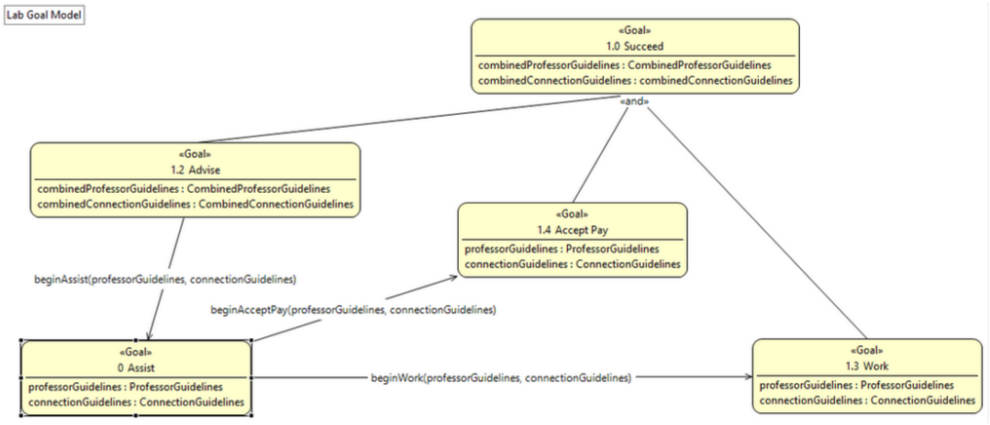


Figure C.2: *Research lab goal model.*

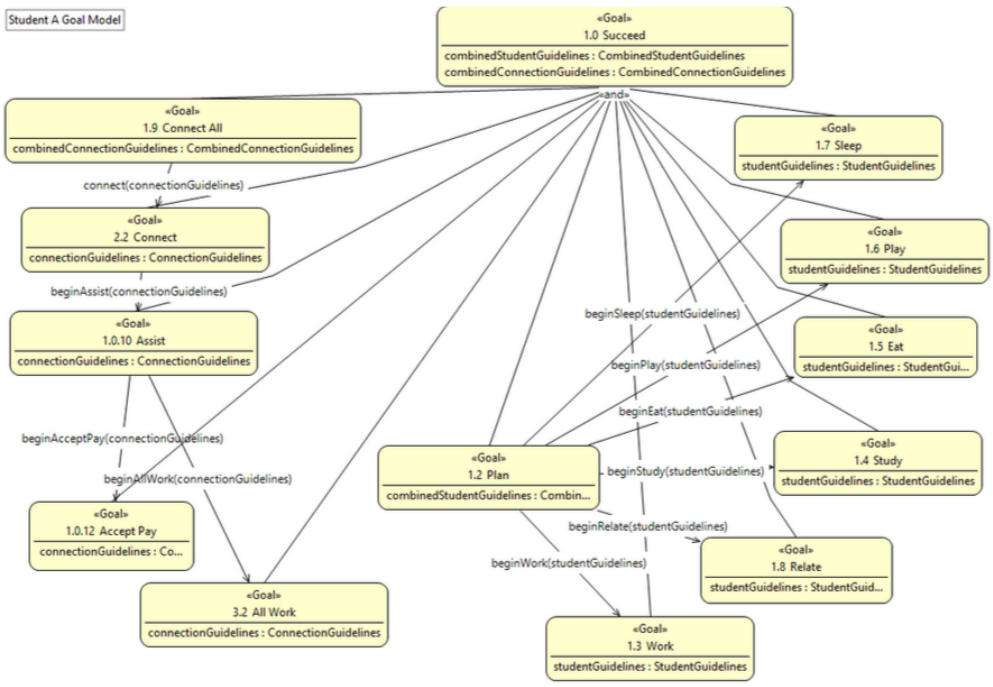


Figure C.3: *Graduate student goal model.*

PDS.

The Adaptive O-MaSE (AO-MaSE) process provided a way to build the systems and implement the necessary features[38]. Additional test-driven development support is planned for additional work as the creation of agent software relies heavily upon design-time configuration and robust error handling and feedback is critically important for the implementation of agent systems, even with the support provided by agentTool3 and the supporting O-MaSE framework. For additional discussion, see Section 8.2.

Appendix D

Acronyms and Glossary

D.1 Acronyms

AASIS	Adaptive Architecture for Systems of Intelligent Systems
AI	Artificial Intelligence
AM	Assignment Manager
API	Application Programming Interface
AO-MaSE	Adaptive Organization-based Multiagent Systems Engineering
APE	agentTool Process Editor
BDI	Belief-Desire-Intention
CC	Control Component
CCEA	Control Component Execution Algorithm
CPS	Cyber-physical systems
DAI	Distributed Artificial Intelligence
DG	Distributed Generation
EC	Execution Component
ECEA	Execution Component Execution Algorithm

EM	Event Manager
GCS	Grid Control System
GMoDS	Goal Model for Dynamic Systems
HMAS	Holonic Multiagent System
HHMAS	Hierarchic Holonic Multiagent System
IEEE	Institute of Electrical and Electronics Engineers
IPDS	Intelligent Power Distribution System
GR	Goal Reasoning
MAS	Multiagent System
MDP	Markov Decision Process
NSF	National Science Foundation
O-MaSE	Organization-based Multiagent Systems Engineering
OAS	Online Auction System
OBAA	Organization-based Agent Architecture for single-organization MAS
$OBAA^{++}$	Organization-based Agent Architecture for multigroup MAS
OM	Organization Model
OMACS	Organizational Model for Adaptive, Computational Systems
OMAS	Organization-based Multiagent Systems
OPF	Optimal Power Flow
OS	Organization Specification
P	Real power
PDS	Power Distribution System
PSA	Plan Selection Algorithm
PV	Photovoltaic
Q	Reactive power

RA	Reorganization Algorithm
SWF	Social Welfare Function
TM	Task Manager
UML	Unified Modeling Language

D.2 Glossary

A glossary of some of the key terms and concepts follows. Descriptions that best reflect the essence of the idea for those unfamiliar has been included below along with the providing source.

Adaptive Organization-based Multiagent Systems Engineering (AO-MaSE). A complete-lifecycle, O-MaSE-compliant methodology for analyzing, designing, and developing complex, multigroup multiagent systems[150].

Agent. Computational system instances that inhabit a complex dynamic environment, sense and act autonomously in this environment in order to achieve a set of goals[150].

AgentTool. A Java-based graphical development environment to help users analyze, design, and implement multiagent systems developed by the Multiagent and Cooperative Robotics (MACR) Laboratory at Kansas State University[194].

Architecture. That which is fundamental or unifying about a system as a whole; the set of essential properties of a system which determine its form, function, value, cost, and risk[195].

Capabilities. Capabilities are atomic entities used to define a skill or capability of

agents[13]. Capabilities can capture soft abilities such as the ability to access resources, communicate, migrate, or computational algorithms. They also capture hard capabilities such as those of hardware agents such as robots, which include sensors and effectors[150].

Complex MAS. An multiagent system with a complex organizational structure containing multiple groups. See also multigroup MAS.

Computer science. Computer science is the scientific and practical approach to computation and its applications[196].

Computer scientist. A computer scientist specializes in the theory of computation and the design of computational systems[196].

Critical Peak Pricing. Rate and/or price structure designed to encourage reduced consumption during periods of high wholesale market prices or system contingencies by imposing a pre-specified high rate or price for a limited number of days or hours[197].

Critical Peak Pricing with Load Control. Demand-side management that combines direct load control with a pre-specified high price for use during designated critical peak periods, triggered by system contingencies or high wholesale market prices[197].

Cyber-physical systems (CPS). Engineered systems that are built from and depend upon the synergy of computational and physical components. Emerging CPS will be coordinated, distributed, and connected, and must be robust and responsive. Examples include the smart electric grid, smart transportation, smart buildings, smart medical technologies, next-generation air traffic management, and advanced manufacturing[198].

Demand Resource or Demand-Side Resource. An electricity consumer that can decrease its power consumption in response to a price signal or direction from a system operator[197].

Direct Load Control. A demand response activity by which the program sponsor remotely shuts down or cycles a customer's electrical equipment (e.g., air conditioner, water heater) on short notice. Direct load control programs are primarily offered to residential or small commercial customers. Also known as direct control load management[197].

Distributed Energy Resources (DER). A changing mix of demand-side resources, including changeable load, dispatchable distributed generation and storage, as well as variable output local generation such as wind and solar. In the event of a disturbance, attack, or natural disaster, these resources can help alleviate constraints or support electrically energized islands that can mitigate the impact to events, and improve response times for post-disturbance reconstruction[199].

Distributed Generation or Distributed Generators (DG). Distributed Generation is a broad term that encompasses both mature and emerging onsite power generation technologies with power output as small as 1 kW and as large as 20 MW[200].

Environment. The external world in which a system or an entity operates. Agents can perceive their environment through sensors and can act on the environment via actuators. Agents may be part of the environment for other agents.

Goal. A desirable state of the world or the objective of a computational process[13].

Goal Model for Dynamic Systems (GMoDS). A software tool that provides a formal definition and decomposition of system goals and the relationships between them and offers a

framework for executing a goal model within an organization. See also Specification Goals, Instance Goals, and Goal Parameters.

Goal Parameters. Guidelines provided to customize a parameterized goal.

Holon. An agent or unit that is at the same time a whole – composed of smaller parts – and also a part of higher level organization. An atomic holon is one considered to be at the lowest level of a particular system - and no additional division is considered. Alternatively, a non-atomic holon plays both a role in a higher level organization and can, itself, be considered as an organization of holonic agents[201].

Holarchy. An organizational approach based on holons.

Holonic Multiagent System. A special kind of multiagent system where an agent may consist of multiple, similar agents acting together, where each agent may either one of the parts, or act as the head (the agent that represents the holon to the greater system).

IEEE Standard 1547. Current IEEE Standard 1547 requires all distributed generators to disconnect from the grid upon loss of power. New standards would allow a PDS to operate as an islanded microgrid with its own resources[202].

Instance Goals. Temporal versions of specification goals created during system execution.

Intelligent Power Distribution Systems project (IPDS). A four-year 1.1 million project focusing on developing an architecture to support the evolving power distribution system[156].

IntelliGrid. EPRI's IntelliGrid initiative is a collaborative effort to create a technical

foundation for a smart power grid that links electricity with communications and computer control to achieve gains in reliability, capacity, and customer services. A major early product is the IntelliGrid Architecture, an open-standards, requirements-based approach for integrating data networks and equipment that enables interoperability between products and systems. This program provides utilities with the methodology, tools and recommendations for standards and technologies when implementing systems such as advanced metering, distribution automation, demand response, and wide-area measurement. The program also provides utilities with independent, unbiased testing of technologies and vendor products[200].

Markov Decision Process (MDP). A framework for modeling decision-making when outcomes are partly random and partly under control of a decision maker. MDPs are used in stochastic processes where the probability of future states depend only on present state and nothing preceding it[203,204,205].

Micogrid. A small energy system capable of balancing captive supply and demand resources to maintain stable service within a defined boundary[206].

Multiagent System. A system consisting of multiple autonomous entities having different information and/or diverging interests[175].

Multigroup Agent. An agent designed to accept or issue assignments in multiple groups and/or multiple systems concurrently.

Multigroup MAS. An multiagent system with a complex organizational structure containing multiple groups. See also complex MAS.

Organization-Based Agent (OBA): An agent capable of reasoning about its organiza-

tion[13], with the ability to *reorganize*, or transition from one organizational state to another, in response to updated goals or changes in the environment.

Organization-based Multiagent Systems Engineering (O-MaSE). A complete-lifecycle methodology for analyzing, designing, and developing heterogeneous multiagent systems[150].

Organizational Model for Adaptive, Computational Systems (OMACS). A model that defines a system in terms of an organization consisting of goals, roles, agents, capabilities, and the relationships between these entities[13].

National Science Foundation (NSF). The United States NSF provides research funding for many advanced research efforts, including the Kansas State Intelligent Power Distribution System Cyber-Physical Systems Project[198].

P (power). Real power, also called active power. The part of the power flow that can be used to perform desired functions. Complex power is the vector sum of real and reactive power. The apparent power is the magnitude of the complex power. Aspects of power are related as shown in the following figure with Real power (P), Reactive power (Q), Complex power (S), Apparent Power (—S—), and Phase of Current (ϕ) as indicated.

Photovoltaic (PV). PV devices absorb sunlight and convert the light energy into electricity that you can be used to supply energy for homes or industrial applications. PV panels allow a home to act as a prosumer depending on the availability of sunlight.

Policies. Organization policies are formally specified rules that describe how an organization may/may not behave in specific situations[150].

Power Distribution System (PDS). The parts of the grid that operate below transmissions levels, generally below 34.5kV, including all utilization voltage equipment plus all lines that feed power to service transformers; and all radial equipment[207].

Power Quality. Power quality refers to the attributes of the power delivered to customers, including voltage, wave form, and harmonics. A power quality problem can be defined as voltage, current, or frequency deviations that result in failure or misoperation of equipment[207].

Prosumer. An entity that can be both a producer and consumer of electricity.

Q (power). See Reactive power.

Software architecture. The software architecture of a program or cyber-system refers to the structure and organization of the system, including its components, the externally visible properties of those components, and the relationships among them[208,209].

Software engineering. The *science and art of designing and making with economy and elegance, [...] systems so that they can readily adapt to the situations to which they may be subjected*[210].

Reactive power (Q). As reactive power increases, the ability to carry real power (R) is reduced and the corresponding efficiency decreases. Uncorrected reactive power makes it more difficult to stabilize grid voltage. See real power for additional information.

Renewable Energy. Energy which comes from natural resources such as sunlight, wind, rain, tides, and geothermal heat, which are naturally replenished. The smart grid will be able to make better use of these energy resources by giving grid operators tools to reduce

power demand quickly when renewable sources such as wind or solar power dips, and it will have more energy storage capabilities to absorb excess renewable power when it isn't needed, then to release that energy when the renewable power declines. In effect, energy storage will help to smooth out the variability in intermittent renewable resources, making them easier to use[211].

Smart Grid (SG). A developing network of transmission lines, equipment, controls and new technologies working together to respond to our evolving demands for electricity[211].

Smart Grid Objectives. Objectives for the smart grid include more efficient transmission of electricity, quicker restoration of electricity after power disturbances, reduced operations and management costs for utilities, lower power costs for consumers, reduced peak demand, increased integration of large-scale renewable energy systems, better integration of customer-owner power generation systems, and improved security[200].

Smart meter. An electrical meter that records consumption of electric energy in intervals of an hour or less and communicates that information at least daily back to the utility for monitoring and billing purposes[212].

Specification Goals. Behavior objectives for the system. Specification goals are instantiated as instance goals during system execution.

Var control. Control of reactive power (VARs). By reducing the amount of reactive power flowing on the distribution feeder, the electric utility can reduce electrical losses and improve the voltage profile[213].

Voltage control. The primary purpose of voltage control is to maintain acceptable volt-

age (120 volts plus or minus 5 percent) at the service entrance of all customers served by the feeder under all possible operating conditions[213].

Volt-Var optimization. Integrated control of both voltage and reactive power combined. Feeder voltage and feeder reactive power flow are closely related and dependent variables. Control actions to change one of the variables can result in opposing control actions to change the other variable. For example, raising the voltage using the substation transformer LTC can produce a voltage rise that could cause capacitor bank controls to remove a capacitor bank from service, thus lowering the voltage. Similarly, placing a capacitor bank in service could cause the LTC to lower the voltage at the substation. The coordinated control of voltage and reactive power is needed to determine and execute volt-VAR control actions that are truly optimal. In addition, adaptive algorithms may be added to allow the system to learn from previous actions and their resulting impacts[213].

Alphabetical Index