Supporting Requirements Reuse in Multi-Agent System Product Line Design and Evolution

Josh Dehlinger
Charles L. Brown Department of Electrical and Computer Engineering
University of Virginia
jdehlinger@virginia.edu

Robyn R. Lutz
Department of Computer Science
Iowa State University and Jet Propulsion Laboratory / Caltech
rlutz@cs.iastate.edu

Abstract

A principal goal of agent-oriented software engineering (AOSE) is to provide the mechanisms for reusing, maintaining and allowing the evolution of agent-based software systems. Our AOSE methodology, Gaia-PL, enables the design and development of multi-agent system product lines (MAS-PL)[1] *by providing the software engineering processes to define and reuse requirements specifications and design artifacts. In this paper we extend our Gaia-PL methodology with automated tool support to enable the reuse and verification of MAS-PL requirements to better facilitate specification reuse during both initial system development and evolution. Specifically, we show how use of our product-line requirements management and verification tool along with feature modeling can support correct variation point selection, reuse and MAS-PL evolution. We illustrate and evaluate this work through an application to a proposed NASA agent-based pico-spacecraft swarm.*

1. Introduction

Asset reuse is highly desirable in software engineering. Methods for software reuse have been a driving force in significantly reducing the time and cost of software development, and industry's demand for shorter, lower-cost software development cycles encourages software methodologies to exploit reuse principles whenever possible [9] [13].

Agent-oriented software engineering (AOSE) provides viable, high-level abstractions, models and approaches for designing and developing the autonomous agents of a multi-agent system (MAS) [14]. A principal goal of AOSE is to provide the mechanisms for reusing and maintaining agent-based software systems [12]. Thus, the viability of AOSE as a software development paradigm partially depends on its ability to achieve reductions in design, development and maintenance/evolution time and cost similar to other reuse-conscious software engineering paradigms.

Although current AOSE methodologies provide natural, high-level abstractions in which software engineers can understand, model and develop complex multi-agent systems, no AOSE methodology has provided the reuse mechanisms at an early stage in the software development life cycle to allow for reuse during initial system design, development, evolution and maintenance [4]. A few AOSE approaches, including [4] and [5], have been proposed, but, in each case, the reuse is exploited during the later stages of design and development (e.g., reusing a component from a component repository [5]). The work described here differs from this previous work in that we present an approach, based on software product-line engineering (SPLE) [9] [13], to capture the reuse potential of the AOSE assets of a multi-agent system product line (MAS-PL) in the requirements analysis and specification stages. MAS-PLs are an emerging research field that focuses on designing and developing agents from reusable software engineering assets.

In previous work [2] [3] we have described our extension, Gaia-PL, to an AOSE methodology, Gaia [14], to enable capturing and reusing AOSE assets developed during the design and development of a MAS-PL using SPLE so that future agent-based systems can be built quickly and easily. However, this initial work provided no assurance that the selection of the roles and variation points was valid and consistent with the constraints and/or dependencies among the roles and variation points (e.g., constraints placed on a variation point necessitating the inclusion or exclusion of other variation points based on design decisions, safety analysis results, etc.).

[1] The term multi-agent system product line (MAS-PL) was coined by Peña et al. in [8] for what we initially called product-line, multi-agent systems in [2] and [3].

The current paper addresses this limitation and extends our Gaia-PL work by detailing how the requirements of a MAS-PL can be documented and reused during system evolution and maintenance by using an SPLE requirements management and verification tool, DECIMAL [7], and feature modeling [9] (i.e., a model to illustrate the relationship among groups of related requirements). In this context, we define *system evolution* as either the updating of an existing agent(s) in a deployed system or the inclusion of additional agents or features in the system. Specifically, the contributions of this paper include:

- Extension of the Gaia-PL methodology to enable the efficient documentation and reuse of AOSE assets during an early stage of the development lifecycle and during maintenance and evolution
- Depiction of how Gaia-PL, aided by the use of feature models, addresses a limitation in Gaia [6] to allow the design of an agent, an agent's role(s) and its variation points in a hierarchical manner
- Description of how DECIMAL can verify and ensure proper reuse of MAS-PL features and requirements in Gaia-PL from a previously developed agent and apply them to a new, slightly different agent during maintenance and evolution
- Application of Gaia-PL on a case study to illustrate and evaluate reuse

The contributions presented here are part of a larger effort to investigate how SPLE can be used in AOSE to produce reusable assets in the requirements specifications, design, maintenance and evolution of an agent-based software system. The long term goal is to provide AOSE methods to develop new agents of a MAS-PL in a timely, cost-effective manner.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of the SPLE and AOSE techniques used in this work as well as related work in developing MAS-PL. Section 3 describes the case study used to motivate and demonstrate our work. Section 4 presents our tool-supported approach for defining and managing the requirements of a MAS-PL using the case study. Section 5 explores how to reuse and verify the requirements for a MAS-PL during initial system development and evolution. Section 6 briefly details the results of an evaluation to demonstrate the reuse inherent in our approach. Finally, Section 7 provides some concluding remarks.

2. Related Work and Background

The work presented in this paper builds upon software product-line engineering and agent-oriented software engineering (AOSE) to provide a requirements-based, reuse-oriented AOSE approach for developing multi-agent system product lines (MAS-PL). In this section, we summarize previous and related work in these areas of software engineering.

SPLE supports the systematic development of a set of similar software systems by understanding, controlling and managing their common, core characteristics (i.e., commonalities) and differences (i.e., variabilities) [9]. SPLE models provide software engineers with development approaches that can contribute to significantly reducing both the time and cost of software requirements specification, design, maintenance and evolution [9]. SPLE methodologies create a line of products and rely on the analysis of the commonalities and variabilities of the members of the product line prior to the design or development of any software engineering artifacts [9].

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission [13]. Product-line requirements are often documented in a Commonality and Variability Analysis (CVA) [13]. The CVA provides a comprehensive definition of the product-line requirements including the *commonalities* (i.e., requirements of the entire product line) and the *variabilities* (i.e., specific features not contained in every member of the product line) of the product line. Some variabilities are *optional* while other variabilities offer *alternative* choices. Finally, the CVA documents the product-line *dependencies* (i.e., constraints amongst selection of the variable features). We additionally utilize a feature model to depict the relationship among grouped common and variable requirements and identify candidate roles and variation points for the MAS-PL to facilitate the reuse during evolution.

We use Weiss and Lai's two-phased Family-Oriented Abstraction, Specification and Translation (FAST) product-line approach to analyze and design software product lines [13] into AOSE. The first phase, the domain engineering phase, defines the product-line requirements (i.e., the commonalities, variabilities and dependencies). The second phase in the FAST process, the application engineering phase, reuses the artifacts generated in the previous phase to develop new product-line members (i.e., agents). We use this model in the work presented here within the context of Gaia's development phases, as shown in Figure 1.

A multi-agent system (MAS) is an application that is "designed and developed in terms of autonomous software entities that can flexibly achieve their objectives by interacting with one another in terms of high-level protocols and terms" [14]. AOSE's objective

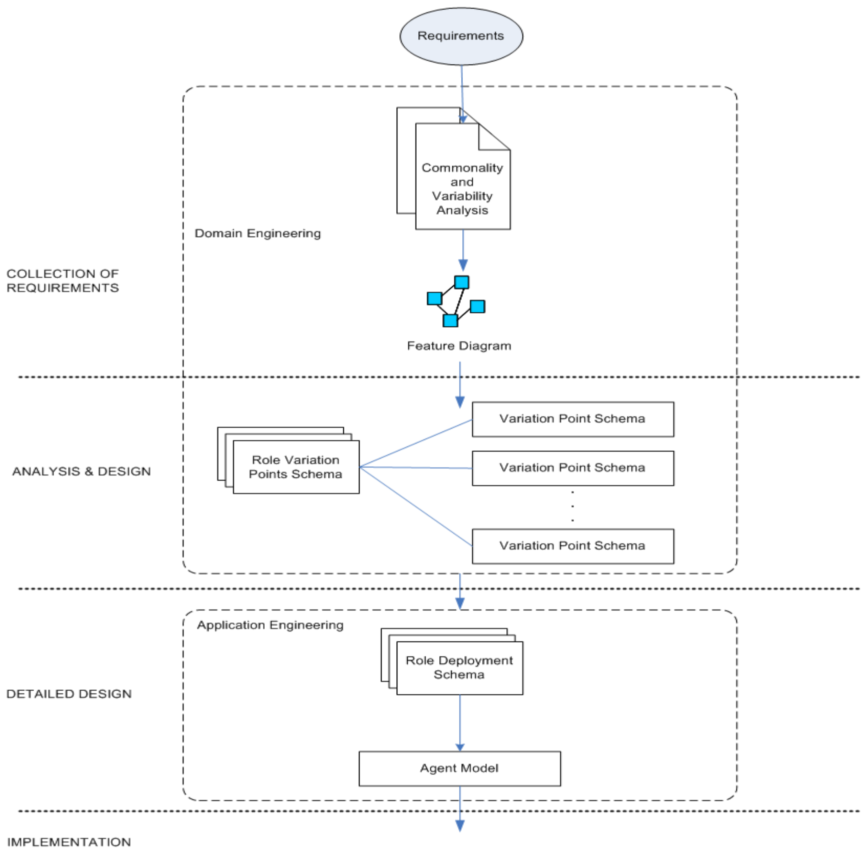


Figure 1. An overview of Gaia-PL phases and assets within the context of FAST and Gaia

is to provide software engineers with the design methodologies to develop the agents of a MAS to address the solution of a particular problem.

AOSE methodologies provide tools and techniques for abstracting, modeling, analyzing and designing MAS early in the development lifecycle. The Gaia methodology adopts an organizational metaphor where each agent within a MAS may play a variety of roles that have specific objectives and where the agents cooperate with each other to accomplish a common organizational goal.

This work builds upon an existing the Gaia [14] methodology to support the development of MAS-PL. We selected the Gaia methodology for this work for several reasons. First, it was a mature AOSE methodology (i.e., it spans from the analysis to the design phases of MAS development). Second, it is an established, well-documented methodology in the AOSE community. Third, the Gaia methodology's design and development process and models provide the best fit with the phases of the product-line engineering approach used in this work. Finally, it is flexible enough that the adaptation of product-line approach for building MAS-PL in Gaia could be further included in other AOSE methodologies.

Our initial work for integrating reuse via SPLE in Gaia-PL [1] [2] [3] addressed some of Gaia's limitations [6] and focused on: (1) devising a method to document agent requirements specifications during the analysis and design phases when an agent must be updated to include new functionality, and (2) providing a process in which the requirements specification schemas, extended from Gaia [14], developed during the analysis phase could be reused for the agents during initial system development or during system evolution. We achieved this by extending Gaia's requirements specification schemas and design process for the agents of a MAS to allow for documentation of common and variable agent behaviors so that reusable assets could be exploited during initial development.

In addressing these limitations, we used the term *variation points* for the differing protocols, activities, permissions and responsibilities available to a specific role of an agent. *Activities* are the computations associated with the role that can be executed without interacting with other agents. *Permissions* are the information resources a role has to read, change and generate. *Responsibilities* define the functionality of a role and are divided into liveness properties and safety properties. A variation point affords the definition of a role greater flexibility and partitions some functionality of a role depending on the agent and system's current configuration. The variation point notion is important because it aids in capturing the different arrangements of agents and promotes reuse. Thus, using this construct, *an agent's role may have one or more variation points*, and the design of an agent's role includes a selection of the role's associated variation points (i.e., the differing protocols, activities, permissions and responsibilities specific to a role). The set of roles and variation points that an agent contains is its *configuration*.

More recently, Peña et al. developed the Methodology for Analyzing Complex Multiagent Systems using a product-line approach to build MAS-PL [8]. Their approach uses and extends our idea of incorporating product-line techniques into AOSE originally reported in [2] [3]. Like this work, Peña et al. utilize a feature model to document the commonalities and variabilities of the MAS-PL [8]. The work presented here, however, differs from that of [8] in that it focuses on the reusability of the MAS-PL's requirements assets both for initial system development and during system evolution rather than building the core architecture.

3. The Prospecting Asteroid Mission

To illustrate, motivate and evaluate this work, we use requirements based on the Prospecting Asteroid Mission (PAM) [8] [11]. The PAM is a 2020-2025 NASA concept mission lasting 5-10 years. It is based on the Autonomous Nano-Technology Swarm technology and is meant to explore the asteroid belt between Mars and Jupiter [11]. The proposed PAM mission will consist of up to 1,000 pico-spacecraft (spacecraft weighing less than 1 kilogram) that will autonomously form subswarms to investigate asteroids of interest in the asteroid belt. The PAM is a product line since, except for a spacecraft's scientific instrumentation specialties, each PAM spacecraft will have identical hardware and core functionality of similar software (i.e., product-line commonalities) to perform necessary functions (e.g., using its solar sails for navigation, avoiding collisions, etc.). Thus, implicit in the mission description is the high degree of reuse needed for its design, development and evolution.

Each PAM spacecraft will be designated as having a *leader*, a *messenger* or a *worker* role (i.e., a product-line alternative variability) [11]. A spacecraft tasked as a *leader* (approximately 10% of the total number) will determine the types of asteroids and data the mission is interested in and will coordinate the efforts of other spacecraft to investigate asteroids to satisfy mission goals. A spacecraft designated as a *messenger* (approximately 10%) will coordinate the communication messages between the *worker* and *leader* spacecraft. Each w*orker* spacecraft (approximately 80%) will contain a single specialized, scientific instrument (e.g., a spectrometer, magnetometer, etc.) and perform targeted scientific investigation using its specialized equipment. Dependencies amongst the variabilities and selection of the variable roles enforce constraints (e.g., based on specialized instrumentation, resources available, etc.).

High redundancy of the spacecraft is needed due to NASA's projection that 60%-70% of the spacecraft will be lost over the duration of the mission due to failures, collisions, etc. [11]. Due to the high anticipated loss, the PAM spacecraft are required to autonomously undertake reconfiguration and maintenance steps. For example, some spacecraft will be able to switch from a *leader* role to a *messenger* role or vice versa if needed (e.g., due to loss and/or failure of spacecraft, etc.).

Further, to protect the swarm from solar radiation, some spacecraft will be tasked, in addition to their other roles, with monitoring the sun for an impending solar storm (i.e., a product-line optional variability). Not all of these spacecraft will be actively monitoring the sun although they all have the capability. Rather, some spacecraft will switch to actively monitoring the sun only when needed (e.g., when spacecraft have been lost). A detailed description including all the requirements for the PAM case study can be found in [1].

Challenges to multi-agent system development. The proposed PAM swarm presents significant challenges to its design and development as an agent-based system:

- Size of the design space. The allowed variability (64 high-level variability requirements in our case study), together with the core functionality (35 high-level commonality requirements) present a large number of possible spacecraft for which to design. There were 160 unique configurations in our case study [1].
- High degree of reuse. Some of the features of the spacecraft were commonalities that necessitate reuse at the requirements level to ensure traceability and identification of agent roles and variation points to use in the requirements specifications [2].

- Hierarchical and changing roles. The hierarchical nature of the roles and variation points requires an approach that allows for hierarchical design and accounts for the changing of a role's behavior depending on context (e.g., environment, failures, etc.) to autonomously address multi-agent system product line (MAS-PL) maintenance needs.
- Constraint verification. A high number (124 in our case study) of dependencies among the variable features necessitates an automated verification method to ensure the selected variation points of an agent during initial design, development and evolution do not violate the constraints (i.e., dependability/safety results, resources available, etc.) of the MAS-PL.

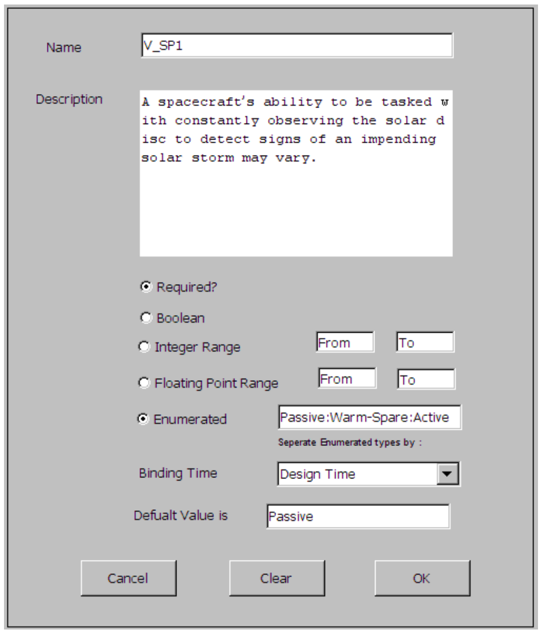
We describe below how the extension of Gaia-PL presented in this paper addresses these challenges using the PAM case study to illustrate the solutions.

4. Documenting Reusable MAS-PL Requirements in Gaia-PL

The Collection of Requirements phase of the Gaia-PL methodology, shown in Figure 1, involves identifying and documenting the multi-agent system product line's (MAS-PL) commonality and variability requirements and dependencies/constraints such that they can be easily reused for the agents of the MAS-PL and during evolution. This section describes the tool-supported documentation and management of MAS-PL requirements and dependencies; the development, organization and identification of roles and variation points; and the elaboration of the derived requirement specifications in the design phases of Gaia-PL [2] [3].

4.1. Tool-Supported MAS-PL Requirements Documentation and Management

Previously, Gaia-PL's method of documenting the requirements of a MAS-PL has been in a textual Commonality and Variability Analysis (CVA) [13]. From the CVA, the MAS-PL requirements can then be further refined so that detailed requirements specifications can be documented. Included within the CVA are the constraints among the variabilities arising from design decisions, safety implications, resource constraints, etc. For a large MAS-PL, such as the Prospecting Asteroid Mission (PAM) swarm, with many constraints, automated CVA tool-support is important to ensure efficient verification in the selection of variability requirements for an agent of the MAS-PL and enable and support reuse.


Name: V_SP1

Description: A spacecraft's ability to be tasked w ith constantly observing the solar d isc to detect signs of an impending solar storm may vary.

[✓ Required?]

Boolean

Integer Range From To

Floating Point Range From To

[✓ Enumerated] Passive:Warm-Spare:Active

Seperate Enumerated types by :

Binding Time: Design Time

Defualt Value is: Passive

Figure 2. MAS-PL variability requirement in DECIMAL

Thus, to extend Gaia-PL to support large, complex MAS-PL and facilitate requirements reuse, we utilize DECIMAL [7], a product-line requirements and management and verification tool. Figure 2 illustrates the documentation of a MAS-PL variability requirement in DECIMAL (for full details of DECIMAL's implementation see [7]). It contains a description, type (i.e., Boolean, Integer, Floating Point or Enumerated), binding time (i.e., the point at which the configuration of the variability requirement's parameter must be selected [13]) and the default value. DECIMAL also provides a mechanism to document dependencies among MAS-PL variability requirements. Within DECIMAL users are provided an intuitive interface to generate constraint rules such as

```
IF V_SP1=ACTIVE THEN V_G1=LEADER ||
          V_G1=MESSENGER
```

This specifies that any spacecraft that has the ability to actively monitor the sun for impending solar storms cannot also have the *worker* role.

The CVA and the requirements documented in DECIMAL provide insight into how to define the roles and the variation points possible in a role for the MAS-PL. The use of DECIMAL also facilitates the creation of a feature model to identify roles and a role's variation points [9] (cf. Section 4.2), and aids in structuring the requirement specifications schemas (cf. Section 4.3) that will assist in requirement reuse during maintenance and evolution (cf. Section 5).

4.2. Hierarchical Design of Roles and Variation Points Using Feature Models

To manage the many possible combinations of variations in the PAM case study, we found it both useful and straightforward to structure both the commonality and variability requirements that were documented in DECIMAL into logical, functional groups. For example, it was useful to group variable requirements according to whether they were associated with *leader*, *messenger* or *worker* roles, as described in Section 3.

These groupings of the requirements also helped guide identification of the roles for the agents. The grouping of variabilities in the Parameters of Variation table [1] (not discussed here due to space limitations) helped define the possible role variation points of the MAS-PL. For example, using the high-level requirement from Figure 2, together with other requirements regarding how the PAM spacecraft shall differ in their functionality and participation in monitoring the sun for impending solar storms, we derived three variation points for the agent role of monitoring and warning of solar storms: passive (i.e., only relaying warning messages of an impending storm), warm-spare (i.e., acting as a backup in monitoring for impending storms) and active (i.e., observing the sun for an impending storm).

From these identified variation points for a role, we can partition the core functionality of a variation point as defined by the default value detailed in DECIMAL. In doing this, a hierarchical design of the features (i.e., groupings of requirements) and subfeatures is developed. This feature model illustrates the relationships (i.e., mandatory/optional) amongst features and subfeatures. In [9], a method to develop a feature model from the requirements of a CVA is given and, thus, we do not cover this process here.

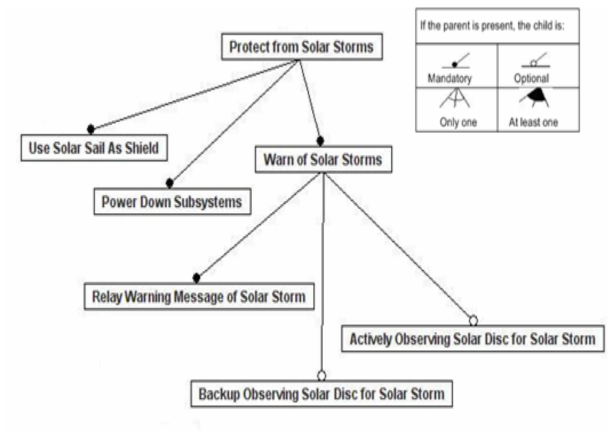

Protect from Solar Storms
Use Solar Sail As Shield
Power Down Subsystems
Warn of Solar Storms
Relay Warning Message of Solar Storm
Actively Observing Solar Disc for Solar Storm
Backup Observing Solar Disc for Solar Storm
If the parent is present, the child is:
Mandatory
Optional
Only one
At least one

Figure 3. Excerpt feature model for the PAM MAS-PL

In Figure 3 shows the feature model for a portion of the PAM case study using the variability requirement from Figure 2 to illustrate the partitioning of the common and variable features that will be found in the variation points of a role. The full feature model of the PAM case study included 47 features derived from the requirements documented in the CVA. Figure 3 also illustrates the hierarchical design of the roles and variation points that will be used to detail the requirement specifications, described in Section 4.3.

To summarize, in the application of this process to the PAM case study, we found that documenting a MAS-PL's requirements in a CVA using the tool support offered by DECIMAL, the construction of a Parameters of Variation table to group similar requirements and the use of a feature model to capture the relationships among features simplified the process of role and variation point identification.

4.3. Development of Reusable Requirements Specifications

The Analysis and Design phase of our Gaia-PL methodology develops the requirements specifications from the requirements documented in the CVA and DECIMAL during the Collection of Requirements Phase, as illustrated in Figure 1, using adaptations of Gaia's schemas [14] such that they can be reused for future systems of a MAS-PL [2] [3].

In our Gaia-PL methodology, roles and variation points are documented in two schemas: the *Role Variation Point Schema* and the *Variation Point Schema*. The Role Variation Point Schema provides the detailed requirements and description of the role and a description of the possible variation points of the role, denoting the mandatory variation point(s) and optional variation point(s). The Variation Point Schema details the activities, protocols, permissions and responsibilities specific to that role's variation point(s).

The feature model, described in Section 4.2, facilitated the identification of the roles and variation points of the MAS-PL. The following heuristics supported their identification:

- Any feature that has mandatory and optional subfeatures is a candidate role with its subfeatures as candidate variation points (e.g., in Figure 3, the feature "Warn of Solar Storms" includes the requirements of a candidate role)
- Any feature that has an "Only One" or "At Least One" cardinality is a candidate role with its subfeatures as candidate variation points
- Any mandatory feature that has no children but has a sibling with children that match one of the

above rules is either a candidate role, with no variation points, or should include its functionality within one of its siblings (e.g., in Figure 3, the feature "Use Solar Sail as Shield" could be consolidated within the "Warn of Solar Storms" feature as a role or become its own role)

- For any feature labeled as a candidate role using the above rules, consider consolidating its functionality with its parent feature's functionality to constitute a role, along with the already identified candidate variation points

These informal heuristics were found to provide sufficient guidance to efficiently identify the roles and variation points for the PAM swarm and structure the requirements specifications schemas for the MAS-PL in a hierarchical manner.

Application of the Analysis and Design phase, described in [2] [3], to the PAM case study, described in Section 3 and fully in [1], produced 13 roles, detailed in Role Variation Point Schemas, and 39 variation points, detailed in Variation Point Schemas. For example, from the partial feature model shown in Figure 3, we identified the role "Solar Storm Warner" from the high-level feature "Warn of Solar Storms" having 3 variation points (1 mandatory, 2 optional) derived from this feature's subfeatures. From this identified role and its variation points, 3 Variation Point Schemas were developed to elaborate and refine the high-level requirements of the feature into concise, actionable requirements specifications that detail the protocols, activities, permissions and responsibilities available and unique to the variation point. Similarly, as described in Section 3, Role Variation Point Schemas were documented for the *leader*, *messenger* or *worker* roles that differentiate the spacecraft of the PAM to denote their differing functionalities.

The identified roles and variation points were further documented in Gaia-PL's Agent Model (cf. Figure 4), an extension of Gaia's [14] Agent Model, to graphically depict the assignment of roles to agents as well as variation points to roles, similar to that of the feature model, shown in Figure 3. The cardinality relationship between agent and role was indicated, and all possible variation points were listed for each role. At runtime, the designer annotates the actual cardinality and the specific possible variation points of an agent instance (typically a one-to-one relationship).

This section described the domain engineering phase of software product-line engineering in which the assets for the entire MAS-PL are developed. The application engineering phase makes use of the assets previously defined and reuses them for the agents of a MAS-PL. We describe this further in the next section.

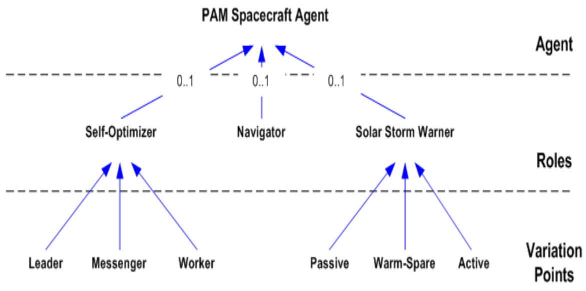


Figure 4. Excerpt Agent Model for the PAM MAS-PL

5. Reusing MAS-PL Requirements in Gaia-PL

This section describes how the multi-agent system product line's (MAS-PL) requirements and requirements specifications, detailed in Section 4, can be reused during the initial deployment of a MAS-PL as well as during maintenance and system evolution (e.g., new agents, roles or variation points). Requirements reuse involves using requirements and requirements specifications previously defined from an earlier system and applying them to a new, slightly different system. Increasing the requirements reuse for a new system can be an effective way to reduce its production time and cost [9] [13]. Requirements reuse for a MAS-PL is simplified and made efficient in Gaia-PL through tool-supported management and verification to ensure that the configuration of an agent (i.e., the selection of the roles and variation points of an agent) satisfies the documented dependencies and constraints among the variation points of the MAS-PL.

5.1 Tool-Supported Requirements Reuse During Initial MAS-PL Development

The agents of a MAS-PL often will be heterogeneous in their functional capabilities yet mostly similar in structure. Heterogeneity may also arise when resources are limited, and different agents of a MAS-PL must assume different roles. Typically, different agents of a MAS-PL assume different roles, functional capabilities, intelligence levels or other possible variation points [2] [3]. Initially deployed members of a MAS-PL will likely contain a role(s) that differs amongst other members in terms of which variation points it is capable of assuming. Several agents of the MAS-PL may have the same role but at different levels of intelligence (e.g., the Prospecting Asteroid Mission spacecraft will have differing capabilities in their ability to monitor for impending solar storms).

Requirements reuse can be exploited during the initial design and development of a MAS-PL in Gaia-PL using MAS-PL requirements documented in DECIMAL. Specifically, rather than repeatedly defining the requirements of a role for any given agent (as would be necessary in Gaia [14]), DECIMAL allows the selection of roles and variation points for a new agent of a MAS-PL and efficiently verifies the selection against the MAS-PL dependencies (124 in the case study described in Section 3 and fully in [1]).

To make this approach scalable, DECIMAL *automatically verifies* that the proposed new agent's set of selected roles and variation points does not violate the defined dependencies. To do this, DECIMAL evaluates each defined dependency against the set of selected roles and variation points to detect any violations (e.g., two roles were selected that can not be selected together, a selected role without also selecting another dependent role(s), etc.). Note that full details of DECIMAL's constraint checking can be found in [7]. If any violations are discovered, DECIMAL flags them so that the developer can rectify the problem. This is important because it ensures that the design decisions and constraints are maintained for the agents of the MAS-PL. Further, the reuse of MAS-PL requirements extends and further facilitates the reuse of requirements specifications in Gaia-PL described in [2] [3].

5.2 Requirements Reuse During MAS-PL Maintenance and Evolution

Gaia-PL supports reuse during MAS-PL maintenance and evolution through the use of DECIMAL (cf. Section 4.1), the hierarchal roles and variation points supported by the feature model (cf. Section 4.2) and the requirements specification schemas (cf. Section 4.3).

After the initial deployment of a MAS-PL, the system can be expected to evolve such that future deployments will require additional and/or new functionality. In the case of the PAM swarm, even if 1,000 spacecraft may be initially deployed, additional spacecraft may have to be deployed if a significant number of spacecraft are lost due to damage or failure (e.g., solar radiation, collisions, etc.) [11]. The new PAM spacecraft deployed to replace the lost spacecraft are likely to contain features not found in previously deployed spacecraft. A deployed MAS-PL can be maintained and evolve in three ways relevant to this work: 1. new agents may be added to the system; 2. new roles with new functionality may be created that future agents can employ; and 3. new variation points may be added to existing roles that future agents can employ. The following paragraphs discuss how these types of evolution in a MAS-PL can be accommodated at the requirements level in Gaia-PL.

When a MAS-PL evolves, a new agent may be deployed to replace a destroyed or failing agent (i.e., MAS-PL maintenance). If this update includes functionality previously defined in the requirements and requirements specifications, it suffices to follow the process described in Section 5.1 as well as in the Detailed Design phase of Gaia-PL, shown in Figure 1 and described in [2] [3]. If, however, the evolution of the MAS-PL involves a new agent with functionality not previously defined in the requirements specifications, updates to the MAS-PL's requirements become necessary. In this case, the process of documenting a MAS-PL's requirements, detailed in Section 4, is used to accommodate system evolution by including new requirements while still reusing the previously documented requirements.

The addition of a new role and/or a new variation point(s) as a part of newly introduced functionality is analogous to the inclusion of a role during initial system development, as described in Sections 4.2 and 4.3. For example, after the deployment of the PAM swarm, mission engineers may decide to include an additional *scout* type of spacecraft (i.e., role) that would be tasked to quickly survey asteroids, assess their relevance to the mission goals and decide which asteroids should be further explored. The new *scout* type of PAM spacecraft will include some of the functionality of the *leader* and *worker* spacecraft (cf. Section 3) but will also include new functionality.

The inclusion of this new type of spacecraft in the PAM MAS-PL will necessitate the updating of portions of the feature model, requirements specifications and Agent Model (shown in Figure 5). Briefly, we document the new requirements and dependencies associated with the new *scout* role and/or variation point(s) using DECIMAL and then follow the process described in Section 4.3 to include the new requirements of the new *scout* role into the feature variation points and instantiate the requirements of a new agent with the new functionality. Again, the use of DECIMAL helps verify the configuration of a new agent of the MAS-PL and provides the automated tool-support needed to efficiently check the constraints and dependencies. Further, DECIMAL verifies the new agent at an early development stage to prevent design errors and avoid costly corrective maintenance later in the development lifecycle.

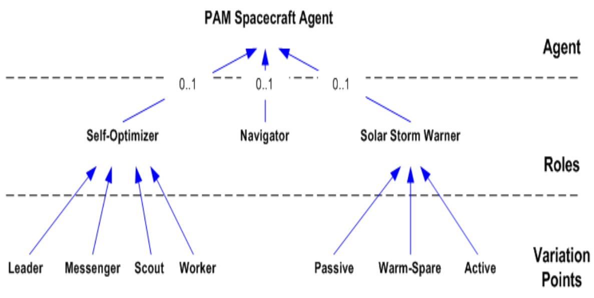


Figure 5. Excerpt Agent Model for the PAM MAS-PL after system evolution

Note that the introduction of new dependency requirements to the MAS-PL may present a situation in which already deployed agents of the MAS-PL would violate the new dependencies. This possibility, however, is unlikely since newly introduced constraints should relate to the new features (i.e., roles and variations points) included into the MAS-PL rather than to existing features. DECIMAL has the capability to check if already deployed agents would violate any of the new dependencies, but it would be the responsibility of the developers to resolve the deployed violations (if necessary).

6. Evaluation and Discussion

To evaluate and discuss the Gaia-PL methodology, we briefly summarize its application to the Prospecting Asteroid Mission (PAM) case study, using Gaia-PL's handling of PAM's challenges to agent development, described in Section 3, to guide the discussion. We compare the results obtained using Gaia-PL to those obtained using the Gaia methodology [14] and offer some caveats for applications to other multi-agent system product lines (MAS-PL).

Large design space with reduced specification times. The application of the Gaia-PL methodology to the PAM case study accommodated the requirements that allowed building 160 unique spacecraft. Approximately one-third of the PAM requirements documented in DECIMAL were common to every spacecraft regardless of its specialized role (*leader*, *messenger* or *worker*). The variability requirements were analyzed and grouped into 48 parameters of variation and 47 features which led to identifying 13 roles and 39 variation points using the heuristics provided in Section 4.2. The documentation of the requirements and requirements specifications for the 39 variation points using the Role Variation Point Schemas took approximately 30 minutes each for a total of 19.5 hours. Thus, for each requirement implemented in a Role Variation Point Schema, it was found in this case study that an average of 7.3 minutes was needed to document the requirement's detailed specification.

High level of reuse in high-variability systems with traceability. The ability to capture the common requirements of a role in a variation point avoids the need to have the common requirements repeated in several role schemas for each of the variation points. Application of the Gaia methodology to the PAM case study increased the number of schemas needed by approximately 19% compared to our Gaia-PL approach. More importantly, the number of redundantly implemented requirements illustrates the advantage of Gaia-PL. In the PAM case study 147 requirements (approximately 66%) specified in the requirements specifications were redundant since they needed to be documented for each variation point to create a new role in Gaia. Assuming an average of 7.3 minutes per requirement to document a requirements specification, as found in our previous application of Gaia-PL, Gaia incurred an *additional* 17.8 hours to derive and document compared to our approach.

Hierarchical and changing roles with Gaia-PL. Gaia-PL clearly documents an agent's ability to change from one set of functionalities of a role to another, while Gaia has to combine the functionality from the variation points into a single role. Gaia-PL, on the other hand, keeps the modularity of the differing types of functionality in a role. In addition, Gaia-PL provides linking relationships between related roles, as described in Sections 4.2 and 4.3, unlike the non-hierarchical nature of Gaia. Gaia-PL also avoids unnecessary repetition of functionality.

Tool-supported constraint verification. The DECIMAL tool documented and managed both the requirements and the dependencies. In PAM there were 124 such dependencies to be verified and maintained. Automated tool support ensured that the variation points of the agents continued to satisfy all relevant constraints in an efficient manner.

Implications for other MAS-PLs. The evaluation of our Gaia-PL methodology led to a few caveats. First, the PAM case study had requirements that fit nicely into a software product-line engineering approach (i.e., the variability requirements of the PAM mission partly focused on the differing functionality of the different types of spacecraft). Second, there was an approximately 2:1 ratio of variable requirements to commonality requirements. These factors contributed to the clear reuse advantage of Gaia-PL. Finally, as in any software design, the expertise of the user may impact the overall reuse potential of the software engineering assets of a MAS-PL. Despite these caveats, this evaluation indicates Gaia-PL's advantages in designing, developing and documenting a MAS-PL that has some degree of variability. Gaia-PL's ability to hierarchically define the roles of an agent, capture the common and variable functionality of an agent and reuse the common functionality of a role to design and develop a wide-range of agents of the MAS-PL recommends its use. In particular, the use of Gaia-PL as an extension of Gaia allows the software developer to take advantage of the reuse potential in Gaia-PL along with the other models, abstractions and analysis tools of Gaia to provide the mechanisms to efficiently build a MAS-PL.

7. Concluding Remarks

In this paper we illustrated the specification and reuse of multi-agent system product line (MAS-PL) requirements. We illustrated how the use of a MAS-PL requirements specification supported by the DECIMAL tool can provide management and verification of the MAS-PL requirements for new agents. A feature model constructed from the MAS-PL requirements was used to identify the roles and variation points of the MAS-PL and to guide the hierarchical design of the requirements specification schemas. Finally, we described an evaluation of our approach on a proposed NASA spacecraft and provided a discussion of its reuse advantages and caveats.

In terms of future work, the Gaia-PL methodology described in this paper focuses on the development and reuse of requirements and requirements specifications of MAS-PL. Peña et al. have described the development of a core architecture for MAS-PL for reuse in future systems [8]. Thus, a natural avenue of future work could include marrying the two approaches to produce a comprehensive MAS-PL methodology to facilitate reuse from requirements to architecture during initial system development, maintenance and MAS-PL evolution. Further, we plan to investigate how agent UML or other modeling and/or architectural languages can be used to further facilitate asset reuse for MAS-PL.

8. Acknowledgements

This research was supported by National Science Foundation grants 0204139, 0541163, and 0702758. The first author was additionally supported by the Charles L. Brown Department of Electrical and Computer Engineering at the University of Virginia.

9. References

[1] J. Dehlinger. "Incorporating Product-Line Engineering Techniques into Agent-Oriented Software Engineering for Efficiently Building Safety Critical Multi-Agent Systems", *Ph.D. Thesis*, Iowa State University, 2007.

[2] J. Dehlinger and R. R. Lutz. "A Product-Line Approach to Promote Asset Reuse in Multi-Agent Systems", In *Software Engineering for Multi-Agent Systems IV*, LNCS 3914, pp. 161-178, 2006.

[3] J. Dehlinger R. R. Lutz. "A Product-Line Approach to Safe Reuse in Multi-Agent Systems", In *Proc. 4th Int'l Workshop on Software Engineering Large-Scale Multi-Agent Systems*, St. Louis, MO, pp. 83-89, 2005.

[4] R. Girardi. "Reuse in Agent-based Application Development", *Proc. Int'l Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, Orlando, FL, 2002.

[5] H. Hara, S. Fujita and K. Sugawara. "Reusable Software Components Based on an Agent Model", In *Proc. Workshop on Parallel and Distributed Systems*, 2000.

[6] T. Juan and L. Sterling. "ROADMAP: Extending the Gaia Methodology for Complex Open Systems", In *Proc. 1st Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, pp. 3-10, 2003.

[7] P. Padmanabhan and R. R. Lutz. "Tool-Supported Verification of Product Line Requirements", In *Automated Software Engineering Journal*, 12(4):447-465, 2005.

[8] J. Peña, M. G. Hinchey, A. Ruiz-Cortes and P. Trinidad. "Building the Core Architecture of a NASA Multiagent System Product Line", In *Proc. 7th Int'l ACM Workshop on Agent Oriented Software Engineering*, Hakodate, Japan, pp. 13-24, 2006.

[9] K. Pohl, G. Bockle and F. van der Linden. *Software Product-Line Engineering*, Springer-Verlag, 2005.

[10] I. Nunes, I., C. Nunes, C., U. Kulesza and C. Lucena. "Developing and Evolving Multi-Agent System Product Lines", In *9th Int'l Workshop Agent-Oriented Software Engineering*, Estoril, Portugal, 2008.

[11] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff. "Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions", In *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, 36(3):279-291, 2006.

[12] A. Tveit. "A Survey of Agent-Oriented Software Engineering", *NTNU Computer Science Graduate Student Conf.*, 2001.

[13] D. M. Weiss and C. T. R. Lai. *Software Product-Line Engineering*, Addison-Wesley, Reading, MA, 1999.

[14] F. Zambonelli, N. R. Jennings and M. Wooldridge. "Developing Multiagent Systems: The Gaia Methodology", In *ACM Transactions on Software Engineering and Methodology*, 12(3):317-370, 2003.