Applying an O-MaSE Compliant Process to Develop a Holonic Multiagent System for the Evaluation of Intelligent Power Distribution Systems

Denise Case and Scott DeLoach

Department of Computing and Information Sciences
Kansas State University,
234 Nichols Hall, Manhattan, Kansas, USA.

{dmcase,sdeloach}@ksu.edu

Abstract. This paper describes the application of an Organization-based Multiagent System Engineering (O-MaSE) compliant process to the development of a holonic multiagent system (MAS) for testing control algorithms for an intelligent power distribution system. The paper describes the Adaptive O-MaSE (AO-MaSE) process, which provides architects and developers a structured approach for testing and iteratively adding functionality in complex, adaptive systems. The paper describes the holonic MAS architecture for the intelligent power distribution system, the challenges encountered while developing the holonic architecture, the lessons learned during the project, and demonstrates how the application of the process enhanced project development.

Keywords: Agent-oriented software engineering, holonic multi-agent systems, adaptive systems, smart infrastructure, intelligent power distribution systems

1 Introduction

Multiagent systems (MAS) are getting significant attention for Power Distribution Systems (PDS) and there is a growing interest in the application of holonic multiagent systems (HMAS) to PDS [22]. HMAS are adaptive, communicative, and autonomous – traits they receive from their MAS heritage – and their hierarchical, recursive structure is a natural fit for PDS systems. Holonic comes from the Greek word holon, a union formed from holos meaning whole and on meaning parts [19]. Thus, holons are parts that are also wholes. In an HMAS, the holon indicates an agent *participating* in one organization that also represents an entire organization itself. The overall organization of nested holonic organization agents is called a holarchy [8]. This hierarchical composition mechanism defines a powerful framework for distributing intelligence and finding local solutions in a recursive manner.

The primary goal of the Intelligent Power Distribution System (IPDS) project at Kansas State University is to demonstrate an HMAS architecture capable of adaptively controlling future PDS that are expected to include a large number of renewable power generators, energy storage devices, and advanced metering and control devices. Specifically, we are considering PDS with a high penetration (between 25% and 75%) of home-based PV systems. The HMAS architecture will also be used to support new analytical algorithms aimed at limiting the impact of information delay, quality and flow on the PDS. The purpose of this paper is to present the O-MaSE compliant software engineering process that we used while developing our prototype HMAS architecture for the IPDS project.

2 Background

Power distribution is estimated to account for approximately 40% of the capital investment in power systems worldwide, roughly comparable to the amount invested in generation, and about twice that invested in transmission assets [22]. PDS automation has lagged the advances in generation, due in part to the distributed nature and the massive number of components that make up the system. Power distribution begins with the primary circuit leaving a substation. It includes a distribution network of 3-phase feeder lines that branch into single-phase lateral lines and a variety of supporting equipment. Lateral lines distribute power through shared transformers that ultimately feed a set of electricity consumers, such as individual homes. Traditionally, control of the system, like the energy, has flowed from the central power source outward and downward toward the end consumer.

However, the increasing presence of renewable, distributed energy resources creates a bidirectional flow of power in the system. Rather than solely consuming power, home owners are installing increasing numbers of rooftop photovoltaic (PV) systems that generate power from solar radiation. The PV electricity generated during peak hours of the day can be greater than what is consumed by the associated home and creates an opportunity for homeowners to sell excess power into the grid. Distributed energy suppliers introduce a need for enhanced information flows, including online auctioning of power between growing numbers of market participants. At the same time, distributed energy sources are subject to intermittency. Wind fluctuation and passing clouds can introduce rapid variation in the amount of power flowing into the system. Rapid change in generation creates significant challenges for maintaining voltages within desired ranges. However, coordinated volt/var control can help provide consistent voltage to consumers while reducing rapid cycling of equipment, improving efficiency, and reducing the overall cost of power generation.

Centralized control is supported by load tap changers (LTC) near the substation that have a limited number of setting changes per day (more rapid cycling is expensive and shortens the equipment life). Distributed control can be supported by the addition of smart inverters that moderate a PV system’s reactive power to offset generation changes and by line capacitors employed in various places along the grid.

Distributed control equipment coupled with centralized, large impact devices like the LTCs create an excellent opportunity for distributed intelligent systems that can adapt reactively and proactively to offer substantial benefits. In addition, distributed intelligent systems can help the increasingly connected remote nodes that are both

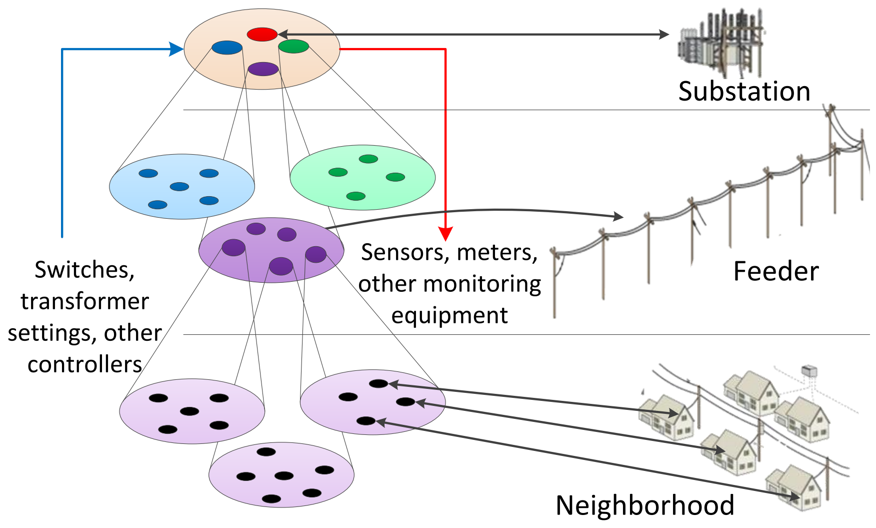

Substation
Feeder
Sensors, meters, other monitoring equipment
Switches, transformer settings, other controllers
Neighborhood

Fig. 1. Application of Holonic MAS in a Power Distribution System

electricity producers and consumers (*prosumers*) work together when disconnected from the grid. During this *islanded mode* participants can work together to provide electricity to critical loads, even though the lack of a stable power source (i.e., power from the grid) creates additional challenges for power supply and quality.

The mapping of our prototype HMAS architecture to a physical distribution system is shown in Fig. 1. The left side of Fig. 1 illustrates the HMAS holarchy of adaptive agent organizations and the right side illustrates the physical system. The lowest level of the holarchy shows peer agents representing single homes acting within a local organization at the neighborhood level (assumed to be represented by the pole transformer serving that set of homes). Each neighborhood organization is represented by a corresponding neighborhood organization agent in the higher lateral-level organization. Nested organizations reflect the physical system from the originating substation down to individual consumer homes. The holarchy, like the physical network, recursively aggregates the systems and organizations to provide an integrated model of the system and provides a framework that allows the IPDS be able to learn and support adaptive, distributed control.

3 Related Work

3.1 Smart Infrastructure Optimization with Agents

Smart infrastructure optimization involves some of the most complex and critical systems in modern society [22]. 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 [23]. When modeling power systems, we are especially interested in agent traits such as autonomy, heterogeneity, adaptivity, social ability, communicability, flexibility, and concurrence [26]. Agents implement goal-based behavior and 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.

Current research projects include a variety of studies involving the application of MAS to power systems, with active research projects focusing on power auctioning, negotiating, volt/var control, distributed communications, and other focus areas [3, 13, 18, 20, 23, 24, 25]. Recent research has also applied HMAS to PDS [2, 15, 17], sometimes in concert with specific agent-oriented software engineering methodologies. 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 will allow us to evaluate a variety of control algorithms and strategies.

3.2 Agent-Oriented Software Engineering Methodologies

For the IPDS project, we needed a reactive, proactive, model-driven, proven MAS framework with a supporting formal process methodology. Several agent-oriented engineering methodologies are available for developing complex, adaptive systems and the methodologies include the metamodels and process flows that provide the structure necessary to engineer MAS. Methodologies that have been successfully employed include ADELFE, ASPECS, INGENIAS, O-MaSE, PASSI, Prometheus, SODA, and Tropos [8]. Some methodologies, such as ASPECS and ANEMONA support holonic concepts [1, 7, 11] and O-MaSE enables hierarchical decomposition via organizational agents. We selected O-MaSE and the associated Organization-based Agent Architecture (OBAA) for our foundations because of the unique tools and process flexibility provided. Together with the agentTool3 integrated development environment, they offer a flexible, extensible system to support reactive control as well as predictive capabilities and proactive control.

4 Foundations

4.1 O-MaSE Process Framework

O-MaSE is an organization-based, role-centered process framework that consists of three main components: a metamodel, method fragments, and guidelines [10]. The metamodel describes system components as shown in Fig. 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-

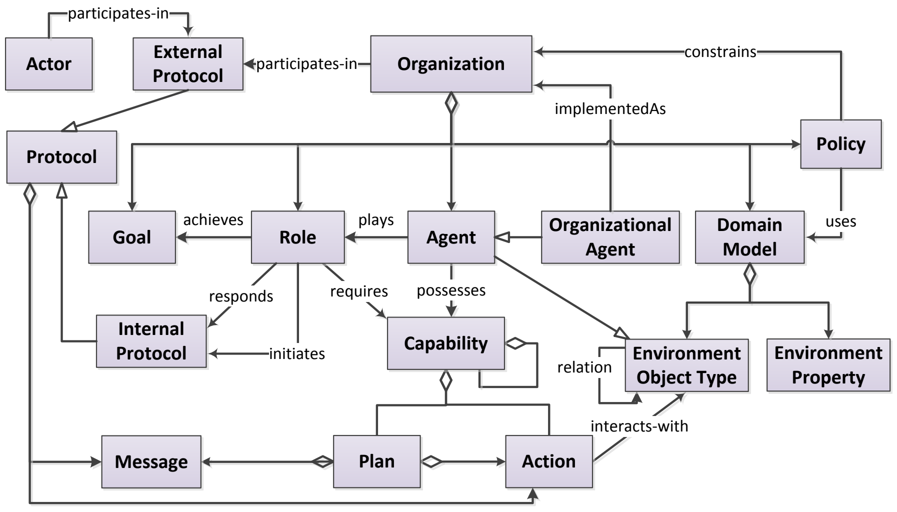


Fig. 2. The O-MaSE Metamodel (multiplicities not shown).

MaSE is supported by agentTool3 modeling tools, which support method creation and maintenance, model creation and verification, and code generation and maintenance.

Linnenberg et al. used the O-MaSE methodology and agentTool3 to develop DEMAPOS (DEcentralized MArket Based POwer Control System) [19] for power trading and we have followed the DEMAPOS convention of combining entities capable of producing and/or consuming electricity into the notion of prosumer agents.

4.2 Holonic Multiagent Architectures

Holonic multiagent architectures introduce additional elements to MAS. Cossentino et al. describe the multilevel interplay between local organizations [7] where an agent participating locally in an organization may represent the entire lower-level organization while also participating as an agent in a higher-level organization. A holonic organization agent may play different roles in different organizations simultaneously as shown in Fig. 4. Participation in multiple organizations is not unique to holonic agents; rather it is the recursive representation of progressively distributed agent organizations that made the holonic approach desirable for evaluating IPDS.

We evaluated a variety of existing systems to provide core MAS functionality and selected the Organization-based Agent Architecture (OBAA) as our foundation. OBAA has been employed in a variety of adaptive systems [21]. OBAA agents include a shared control component enabling basic communication capabilities within the local organization coupled with an embedded knowledge representation of the organization in which the agent operates. In OBAA, each control component coordinates with the team and maintains a complete copy of the local organization

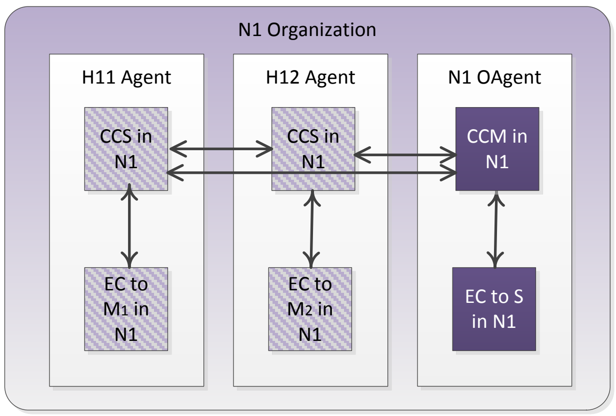


Fig. 3. Each participating OBAA agent includes a control component and a domain-specific execution component to enable participation in the organization.

knowledge. Each OBAA agent also has a domain-specific execution component that manages roles and capabilities as shown in Fig. 3.

The primary objective of the generic OBAA architecture is to offer a general framework of classes from which domain-specific adaptive systems can be built. The OBAA framework is built on the Organization Model for Complex, Adaptive Systems, OMACS [10], and includes an executable goal model, GMODS [9]. The control component parts provide the functionality for agents to get initialized, register with an acting supervisor and, once the registration process is complete, to begin executing their assigned tasks. An initial agent registration process was defined to get the system running; it will be enhanced to incorporate leader election as the project continues.

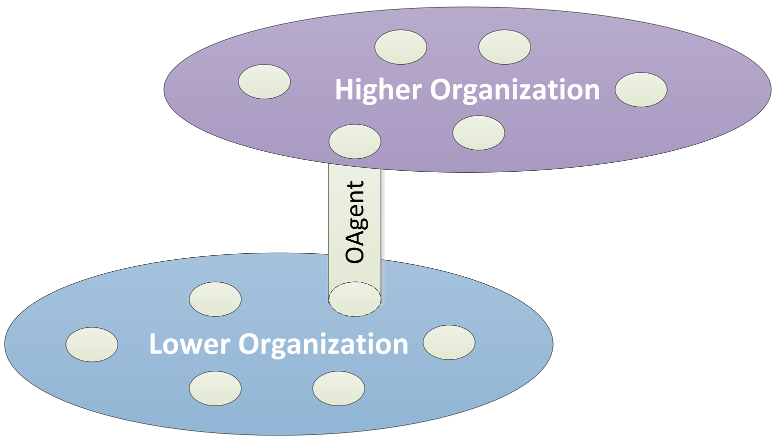


Fig. 4. A holonic organization agent (OAgent) serves as the master agent in its lower-level organization (blue) and a peer agent in a higher-level organization (purple).

5 AO-MaSE: An O-MaSE Compliant Process

O-MaSE 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].

MAS and HMAS are complex models of complex systems. Getting started with such frameworks can be challenging [16]. 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 [4]. 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 [5]. 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 [5]. In addition, research at the University of Vigo in Spain has adapted the INGENIAS methodology to follow the agile process SCRUM with promising results [12].

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 which offers a solid core from which increasingly complex analyses and behavior can evolve. AO-MaSE follows the O-MaSE compliant process includes three iterations that includes the tasks and work products shown in Fig. 5. The process follows four key strategies:

- Start simply and add incrementally; in the OBAA framework, agents given even simple goals 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.

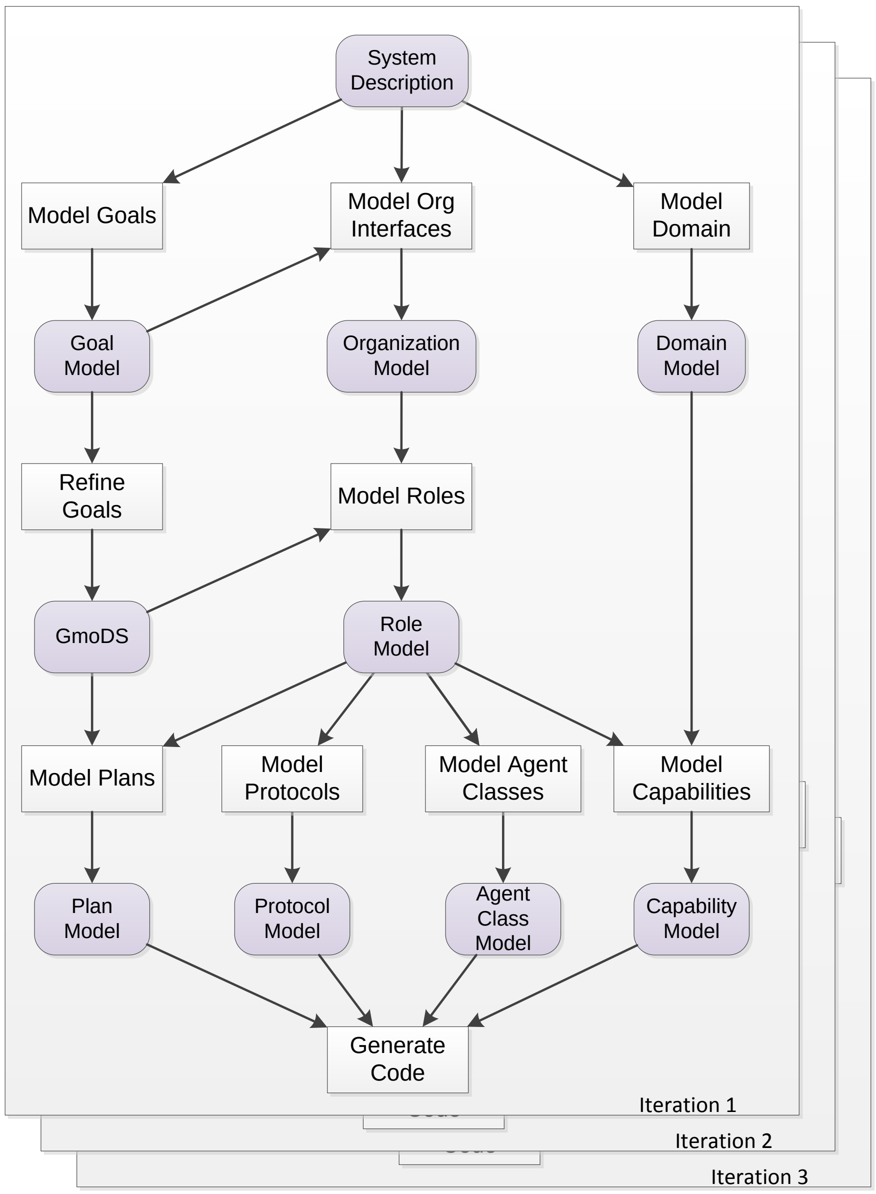


Fig. 5. AO-MaSE Iterative Process (tasks are shown as rectangles, work products are shown as rounded rectangles).

6 Applying AO-MaSE in the Development of IPDS

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.

6.1 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 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.
4. Define plans to perform each role.
5. Define capabilities specific to each plan; define a local domain-specific communication capability and an external controller communication capability.
6. Assign role requirements. Most roles require control communication (for OBAA agents), the local domain-specific communication, and the appropriate role-specific capability. Certain roles will require external controller communication capability.
7. Define a limited number of plan states, e.g. *INIT*, *EXECUTE*, and *STOP*.
8. Define plan state transitions and state behaviors by defining and calling capability methods.
9. Define agent classes based on the problem domain.
10. Configure agent instances with associated capabilities and attributes.
11. Configure environment object instances such as sensors and actuators.
12. Implement code components by extending the OBAA framework.
13. Configure, implement, debug, test, and execute the initial vertical spike.
14. Where possible, follow parallel, explicit naming conventions that differ only in type (as shown in the IPDS models).

The project began with the specification of requirements. From the set of requirements for the initial phase of the project we selected an initial focus that allowed us to test core functionality – the distribution and management of goals within a local organization.

Since achieving each goal requires substantial infrastructure, we started with 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, we created a role 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, we were specifying the methods required of each capability. Parallel naming conventions for goals, roles, plans, and role-specific default capabilities aided clarity and were used to employ additional code automation. Agent classes did not parallel the goal or plan names. Instead, they reflected the physical installation or focus of the agent type. We began with a 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 OBAA framework can be employed immediately if one control component master is declared for any local organization. We 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. Some key models from Iteration 1 are shown in Fig. 6. The models illustrate the parallel naming conventions between goals, roles, and plans, and although only a small subset of the models created are included, helps illustrate the infrastructure support underlying an agent-based adaptive system.

6.2 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. We began working with the new holonic organization agents and the development focused on enhancing the plan states and capability method calls. Additional capability classes were added, providing additional differentiation and room for expanded functionality. Capabilities were implemented with simple algorithms that served to define the expected interfac-

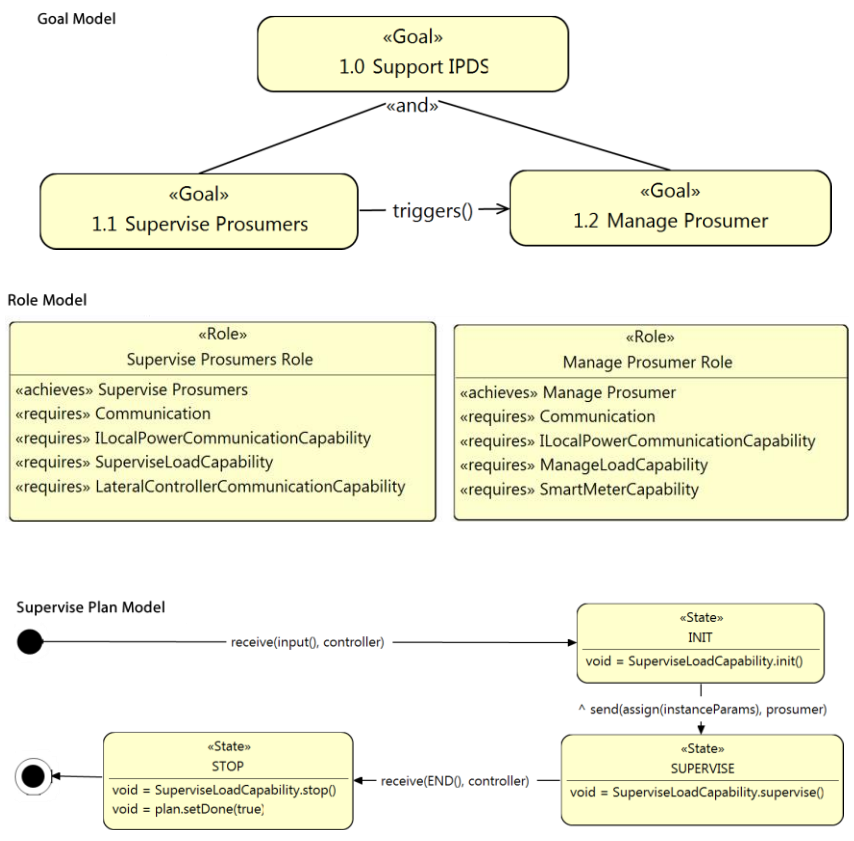

Goal Model
«Goal» 1.0 Support IPDS
«and»
«Goal» 1.1 Supervise Prosumers
triggers()
«Goal» 1.2 Manage Prosumer
Role Model
«Role» Supervise Prosumers Role
«achieves» Supervise Prosumers
«requires» Communication
«requires» ILocalPowerCommunicationCapability
«requires» SuperviseLoadCapability
«requires» LateralControllerCommunicationCapability
«Role» Manage Prosumer Role
«achieves» Manage Prosumer
«requires» Communication
«requires» ILocalPowerCommunicationCapability
«requires» ManageLoadCapability
«requires» SmartMeterCapability
Supervise Plan Model
receive(input(), controller)
«State» INIT
void = SuperviseLoadCapability.init()
^ send(assign(instanceParams), prosumer)
«State» SUPERVISE
void = SuperviseLoadCapability.supervise()
receive(END(), controller)
«State» STOP
void = SuperviseLoadCapability.stop()
void = plan.setDone(true)

Fig. 6. Partial set of first-iteration IPDS models with AO-MaSE conventions.

es that would be required to support more complex optimization algorithms that were being developed in parallel research projects.

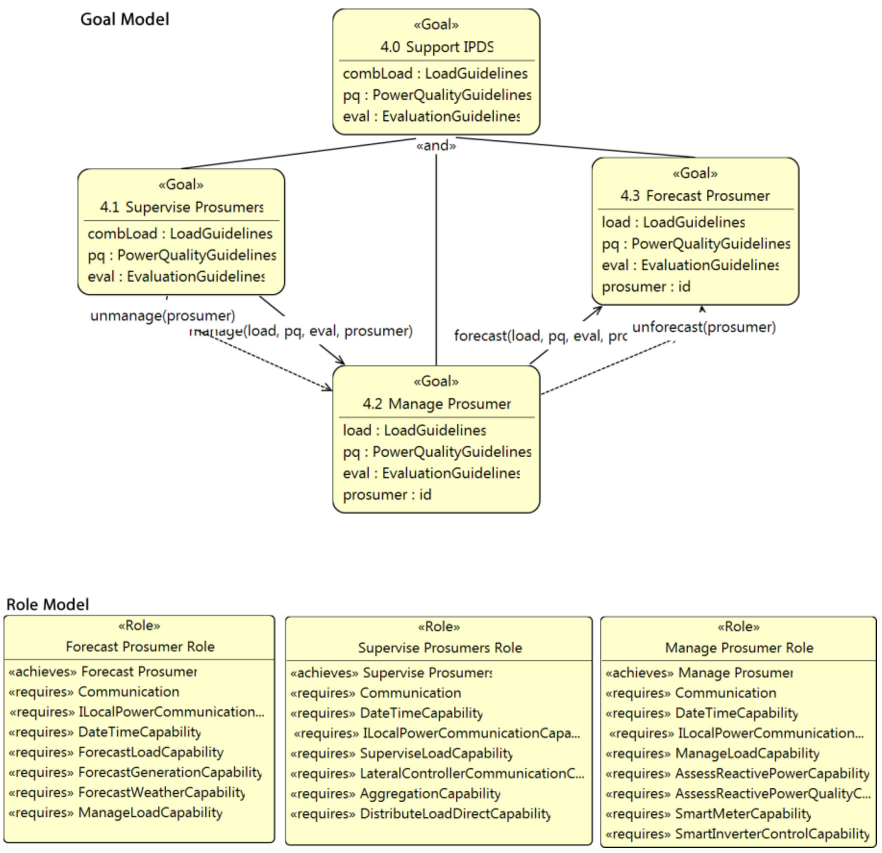
The goal model was enhanced to include parameterized goals with the external controller providing combined guidelines for the organization. Additional triggers were added to the refined goal model. The supervise goal, which had been distributing combined goals among participants during the INIT state, 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.

6.3 Iteration 3 – Extending Functionality

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, we were able to add new features more easily. Additional goals brought additional triggering events and goal parameters. The expanding goal models and role models are shown in Fig. 7.


Goal Model
«Goal»
4.0 Support IPDS
combLoad : LoadGuidelines
pq : PowerQualityGuidelines
eval : EvaluationGuidelines
«and»
«Goal»
4.1 Supervise Prosumers
combLoad : LoadGuidelines
pq : PowerQualityGuidelines
eval : EvaluationGuidelines
«Goal»
4.3 Forecast Prosumer
load : LoadGuidelines
pq : PowerQualityGuidelines
eval : EvaluationGuidelines
prosumer : id
unmanage(prosumer)
manage(load, pq, eval, prosumer)
forecast(load, pq, eval, prc
unforecast(prosumer)
«Goal»
4.2 Manage Prosumer
load : LoadGuidelines
pq : PowerQualityGuidelines
eval : EvaluationGuidelines
prosumer : id
Role Model
«Role»
Forecast Prosumer Role
«achieves» Forecast Prosumer
«requires» Communication
«requires» ILocalPowerCommunication...
«requires» DateTimeCapability
«requires» ForecastLoadCapability
«requires» ForecastGenerationCapability
«requires» ForecastWeatherCapability
«requires» ManageLoadCapability
«Role»
Supervise Prosumers Role
«achieves» Supervise Prosumers
«requires» Communication
«requires» DateTimeCapability
«requires» ILocalPowerCommunicationCapa...
«requires» SuperviseLoadCapability
«requires» LateralControllerCommunicationC...
«requires» AggregationCapability
«requires» DistributeLoadDirectCapability
«Role»
Manage Prosumer Role
«achieves» Manage Prosumer
«requires» Communication
«requires» DateTimeCapability
«requires» ILocalPowerCommunication...
«requires» ManageLoadCapability
«requires» AssessReactivePowerCapability
«requires» AssessReactivePowerQualityC...
«requires» SmartMeterCapability
«requires» SmartInverterControlCapability

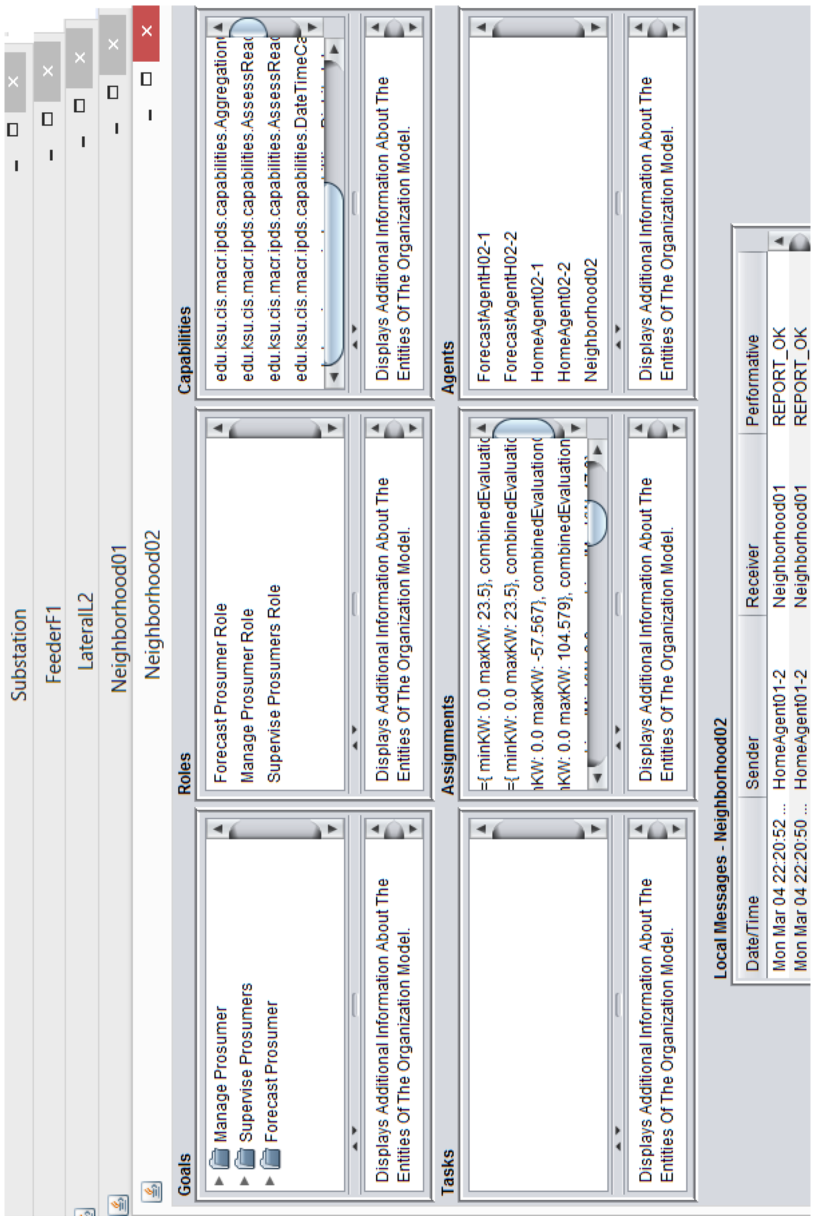
As communications are added to plan diagrams, they include the specification of the performative, the type of message content, and the role of the agent with which the communication takes place. Message classes and their associated message content classes were implemented for each communication capability.

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.

Fig. 8 shows a view into a running IPDS system. There is one debug window for each organization agent. The organization goals appear in each top left panel. Roles appear in the top center panel. Participating agents are shown in the lower right panel. In the center bottom panel, the 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, we can see the current values of the agent's goal parameters as they are adjusted by their local organization supervisor. At this point in the simulation, the prosumer goals are being distributed in accordance with each participant's demand. Maximum kW guidelines may be positive or negative. Negative upper boundaries can be assigned to a participant generating more PV power than the participant is consuming.

The extended system has passed a variety of tests and demonstrates the operation of multiple goal models and assignments. During this most recent iteration, the system has been extended to additional levels of the holarchy in preparation for the evaluation of the initial test case shown in Fig. 9, involving 62 location-based hosts and approximately 46 local IPDS organizations. This test case is based on the IEEE Distribution System Analysis Subcommittee 37-Node Test Feeder case [14] that begins with the substation node labeled “1”. Electricity is distributed out along the 3-phase feeder lines. Extensions to the IEEE test case have been made to test the system down to the home level. In our version, we have added four nodes along a single-phase lateral line (39-42), four nodes corresponding to agents running on neighborhood transformers (43,48,53,58), with four homes being supplied by each transformer. In our first trials, one of each of the four homes is equipped with roof-mounted solar PV panels. For example, Home 44 will have solar generation capabilities, but Home 45, 46, and 47 under Neighborhood Transformer 43 will not.


Substation

FeederF1

LateralL2

Neighborhood01

Neighborhood02

Goals
- Manage Prosumer
- Supervise Prosumers
- Forecast Prosumer

Displays Additional Information About The Entities Of The Organization Model.

Roles
- Forecast Prosumer Role
- Manage Prosumer Role
- Supervise Prosumers Role

Displays Additional Information About The Entities Of The Organization Model.

Capabilities
- edu.ksu.cis.macr.ipds.capabilities.Aggregation
- edu.ksu.cis.macr.ipds.capabilities.AssessRea
- edu.ksu.cis.macr.ipds.capabilities.AssessRea
- edu.ksu.cis.macr.ipds.capabilities.DateTimeCa

Displays Additional Information About The Entities Of The Organization Model.

Tasks

Displays Additional Information About The Entities Of The Organization Model.

Assignments
- ={ minKW: 0.0 maxKW: 23.5}, combinedEvaluatic
- ={ minKW: 0.0 maxKW: 23.5}, combinedEvaluatic
- ıKW: 0.0 maxKW: -57.567}, combinedEvaluation
- ıKW: 0.0 maxKW: 104.579}, combinedEvaluation

Displays Additional Information About The Entities Of The Organization Model.

Agents
- ForecastAgentH02-1
- ForecastAgentH02-2
- HomeAgent02-1
- HomeAgent02-2
- Neighborhood02

Displays Additional Information About The Entities Of The Organization Model.

Local Messages - Neighborhood02

Date/Time	Sender	Receiver	Performative
Mon Mar 04 22:20:52 ...	HomeAgent01-2	Neighborhood01	REPORT_OK
Mon Mar 04 22:20:50 ...	HomeAgent01-2	Neighborhood01	REPORT_OK

Fig. 8. Simple IPDS Holarchy (Substation, Feeder, Laterals, Neighborhoods)

Each number on the test case diagram corresponds to a physical location or node that could host IPDS agents. These locations may have sensors and/or actuators depending on the physical configuration being simulated. Generally, we assume that real power (P) and reactive power (Q) consumption (load) values are available by

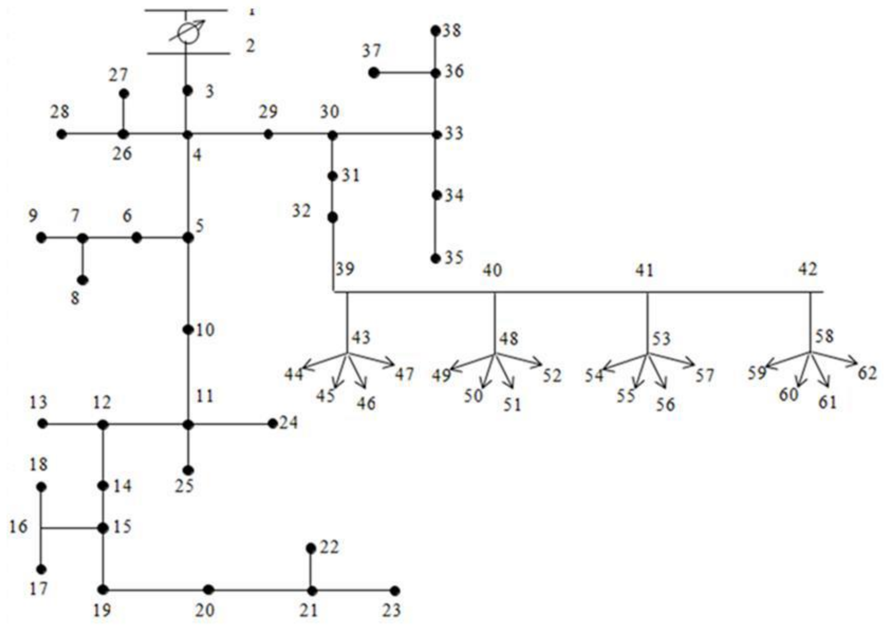


Fig. 9. Initial 62-Node IEEE Test Case**.**

phase at each of the nodes. In addition, homes equipped with PV may have sensor readings available for the real power generated. Actuators or controllable equipment range 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 allow for some moderation of the 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. A summary of the eleven data values calculated by the MatLab simulation engine is shown in Table 1. The simulation will receive new simulated readings every second for each of the 62 nodes shown above. A similar array will be used to provide calculated control values back to the MatLab data simulation in order to calculate the next set of simulated sensor readings.

7 Software Engineering Challenges

Architecting complex, adaptive systems can be challenging. Employing existing components can facilitate design, but a true understanding of the interrelationships between framework components may take a while to develop. Designing a new organization-based HMAS for IPDS required developing a detailed understanding of the O-MaSE metamodel, the OBAA framework, and GMoDS, and understanding the

Table 1. Simulated Second-by-Second Data Values (calculated in MatLab).

Abbrev	Description	Phase	Type	Value
PA	Column 1 - Phase A - P (load)	A	Load	P
QA	Column 2 - Phase A - Q (load)	A	Load	Q
PB	Column 3 - Phase B - P (load)	B	Load	P
QB	Column 4 - Phase B - Q (load)	B	Load	Q
PC	Column 5 - Phase C - P (load)	C	Load	P
QC	Column 6 - Phase C - Q (load)	C	Load	Q
PG	Column 7 - P (generation)	-	Gen	P
QG	Column 8 - Q (generation)	-	Gen	Q
VA	Column 9 - Voltage Phase A	A	Load	Voltage
VB	Column 10 - Voltage Phase B	B	Load	Voltage
VC	Column 11 - Voltage Phase C	C	Load	Voltage

connections between the various elements, how they were related to code structures and most importantly, how they drive behavior during execution. Initial attempts took longer than expected. The AO-MaSE process was developed as a way to illustrate the system design in a concrete manner. Standards have allowed greater automation and agentTool3 has been updated. The process of debugging configuration files resulted in the creation of IPDS builder factories that were used to generate additional test cases. Strict naming and documentation requirements improve clarity and consistency. Although the number of classes required is substantial, the focus of each is such that enhancements have been proceeding in parallel. The IPDS simulation must be able to evaluate control strategies that are not yet defined and the ability to quickly respond to new requirements and system enhancements will continue to be crucial. Additional automated testing offers support for evolutionary refactoring as the system functionality expands and initial experiments with new specification and testing frameworks appears promising.

8 Results

The project resulted in the development of a standard, repeatable process for implementing IPDS simulations. 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 used to test implementation of the required goals, roles, capabilities, and plans required for the initial IEEE test cases developed by the electrical engineering simulation team. The AO-MaSE process was employed during the development of a new type of organization that will allow the additional implementa-

Table 2. Implementation of O-MaSE-compliant MAS with and without AO-MaSE.

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 post-fix object type names (e.g. SmartMeter*Capability*, ManagePower*Goal*) improves code readability and maintainability.	Post-fix 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.

tion of self-management for agents participating in multiple organizations. The process and tools provided a path that has allowed implementation of new iterations of end-to-end functionality within days. A comparison of O-MaSE compliant systems, some of which were developed using the AO-MaSE guidelines and some of which were not, is shown in Table 2.

9 Conclusions

This paper describes the iterative design and construction of an architecture prototype for an intelligent power distribution system with the AO-MaSE process. 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 early feedback into the behavior of a complex, recursive HMAS. Architectures that do not perform as expected can be abandoned sooner and alternate versions can be tested. 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 MAS. Specific recommendations are included with examples taken from the initial IPDS implementation.

10 Future Work

Our initial iterations have allowed us to test several critical issues early. The next phase will involve an initial assessment of potential self-organizing abilities, grid-based leader election algorithms, implementation of voltage/var control strategies during periods of renewable intermittency, initial critical power supply strategies, and reconnection approaches after islanded operation. A real-time visualization of PDS operation will be created and agent negotiation and support capabilities will be tested between complementary lateral power lines in our 62-node test case and additional extended test cases. The AO-MaSE process will continue to be employed and refined to support a more adaptive implementation of evolving functionality. Additional implementation of agile-recommended testing strategies may be included as well, based on some of the interesting work being done with SADAAM, test agents, and test-driven development in agile agent-oriented software engineering [6].

11 References

1. Argente, E., Julian, V., Botti, V.: MAS modeling based on organizations. Agent-Oriented Software Engineering IX, pp. 16-30 (2009)
2. Asano, H.: Holonic Energy Systems: Coevolution of Distributed Energy Resources and Existing Network Energy, International Symposium on Distributed Energy Systems and Micro Grids, The University of Tokyo, December 7-8 (2005)
3. Baxevanos, I. S., Labridis, D. P.: Implementing multiagent systems technology for power distribution network control and protection management. Power Delivery, IEEE Transactions on, 22(1), pp. 433-443 (2007)
4. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M Thomas, D.: Manifesto for agile software development. The Agile Alliance, 2002-04 (2001)
5. Chella, A., Cossentino, M., Sabatucci, L., Seidita, V.: Agile PASSI: An agile process for designing agents. International Journal of Computer Systems Science & Engineering, 21(2), pp. 133-144 (2006)
6. Clynch, Neil, and Rem Collier: Sadaam: Software agent development-an agile methodology. Proceedings of the Workshop of Languages, methodologies, and Development tools for multi-agent systems (LADS'007), Durham, UK (2007)
7. Cossentino M, Gaud N, Hilaire V, Galland S Koukam A.: ASPECS: an agent-oriented software process for engineering complex systems. J of Auton Agents and Multiagent Syst. 20, pp. 260-304 (2009)

8. Cossentino, M, Gleizes, M, Molesini, A, Omicini, A.: Processes engineering and aose. In Marie-Pierre Gleizes and Jorge Gomez-Sanz, editors, Agent-Oriented Software Engineering X, volume 6038 of Lecture Notes in Computer Science, pp. 191–212. Springer Berlin / Heidelberg (2011)
9. Cossentino, M., Gaud, N., Galland, S., Hilaire, V., Koukam, A.: A holonic metamodel for agent-oriented analysis and design. Holonic and Multi-Agent Systems for Manufacturing, pp. 237-246 (2007)
10. DeLoach, S. A., Garcia-Ojeda, J. C.: O-MaSE: A customisable approach to designing and building complex, adaptive multi-agent systems. International Journal of Agent-Oriented Software Engineering, 4(3), pp. 244-280 (2010)
11. Gaud, N., Galland, S., Hilaire, V., Koukam, A.: An organisational platform for holonic and multiagent systems. Programming Multi-Agent Systems, pp. 104-119 (2009)
12. Gómez-Rodríguez, Alma M., González-Moreno, Juan C.: Comparing agile processes for agent oriented software engineering. Proceedings of the 11th international conference on Product-Focused Software Process Improvement (PROFES'10), M. Ali Babar, Matias Vierimaa, and Markku Oivo (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 206-219 (2010)
13. Huang, K., Cartes, D. A., Srivastava, S. K.: 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), pp. 1016-1021 (2007)
14. IEEE Power and Energy Society: Distribution Test Feeders. http://ewh.ieee.org/soc/pes/dsacom/testfeeders/.
15. Ionita, S,: Multi Agent Holonic Based Architecture for Communication and Learning about Power Demand in Residential Areas, Machine Learning and Applications, Fourth International Conference on, pp. 644-649. 2009 International Conference on Machine Learning and Applications (2009)
16. Jennings, N. R.: On agent-based software engineering. Artificial intelligence, 117(2), pp. 277-296 (2000)
17. Jiang, Z.: Agent-based power sharing scheme for active hybrid power sources. Journal of power sources, 177(1), pp. 231-238 (2008)
18. Kim, H. M., & Kinoshita, T.: Multiagent system for Microgrid operation based on power market environment. In Telecommunications Energy Conference (INTELEC). 31st International, pp. 1-5. IEEE (2009)
19. Linnenberg, T., Wior, I.,Schreiber, S. Fay, A.: A market-based multi-agent-system for decentralized power and grid control, Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on , pp.1-8, 5-9 Sept (2011)
20. Malekpour, A.R., Pahwa, A.: Reactive power and voltage control in distribution systems with photovoltaic generation, North American Power Symposium (NAPS), pp.1-6, Sept (2012)
21. Oyenan, W. H., DeLoach, S. A.: Towards a systematic approach for designing autonomic systems. Web Intelligence and Agent Systems, 8(1), pp. 79-97 (2010)
22. Pahwa A, DeLoach S.A., Das S., Natarajan B., Ou X., Andresen D., Schulz N., Singh G.: Holonic multi-agent control of power distribution systems of the future. In: CIGRE Grid of the Future Symposium (2012)
23. Vishwanathan, V., McCalley, J., Honavar, V.: A multiagent system infrastructure and negotiation framework for electric power systems. In Power Tech Proceedings, 2001 IEEE Porto, Vol. 1, pp. 6. IEEE (2001)

24. Zabet, I., Montazeri, M.: Decentralized control and management systems for power industry via multiagent systems technology, Power Engineering and Optimization Conference (PEOCO), 2010 4th International, pp. 549 – 556 (2010)
25. Zhong Z, McCalley, J.D., Vishwanathan, V., Honavar, V.: Multiagent system solutions for distributed computing, communications, and data integration needs in the power industry, Power Engineering Society General Meeting, Vol.1, pp. 45-49. June (2004)
26. Zhou, Z., Chan, W. K. V., Chow, J. H.: Agent-based simulation of electricity markets: a survey of tools. Artificial Intelligence Review, 28(4), pp. 305-342 (2007)