Journal of Software Systems and Modeling manuscript No.
(will be inserted by the editor)

Modeling NASA Swarm-Based Systems

Using Agent-Oriented Software Engineering and Formal Methods

Joaquin Peña · Christopher A. Rouff · Mike Hinchey · Antonio Ruiz-Cortés

Received: date / Revised version: date

Abstract The need to collect new data and perform new science is causing the complexity of NASA missions to continually increase. This complexity needs to be controlled via new technological advancements and balanced with a reduction in mission and operation costs. Planned and hypothesized missions involve self-management, biological-inspiration based on swarms, and autonomous operation as a means of achieving these goals. We consider a tailored software engineering approach to developing such systems based on agent-oriented software engineering and formal methods. We report on the advances in modeling, implementing, and testing NASA swarm-based concept missions.

Keywords Swarms · Emergent Behavior · Agent Oriented Software Engineering · Formal Methods

The work reported in this article was supported by the Spanish Ministry of Science and Technology under grants TIC2003-02737-C02-01 and TIN2006-00472, by the NASA Software Engineering Laboratory, NASA Goddard Space Flight Center, Greenbelt, MD, USA, and by the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) and by Science Foundation Ireland grant number 03/CE2/I303-1 to Lero—the Irish Software Engineering Research Centre.

J. Peña
University of Seville, Seville, Spain
E-mail: joaquinp@us.es

C. Rouff
Lockheed Martin Corporation, Advanced Technology Laboratories
Arlington, VA, USA
E-mail: christopher.rouff@lmco.com

M. Hinchey
Lero—the Irish Software Engineering Research Centre
University of Limerick, Ireland
E-mail: mike.hinchey@lero.ie

A. Ruiz-Cortés
University of Seville, Seville, Spain
E-mail: aruiz@us.es

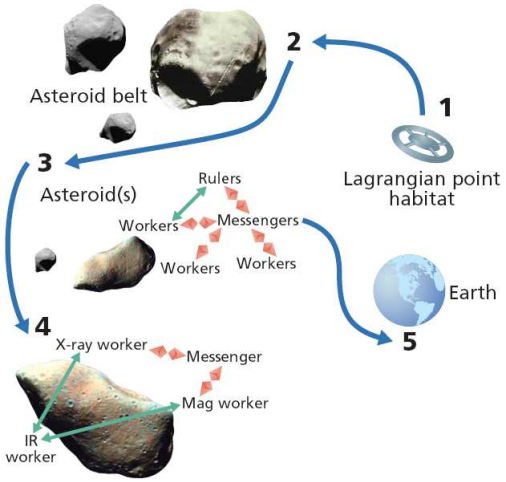

Fig. 1 ANTS Mission Concept

1 Introduction

NASA is investigating new concepts for future space exploration to conduct new science and reduce costs. Potential new mission concepts include the use of multiple spacecraft, unmanned autonomous vehicles (UAVs) and armies of robots, operating like swarms in nature [3], to explore harsh remote environments. Such missions will be necessarily unmanned and highly autonomous, exhibiting the properties of autonomic systems, being self-protecting, self-healing, self-configuring, and self-optimizing [17].

Swarms [1,5] consist of a large number of simple agents that have local interactions (between each other and the environment). There is no central controller directing the swarm and no one agent has a global view; they are self-organizing based on the emergent behaviors of the simple interactions. This kind of system has many advantages since complex behaviors emerge from the definition of simple individual behaviors, thus reducing the design effort. Its unpredictability, however, represents a problem since unexpected behaviors may cause the failure of a mission.

Formal methods can provide guidance in determining possible emergent behaviors that must be considered. They may be used to propose certain properties that might or might not hold, or certain emergent behaviors that may arise [15]. Verifying emergent behavior has been little addressed by the formal methods community, though there has been work by computer scientists on analyzing biological systems [7,16].

Complexity is a particular problem in developing such swarm-based mission. To address this, we have been investigating an approach based on formal methods and Agent-Oriented Software Engineering (AOSE).

2 Case study

The Autonomous Nano-Technology Swarm (ANTS) concept mission [3,4], illustrated in Figure 1 [17], involves the launch of a swarm of autonomous pico-class (approx. 1kg) spacecraft to explore the asteroid belt for asteroids with certain scientific characteristics. A transport ship travels to a Lagrangian point where gravitational forces on small

objects are negligible. From here, 1000 spacecraft, manufactured en route, are launched into the asteroid belt. Approximately 80 percent of the spacecraft will be workers that, due to their small size and power limitations, will carry a single specialized instrument (such as a magnetometer) and will collect a specific type of data.

Crucial to the mission is the ability to modify operations to reflect the changing nature of the mission, and to operate autonomously due to the distance from Earth and low-bandwidth communications.

The swarm will form sub-swarms, under the control of a ruler, which contains models of the types of science that are to be pursued. The ruler will coordinate workers to collect data on specific asteroids and will determine which asteroids to examine further. If the data matches a profile of interest, an imaging spacecraft will be sent to the asteroid to ascertain the exact location and to create a rough model to be used by other spacecraft for maneuvering around the asteroid. Other teams of spacecraft will then coordinate to finish mapping the asteroid to form a complete model.

3 MaCMAS

MaCMAS is the AOSE methodology that we use for modeling swarm-based systems and is based on previously developed concepts [10][1]. It is specially tailored to model complex Multiagent Systems covering all the requirements for tackling complexity shown previously and to the best of our knowledge is unique in that it covers all of these principles.

MaCMAS follows an organizational metaphor. It is based on multi-agent systems (MASs) mimicking human organizations, which can be also applied to swarm-based structures.

The organizational metaphor has been proven to be one of the most appropriate tools for engineering these kinds of systems, and has been successfully applied by other methodologies, e.g., [9,18]. It shows that a MAS/swarm organization can be observed from two viewpoints [18]: an *acquaintance* point of view, which shows the organization as the set of interaction relationships between the roles played by agents, and a *structural* point of view, which shows agents as artifacts that belong to sub-organizations, groups and teams. In the latter view agents are also structured into hierarchical structures showing the social structure of the system.

Since any structural organization must include interactions between agents in order to function, it is safe to say that the acquaintance organization is always contained in the structural organization. Therefore, a natural map is formed between the acquaintance organization and the corresponding structural organization. This is the process of assigning roles to agents [18]. Then, we can conclude that any acquaintance organization can be modeled orthogonally to its structural organization [6].

This concept has been applied to the models we developed for the NASA ANTS concept mission, where each spacecraft is assigned a set of roles that change over time.

3.1 Modeling with McCMAS

MaCMAS focuses on the acquaintance organization view providing a set of UML2.0-based models, a software process to build them, and a set of techniques to compose,

[1] See http://james.eii.us.es/MaCMAS/ for details and case studies.

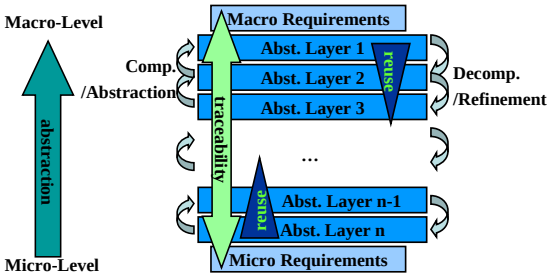

Fig. 2 Overview of the structure of MaCMAS models

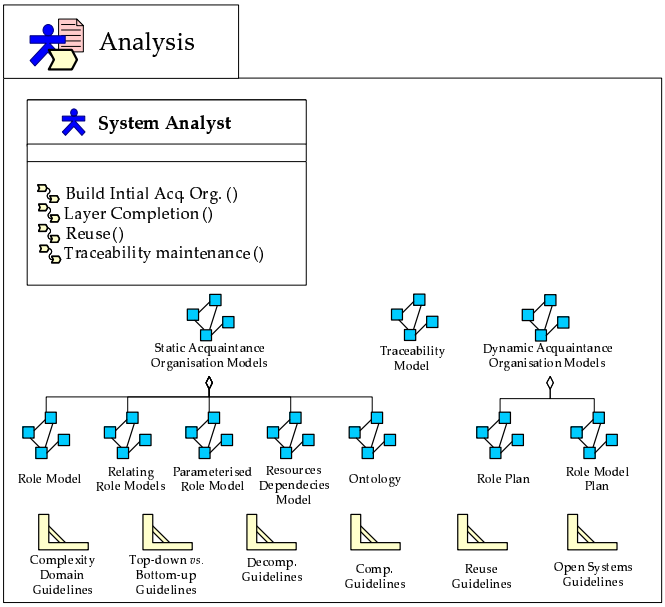

Fig. 3 Acquaintance analysis discipline

decompose, refine and abstract models, as required by the above principles to address complexity. These models are not orthogonal to formal methods, but complementary since they provide a graphical representation of the system that improves the understanding of the formal specifications and the formal methods provide a strong mathematical support for MaCMAS.

Figure 2 summarizes the structure of models produced by MaCMAS and how they are obtained. Roughly speaking, MaCMAS produces a set of models of the system at different levels of abstraction in order to tackle the complexity of a system iteratively. Thus, a model of the system at the micro-level, where all details are modeled, can be linked with a model of the system at the macro-level where only relevant properties are modeled using abstraction. This allows us to ensure properties at the micro, macro, and intermediate levels of the system by means of formal methods. In addition to this structure of models, as shown in the following, MaCMAS also provides techniques to refine and abstract these models to complete the layers.

Figure 3 summarizes all the models, activities and guidelines included in MaCMAS.

The most important are:

a) Static Acquaintance Organization Models: These show the static interaction relationships between roles in the system and the knowledge processed by them. In this category, we find models for representing the ontology managed by agents, models for representing their dependencies, and role models. The latter are the most important; they show an acquaintance sub-organization as a set of roles collaborating by means of several *multi-Role Interactions* (mRI) [11]. mRIs are used to abstract the acquaintance relationships amongst roles in the system. As mRIs allow abstract representation of interactions, we can use these models at whatever level of abstraction we desire. This allows an abstract representation of complex processes that implies the use of AI techniques such as negotiation, learning or self-* techniques, allowing the modeling of emergent features by means of abstraction.
b) Dynamic Acquaintance Organization Models: The behavioral aspect of an organization shows the sequencing of mRIs in a particular role model. It is represented by two equivalent models:
   Role plan: represents the plan of each role which is the sequence of roles in the MRIs. The role plan is represented using UML 2.0 ProtocolStateMachines [8]. It is used to focus on a certain role, while ignoring others.
   Role model plan: represents the order of mRIs in a role model with a centralized description. It is also represented using UML 2.0 StateMachines. It is used to facilitate the understanding of the whole behavior of a sub-organization.
c) Traceability Model: This model shows the relationships of models in different abstraction layers. It shows how mRIs are abstracted, composed or decomposed by means of *classification*, *aggregation*, *generalization* or *redefinition*. Note that we usually show only the relationships between interactions because they are the focus of modeling, but all the elements that compose an mRI can also be related. Since an mRI presents a direct correlation with system goals, traceability models clearly show how a certain system requirement is refined and materialized at a level of abstraction, from micro to macro-level.

3.2 Modeling autonomous and autonomic systems using MaCMAS

MaCMAS can be used to model autonomous and autonomic properties of swarm-based systems. Figure 4 shows the traceability diagram obtained after applying all the software processes of MaCMAS to the ANTS concept mission. This diagram summarizes the mRIs in the system structured by layers of abstraction. In this diagram, the top layer is the most abstract, i.e. the macro-level. As each node represents a system-goal, we can see the division of tasks undertaken to develop the system. As each mRI is inside a role model, we can also see which roles have been determined by observing the role models. In the model shown, we have depicted several sub-regions. Horizontal subdivisions depict layers of abstraction, while the vertical line denotes the distinction between the autonomic properties of the system and the autonomous properties. In addition to mRIs, MaCMAS also uses UML packages to represent role models that contain several mRIs, as seen in Figure 4.

To foster reuse, to model an autonomous or an autonomic property in a sufficiently generic and generalized way, and to enable the deploying of these features at runtime, properties must be independent of the concrete agents over which they will be deployed.

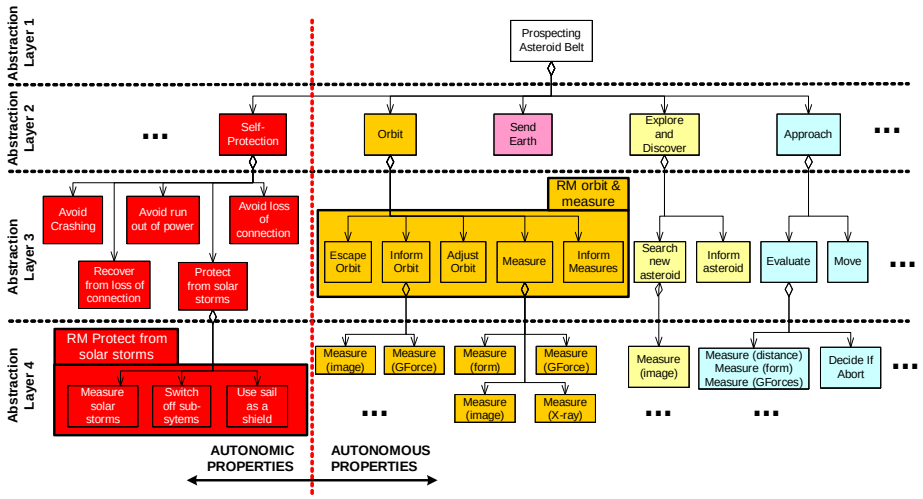

Fig. 4 Traceability model of ANTS

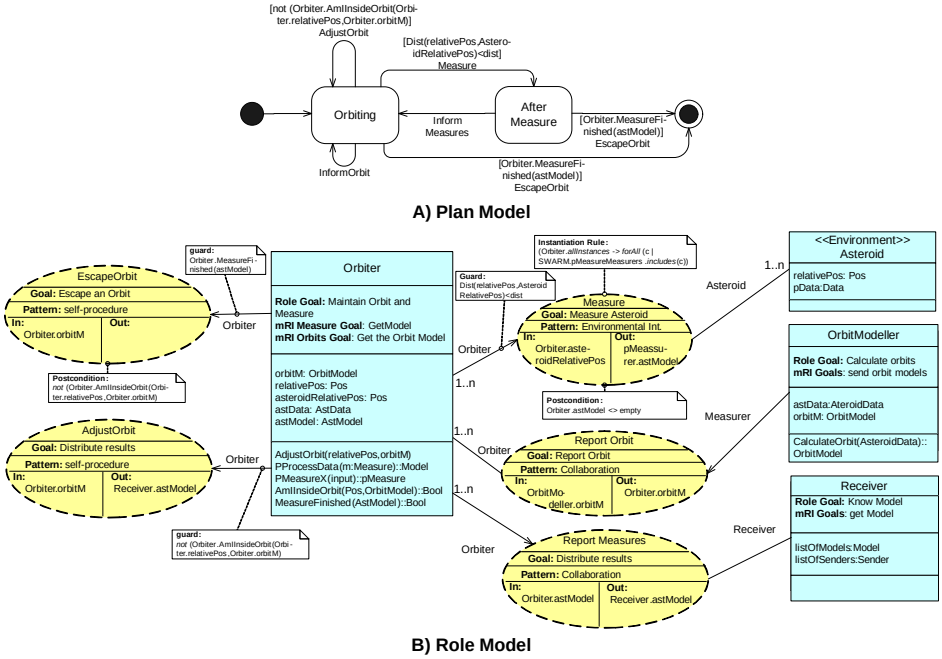

Fig. 5 Orbiting and measuring an asteroid autonomous property

As discussed, to represent this kind of organization, MaCMAS proposes two kinds of models—one for showing the relationships between roles, that is, role models, and another to show how these relationships evolve over time, that is to say, plan models.

For example, showing the autonomous process of orbiting an asteroid to take a measurement requires at least two models—its role model and its plan model. Note that the plan model can be formaly specified allowing us to check for various properties

or conditions such as deadlocks, etc. Figure 5b shows the role model from the third layer of abstraction of Figure 4. In this model there are two kinds of elements: roles, which are represented using interface-like icons, and mRIs, which are represented as collaboration-like icons. The roles show the general and particular goals when participating in an interaction with other roles or with the environment (represented using interfaces with the <<environment>> stereotype). Roles also represent the knowledge they manage (middle compartment) and the services they offer (bottom compartment). For example, the goal of the *Orbiter* role is "maintain the orbit and measure [the asteroid]", while its goal when participating in the *Report Orbit* interaction is to obtain a model of the orbit it must follow. In addition to roles, mRIs also show us some important information. They must also show the system-goal they achieve when executed, the kind of coordination that is carried out when executed, the knowledge used as input to achieve the goal, and the knowledge produced. For example, the goal of the mRI *Report Orbit* is to "Report the Orbit". It is done by taking as input the knowledge of the *OrbitModeler* regarding the orbit and producing as output the model for the orbit (orbitM) in the *Orbiter* role.

In Figure 5a, we show the plan model of this role model where the order of execution of all its mRIs is shown. As can be seen, the *Orbiter*, while it is in orbit, is adjusting its orbit and and reporting measurements. When it has completed constructing a model of the asteroid, it escapes the orbit using its knowledge of the orbit model (*orbitM*).

Autonomic properties can be also modeled in this way. As role models can be used at any level of abstraction, we can use them for specifying autonomic properties that concern a single agent, or even a group of agents when dealing with autonomic properties at the swarm level. Thus, as shown in the traceability model, we have a role model at abstraction layer 2 that shows the swarm autonomic behavior, while at layer 4, we show autonomic properties at the level of individual spacecraft.

Figure 6 illustrates a model at abstraction layer 4 for a self-protection autonomic property for protecting against solar storms. The role model for this property is shown in Figure 6b, and since it is a property at the individual level, a single role is shown (*SelfProtectSpaceCraft*). Its plan model is shown in Figure 6a. As all the spacecraft can be affected by solar storms, this role is applied to all the spacecraft in the swarm.

3.3 MaCMAS for modelling Multiagent Systems Product Lines (MAS-PL)

Many organizations, and software companies in particular, develop a range of products over periods of time that exhibit many of the same properties and features. The multiagent systems community exhibits similar trends. However, the community has not as yet developed the infrastructure to develop a core multiagent system from which concrete (substantially similar) products can be derived.

The software product line (SPL) paradigm augurs the potential of developing a set of core assets for a family of products from which customized products can be rapidly generated. This reduces time-to-market, costs, etc. [2], while simultaneously improving quality by making the design, implementation and testing more financially viable by amortizing it over several products. The feasibility of building MASs product lines is presented in [14]. All NASA swarm-based missions present many common features, thus it is feasible to apply a MAS-PL approach improving reuse, and thus improving our capabilities to deal with its complexity. This may also dramatically reduce the cost and time to develop related missions.

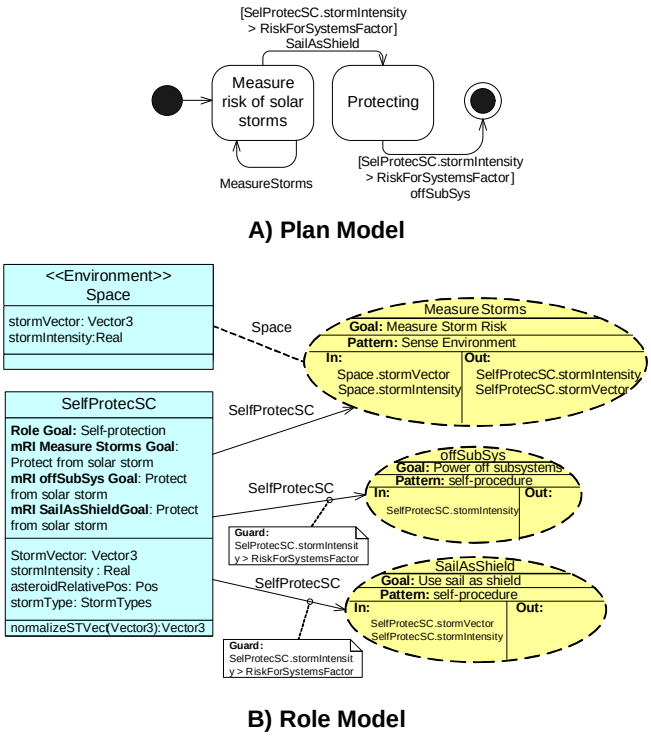

Fig. 6 Self-protection from solar storms autonomic property model

To start a product line a core architecture for the family of software products must be identified. Figure 7 shows the SPEM software process to build the core architecture.

3.4 Modeling evolving systems using MaCMAS

MaCMAS is also able to model evolving systems as required by swarm-based NASA missions [12]. MaCMAS is based on viewing different instances of a system as it evolves as different "products" in a Software Product Line. That Software Product Line is in turn developed with an agent-oriented software engineering approach and views the system as a Multiagent System Product Line.

Each product in a MAS-PL is defined as a set of features. Given that all the products present a set of features that remain unchanged, the core architecture is defined as the part of all of the products that implement these common features [13]. Thus, a system can evolve by changing, or evolving, the set of non-core features.

In [13], we show that a feature correlates with a role model. Thus, for a system to evolve from one state to another, we must compose or decompose the role models.

We represent the evolution plan using a UML state machine where each state represents a product, and each transition represents the addition or elimination of a set of features. Also, the conditions in the transitions represent the properties that must hold in the environment and in the system in order to evolve to the new product.

Figure 8 shows part of the evolution plan of the ANTS concept mission. There we represent two products, one representing the swarm when orbiting an asteroid, and

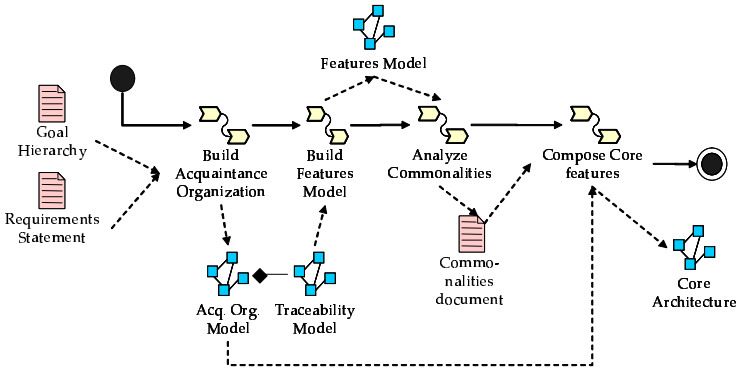

Fig. 7 Overview of the process for building the core architecture of a MAS-PL

another representing the swarm when orbiting and protecting from a solar storm. As can be seen, we add or delete the feature corresponding to "protect from solar storm" depending on whether or not the swarm is under risk of a solar storm.

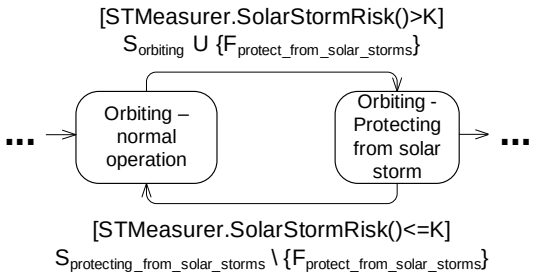

Fig. 8 Evolution plan of our case study

The main advantage of this approach resides in the fact that it allows us to derive a formal model of the system and of each state that it may reach. This allows us to clearly specify the differences from one state of the architecture and any subsequent states of that evolving system. This significantly improves our capabilities to understand, analyze and test evolving systems. Additionally, thanks to the use of MaCMAS which allows for the description of the same feature at different levels of abstraction, we can also specify and test the architectural changes at different levels of abstraction.

Finally, such an approach provides support at run time for the addition and deletion of roles in the architecture. It provides reflection mechanisms that enable understanding of the features, roles, and agents at different levels of abstraction, providing capabilities for ensuring quality of service by means of self-organization, self-protection, and other self-* properties identified by the Autonomic Computing initiative.

4 Conclusions and Future work

A means of achieving ambitions goals for conducting new science in future NASA missions is based on the use of biological inspiration from swarms in nature. When tackling any system development, and particularly something as complex and ambitious

as the NASA ANTS concept mission, complexity can quickly become unmanageable, negating the value of new technologies, such as self-management.

Our experience of modeling swarm-based systems has been that a combination of UML-based tailored models and formal methods augurs much promise. While promising results have been achieved to date, we are continuing to improve the integration of these approaches and to improve the support tools that we have been developing.

References

1. Gerardo Beni and Jing Want. Swarm intelligence. In *Proc. Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, Tokyo, Japan, 1989. RSJ Press.
2. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison–Wesley, August 2001.
3. S. A. Curtis, J. Mica, J. Nuth, G. Marr, M. L. Rilee, and M. K. Bhat. ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to Asteroid Belt resource exploration. In *Proc. Int'l Astronautical Federation, 51st Congress*, Oct. 2000.
4. Steven A. Curtis, W. F. Truszkowski, Michael L. Rilee, and Pamela E. Clark. ANTS for the human exploration and development of space. In *Proc. IEEE Aerospace Conference*, Big Sky, Montana, USA, 9–16 March 2003.
5. M. Hinchey, R. Sterritt, and C. Rouff. Swarms and swarm intelligence. *IEEE Computer*, 40(4):111–113, April 2007.
6. E. A. Kendall. Role modeling for agent system analysis, design, and implementation. *IEEE Concurrency*, 8(2):34–41, April/June 2000.
7. W. Michael and L. Holcombe. Mathematical models of cell biochemistry. Technical Report CS-86-4, Sheffield University, UK, 1986.
8. Object Management Group (OMG). Unified modeling language: Superstructure. version 2.0. Final adopted specification ptc/03–08–02, OMG, August 2003. www.omg.org.
9. H. Van Dyke Parunak and James Odell. Representing social structures in UML. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the 5^{th} Int. Conf. on Autonomous Agents*, pages 100–101, Montreal, Canada, 2001. ACM Press.
10. J. Peña. *On Improving The Modelling Of Complex Acquaintance Organisations Of Agents. A Method Fragment For The Analysis Phase*. PhD thesis, University of Seville, 2005.
11. J. Peña, R. Corchuelo, and J. L. Arjona. A top down approach for mas protocol descriptions. In *ACM Symposium on Applied Computing SAC'03*, pages 45–49, Melbourne, Florida, USA, 2003. ACM Press.
12. J. Peña, M. G. Hinchey, M. Resinas, R. Sterritt, and J. L. Rash. Managing the evolution of an enterprise architecture using a MAS-product-line approach. In *5th International Workshop on System/Software Architectures (IWSSA'06)*, page to be published, Nevada, USA, 2006. CSREA Press.
13. J. Peña, M. G. Hinchey, and A. Ruiz-Cortés. Building the core architecture of a NASA multiagent system product line. In *7th International Workshop on Agent Oriented Software Engineering 2006*, page to be published, Hakodate, Japan, May, 2006. LNCS.
14. J. Peña, M. G. Hinchey, and Antonio Ruíz-Cortes. Multiagent system product lines: Challenges and benefits. *Communications of the ACM*, December 2006.
15. Christopher A. Rouff, Michael G. Hinchey, Walter F. Truszkowski, and James L. Rash. Experiences applying formal approaches in the development of swarm-based space exploration systems. *International Journal of on software Tools for Technology Transfer. Special Issue on Formal Methods in Industry*, 8(6), 2006.
16. D. J. T. Sumpter, G. B. Blanchard, and D. S. Broomhead. Ants and agents: a process algebra approach to modelling ant colony behaviour. *Bulletin of Mathematical Biology*, 63(5):951–980, September 2001.
17. W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff. Autonomous and autonomic systems: A paradigm for future space exploration missions. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 2006.
18. F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the GAIA methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3), September 2003.