A Model-Driven Architecture Approach for Modeling, Specifying and Deploying Policies in Autonomous and Autonomic Systems

Joaquin Peña
University of Seville
Spain
joaquinp@us.es

Michael G. Hinchey
NASA Goddard Space Flight Center
USA
Michael.G.Hinchey@nasa.gov

Roy Sterritt
University of Ulster
Northern Ireland
r.sterritt@ulster.ac.uk

Antonio Ruiz-Cortés
University of Seville
aruiz@us.es

Manuel Resinas
University of Seville
resman@tdg.lsi.us.es

Abstract

Autonomic Computing (AC), self-management based on high level guidance from humans, is increasingly gaining momentum as the way forward in designing reliable systems that hide complexity and conquer IT management costs. Effectively, AC may be viewed as Policy-Based Self-Management. The Model Driven Architecture (MDA) approach focuses on building models that can be transformed into code in an automatic manner. In this paper, we look at ways to implement Policy-Based Self-Management by means of models that can be converted to code using transformations that follow the MDA philosophy. We propose a set of UML-based models to specify autonomic and autonomous features along with the necessary procedures, based on modification and composition of models, to deploy a policy as an executing system.

1 Introduction and Motivation

Autonomic Systems (encompassing both Autonomic Computing and Autonomic Communications) is an emerging field for the development of large-scale, self-managing, complex distributed computer-based systems [1]. In introducing the concept of Autonomic Computing, IBM's Paul Horn likened the needs of large scale systems management to that of the human Autonomic Nervous System (ANS). The ANS, through self-regulation, is able to effectively monitor, control and regulate the human body without the need for conscious thought [5].

As in all emerging fields, there are many fruitful areas for concern, that are worthwhile targets for research and development. Many issues are yet to be addressed, such as, for example, how autonomic managers, which together with the component being managed make up an autonomic element, should be designed in order to exist in a collaborative autonomic environment, and ultimately provide self-management of the system to the highest degree possible.

The long-term strategic vision of AC highlights an overarching self-managing vision where the system would have such a level of "self" capability that a senior (human) manager in an organization could specify business policies—such as profit margin on a specific product range, or system quality-of-service for a band of customers—and the computing and communications systems would do the rest themselves.

The main idea behind a Model-Driven Architecture is separation of the specification of the operation of a system from the details of the way that system uses the capabilities of its platform. With the purpose of abstracting away platform details, MDA involves two main types of models. The platform-independent model (PIM) which provides a model of the system without platform details, and the platform specific-model (PSM), which is obtained by means of transformation of the PIM model.

We propose an MDA approach for applying policies to autonomic systems. This avoids platform-dependent details unnecessary at the level of abstraction of a policy, and employs transformations of models to bring the policy through the necessary levels to be transformed into an implementation [8]. This is based on our previous work applying an Agent-Oriented methodology called MaCMAS (Methodology Fragment for Analyzing Complex Multi-Agent Systems) to model, specify and deploy policies at runtime [15]. In this paper, we extend this work to, using an MDA approach, automate the process of applying a new policy. Essentially, we propose UML-based PIMs for specifying autonomous and autonomic properties of the system and an

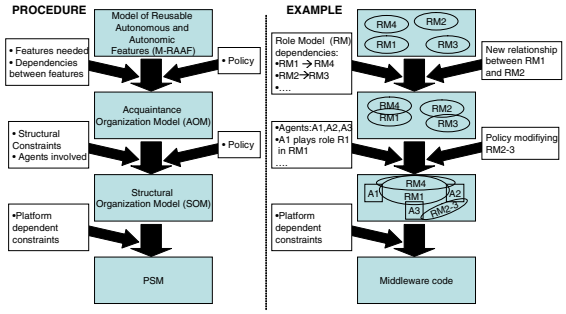

Figure 1. Summary of our MDA approach

operation to transform these models in order to implement changes specified by a policy in the running system.

In addition, to illustrate our approach, we use an example from the NASA ANTS concept mission (described in Section 3). This mission involves the use of a swarm of pico-class spacecraft to explore and collect data from the asteroid belt, and exhibits both autonomous and autonomic properties.

2 Using MDA for applying policies

Figure 1, depicts the main steps and models of our approach. We propose using three PIM models and one PSM model. These models are as follows:

Reusable Autonomous & Autonomic Features (M-RAAF): The first model we produce is a platform independent model where each autonomous or autonomic feature is modeled in isolation and without platform-dependent details. This first model, that we call *Model of Reusable Autonomous and Autonomic Features* (M-RAAF), allows us to improve our capabilities for reusing features across systems since we can maintain a repository of models [3]. Each feature is modeled separately by means of a role model, as is shown in Figure 1.

Acquaintance Organization Model (AOM): The AOM shows the organization of agents as the set of interaction relationships between the various roles played by agents. Role models are also used to represent this model, but, as shown in the figure, in this stage, role models do not represent isolated features, but are composed to show how these features relate in the system at hand.

Structural Organization Model (SOM): The SOM shows agents as artifacts that belong to sub-organizations, groups, and teams. In this view agents are also structured into hierarchies showing the social structure of the system. In this model, agents are assigned a set of roles.

Platform Specific Model (PSM): Finally, a platform specific model is used to deploy the policies over the running system.

Note that the distinction between the Acquaintance Organization Model (AOM) and the Structural Organization Model (SOM) is usual in Agent-Oriented Software Engineering, e.g. in the GAIA methodology [17].

As can also be observed in the figure, we use transformations between these models. These transformations are the following:

Transformation from M-RAAF to AOM: This transformation consists of taking such features as are needed for our system, and composing those that are dependent. For this we use information on the dependencies between these features, and the process is carried out by role model composition. In addition, given that in this schema each feature is separate and we know their dependencies, as shown in the figure, when a new policy has to be deployed into the system, we can analyze it to find out which features are affected. Then, adding the new dependencies introduced by the policy, we can modify the features and transform their models, to ultimately result in the PSM that will be deployed over the running system.

Transformation from AOM to SOM: This transformation consists of taking the set of agents needed in our system, and the structural constraints of the organization they must adopt, and assign to each agent the role it must play. This process is done by role model composition also. The structural organization is formed by agents playing roles. These agents must use interactions in order to function, and thus, it is safe to say that the acquaintance organization is always contained in the structural organization [6]. This allows us to define a natural mapping between the acquaintance organization and the corresponding structural organization. The mapping consists of assigning roles to agents and is the one used for this transformation from AOM to SOM[17].

Transformation from SOM to PSM: This is done by adding platform specific details to the SOM, such as generating code taking into account type conversions, mechanisms of assigning roles to running agents, and so on.

When a new policy is to be applied we can do so at the M-RAAF level or at the AOM level. In the former case, we can specify policies that add new features, or that establish new dependencies between existing features or new

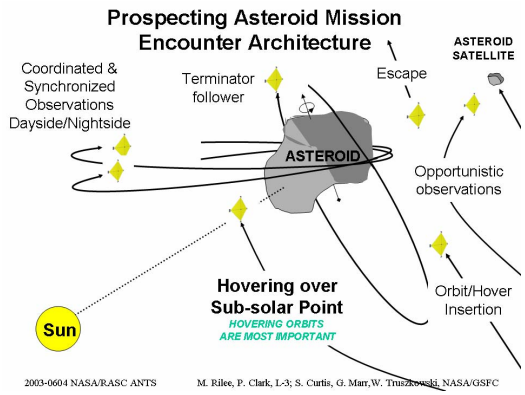

Figure 2. ANTS encounter with an asteroid

features. Once the features affected by a policy have been identified and we have applied the changes to the M-RAAF, we can apply the transformations to propagate the changes through to the running system. In the latter case, we can specify policies that change the roles played by agents or their organization (that is to say, the structural organization). In this case, changes can be also propagated by means of transformations.

3 NASA ANTS Case Study

In this section, we briefly introduce ANTS, a NASA concept mission, that illustrates properties of several potential exploration missions. We show two models of an autonomous and autonomic property of the system.

3.1 ANTS Mission Overview

The Autonomous Nano-Technology Swarm (ANTS) mission [2, 16] is a concept mission that involves the use of swarms of autonomous pico-class (approximately 1kg) spacecraft that will search the asteroid belt for asteroids that have specific characteristics. The mission is envisioned to consist of approximately 1,000 spacecraft launched from a factory ship. As illustrated in Figure 2, the swarm is envisioned to consist of several types of spacecraft. Many of these spacecraft (called specialists) will have a specialized single instrument for collecting particular types of data. To examine an asteroid, several spacecraft will have to form a sub-swarm, under the control of a ruler, and collaborate to collect data from asteroids of interest, based on the properties of that asteroid. This will be achieved using an insect analogy of hierarchical social behavior with some spacecraft directing others.

3.2 Autonomic Properties of ANTS

The ANTS system may be viewed as an Autonomic System as it meets four key requirements: self-configuration, self-healing, self-optimization and self-protection, as illustrated in [16]. Here we focus on self-configuration properties as these are illustrated in our case study.

ANTS is self-protecting: The self protecting behavior of the team will be interrelated with the self-protecting behavior of the individual members. The anticipated sources of threats to ANTS individuals (and consequently to the team itself) will be collisions and solar storms.

Collision avoidance through maneuvering will be limited because ANTS individuals will have limited ability to adjust their orbits and trajectories, due to limited thrust for positioning. Individuals will have the capability of coordinating their orbits and trajectories with other individuals to avoid collisions with them. Given the chaotic environment of the asteroid belt and the highly dynamic trajectories of the objects in it, occasional near approaches of interloping asteroidal bodies (even small ones) to the ANTS team may present threats of collisions with its individuals. Collision-avoidance maneuvering for this type of spacecraft presents a great challenge and is currently under consideration. The main self-protection mechanism for collision avoidance is achieved through the process of planning. The plans involve constraints that will result in acceptable risks of collisions between individuals when they carry out their observational goals. In this way, ANTS exhibits a kind of self-protection behavior against collisions.

Another possible ANTS self-protection mechanism involves protection against the effects of solar storms, which is the basis of the case study we use later in this paper. Charged particles from solar storms could subject individuals to degradation of sensors and electronic components. The increased solar wind from solar storms could also affect the orbits and trajectories of the ANTS individuals and thereby could jeopardize the mission. Specific mechanisms to protect ANTS spacecraft against the effects of solar storms have not yet been determined. A potential mechanism might, for example, provide spacecrafts with a solar storm sensing capability through on-board, direct observation of the solar disk. When the spacecrafts recognize that a solar storm threat exists, they would invoke their goal of protecting themselves from the harmful effects of a solar storm. Part of the protective response might be to orient solar panels and sails to minimize the impact of the solar wind. An additional response might be to power down unnecessary subsystems to minimize disruptions and damage from charged particles.

4 Proposed Models

The first two PIM models, i.e., M-RAAF and AOM, are specified using the role model concept. We use an extension of UML2.0 collaborations [14]. Although a larger number of models are necessary specify the M-RAAF and the AOM (cf. [14] for further details), in the following we present only the more important for the purposes of this paper:

a) Static View: This shows the interaction relationships between roles in the system. For our purposes, the main models in this are:

Role Models: show an acquaintance sub-organization as a set of roles interacting by means of several *multi-Role Interactions* (mRI) [12, 13]. An mRI is an *institutionalized pattern of interaction* that abstractly represents the fulfillment of a system goal without detailing how this is achieved. Thus, using mRI as the minimum modeling element for interactions we do not have to take into account all of the details required to fulfill a complex system goal nor the messages that are exchanged at stages where these details have not been identified clearly, are not known, or are not even necessary. This allows us to have abstract models where intelligent behavior is carried out by means of neural networks, fuzzy logic, etc., (as, for example, is required in ANTS, cf. Section 3), without the necessity of dealing with all the details. In addition, the direct correlation between system goals and mRIs allows us to establish a clear traceability between goal-oriented requirement documents and analysis models. This is also important for our goal in this paper, since policies usually address system goals. Having this kind of model helps to simplify the way in which policies are specified, and deployed in the system at runtime. We use role models to represent autonomous and autonomic properties of the system at the level of abstraction we need.

Parameterized Role Models: are role models where some of the elements are parameterized and can be instantiated to obtain a particular role model. This model allows us to generalize the specification of features at the level of the M-RAAF model, improving its reusability.

b) Behavioral View: The behavioral view shows the sequencing of mRIs in a particular role model. It is represented by two equivalent models:

Plan of a role: This separately represents the plan of each role in a role model showing how the mRIs of the role are sequenced. It is represented using UML 2.0 ProtocolStateMachines [11, p. 422], and is used to focus on a particular role, while ignoring others.

Plan of a role model: represents the order of mRIs in a role model with a centralized description. It is represented using UML 2.0 StateMachines [11, p. 446]. It is used to facilitate easy understanding of the whole behavior of a sub-organization.

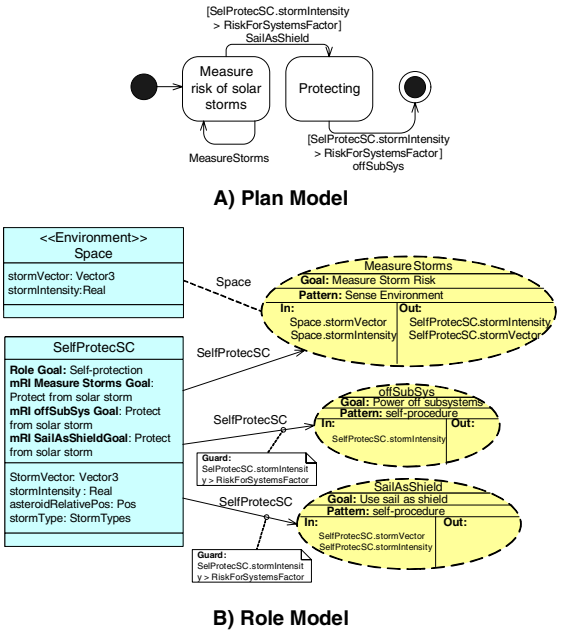

Figure 3. Model of autonomic property of self-protection from solar storms

4.1 Example of a model of reusable autonomous and autonomic features of ANTS

To foster reuse, to model an autonomous or an autonomic property in a sufficiently generic and generalized way, and to enable a policy to be deployed at runtime, features at the M-RAAF must be modeled independently of the other features they may be related to and of to the concrete agents over which they will be deployed.

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. Figure 4b shows the role model for this example. In this model there are two kinds of elements: roles, which are represented using interface-like icons, and mRIs, which are represented

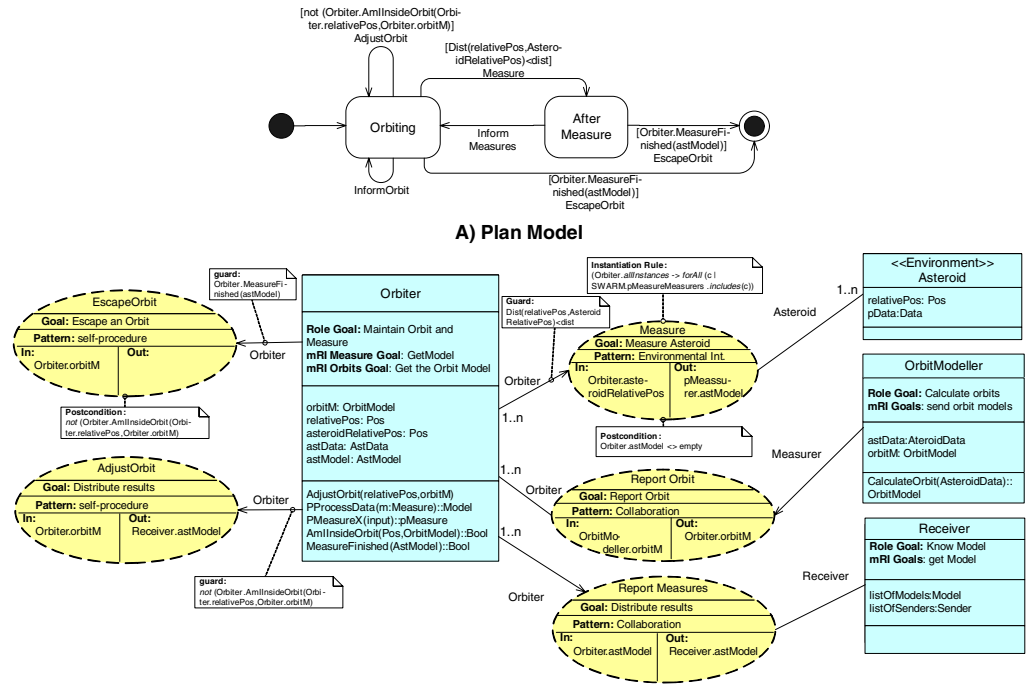

Figure 4. Autonomous property of orbiting and measuring an asteroid

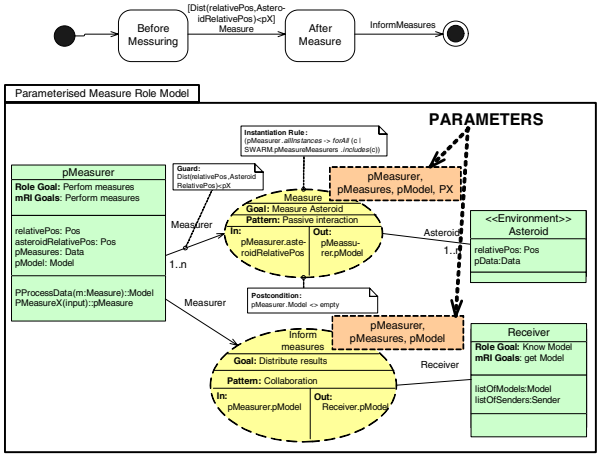

Figure 5. Parameterized role model of *Measure*

as collaboration-like icons. Roles indicate which is their general goal and their particular goals when participating in a certain interaction with other roles or with some part of 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 maintain 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.

Continuing with the example, in Figure 4a, 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 measuring and reporting measures. And 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. Here we illustrate a model for a self-protection autonomic property: protecting from solar storms. The role model for this property is shown in Figure 3b, and, as can be seen, as it is a property at the individual level, a single role is shown (*SelfProtectSpaceCraft*). Its plan model is shown in Fig-

ure 3a. As all the spacecraft can be affected by solar storms, this role will be applied to all the spacecraft in the swarm when transforming into the SOM, thereby adding this autonomic property to all of the spacecraft.

In addition, the feature for measuring may have been modeled in isolatation, as shown in Figure 5 using a parameterized role model. However, in the model used for our example, we decided not to isolate this feature.

5 Transforming from RAAF to AO

As shown previously, we must compose role models in order to build the AOM that contains the features needed from the M-RAAF. We have to take into account that when composing several role models, we can find:

emergent roles or mRIs : roles and mRIs that appear in the composition yet they do not belong to any of the initial role models;

composed roles or mRIs : the roles and mRIs in the resultant models that represent several initial roles as a single element;

unchanged roles or mRIs : those that are left unchanged and imported directly from the initial role models;

Once we have identified the roles and mRIs that have to be composed, we must complete the composite role model. Importing an mRI or a role simply requires us to add it to the composite role model; this step is trivial and we do not detail it here. The following describes how role models and plans are composed.

5.1 Composing roles

When several roles are merged in a composite role model, their elements must also be merged:

1. **Goal of the role:** The new goal of the role abstracts all the goals of the role to be composed. This information can be found in requirements hierarchical goal diagrams or we can add it as the *and* (conjunction) of the goals to be composed. In addition, the role goal for each mRI can be obtained from the goal of the initial roles for that mRI.

2. **Cardinality of the role:** THis is the same as in the initial role for the corresponding mRI.

3. **Initiator(s) role(s):** If mRI composition is not performed, as in our case, this feature does not change.

4. **Interface of a role:** All elements in the interfaces of roles to be merged must be added to the composite interface. Notice that there may be common services and knowledge in these interfaces. When this happens, they must be included only once in the composite interface, or renamed, depending on the composition of their ontologies, as we show below.

5. **Guard of a role/mRI:** The new guards are the *and* (conjunction) of the corresponding guards in initial role models if roles composed participate in the same mRI. Otherwise, guards remain unchanged.

6. **Ontologies of an mRI:** The new ontology must cover all the terms described in all the ontologies of roles to be composed (cf. [4, 9, 10]). This procedure also shows how to deal with repeated knowledge in the interface of roles to be composed. That is to say, if as a result of ontology composition, a knowledge entity that is repeated in several roles is shown as the same element in the composed ontology, we can include it just once; if it results in different elements in the composed ontology, we must rename them.

5.2 Composing plans

The composition of plans consists of setting the order of execution of mRIs in the composite model, using the role model plan or role plans. We provide several algorithms to assist in this task: extraction of a role plan from the role model plan and vice versa, and aggregation of several role plans; see [12] for further details of these algorithms.

Because of these algorithms, we can keep both plan views consistent automatically. Several types of plan composition can be used for role plans and for role model plans:

Sequential: The plan is executed atomically in sequence with others. The final state of each state machine is superposed with the initial state of the state machine that represents the plan that must be executed, except the initial plan that maintains the initial state unchanged and the final plan that maintains the final state unchanged.

Parallel: The plan of each model is executed in parallel. It can be documented by using concurrent orthogonal regions of state machines (cf. [11, p. 435]).

Interleaving: To interleave several plans, we must build a new state machine where all mRIs in all plans are taken into account. Notice that we must usually preserve the order of execution of each plan to be composed. We can use algorithms to check behavior inheritance to ensure that this constraint is preserved, since to ensure this property, the composed plan must inherit from all the initial plans [7].

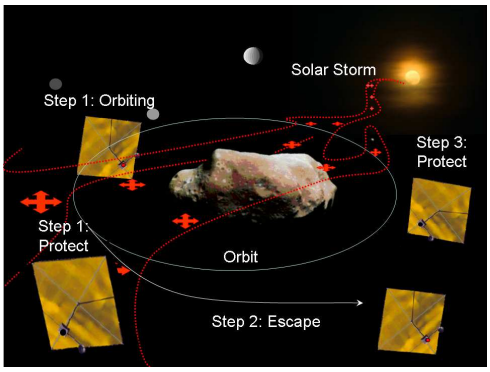

Figure 6. Policy for protecting from solar storms when orbiting

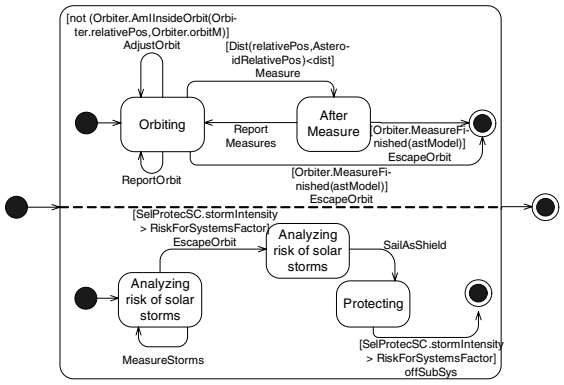

Figure 7. Composed plan

The composition of role model plans has to be performed following one of the plan composition techniques described previously. Later, if we are interested in the plan of one of the composed roles, we can extract it using the algorithms mentioned previously. We can also perform a composition of role plans following one of the techniques to compose plans described previously. Later, if we are interested in the plan of the composite role model, for example for testing, we can obtain it using the algorithms mentioned previously.

5.3 Example of applying a new policy to the ANTS case study

We use the following hypothetical scenario to document our example: It has been discovered that several spacecraft have collided with an asteroid as a result of self-protection from a solar storm. As a result, it has been decided to avoid protection from solar storms while orbiting, adding the following policy to the system, shown graphically in Figure 6.

If a spacecraft is orbiting and measuring an asteroid and it determines that there exists risk of a solar storm, the spacecraft must first escape the orbit and later power down subsystems and use its sail as a shield.

Note that we have limited the policy to two role models to simplify the example, but in the real world we must also take into account the rest of the autonomic properties and associated role models involved in orbiting an asteroid.

The first part of the policy shows the context where it is applied, determining the role models of the features that should be taken into account. In this example, two role models are affected by the policy introducing a new dependency between them.

If a spacecraft is orbiting and measuring an
Role Model
asteroid and it measures that there exists risk of a solar storm,
Interaction
the spacecraft must *first* escape the orbit and *later*
Interaction
power down subsystems *and* use its sail as a shield
Interaction **Interaction**

As a result, these features are taken from the M-RAAF and composed following the constraints imposed by the policy to obtain a modification of the original AOM. The composition of both role models, that is to say, the part of the AOM that is changed by the policy, is shown in Figure 8. As we can see, the roles *Orbiter* and *SelfProtectSC* have been composed into a single role called *SelfProtectingOrbiter* following the prescription shown in Section 5.1. We can observe that the rest of the roles have been left unchanged and that all mRIs have also been added without changes.

In addition, as the self protection must be taken into account during the whole process of orbiting and measuring, and not just in a concrete state, we must perform a parallel composition, as it is shown in Figure 7. Note also, that the policy tells us the order of mRIs we must follow for self-protection, adding the *Escape Orbit* mRI before protection, which results in the new state machine shown.

Later, the new AOM is re-transformed into the SOM, and this re-transformed into the PSM adding the platform specific details needed. Finally, these changes are deployed over the running system.

6 Conclusions

We have presented a extension of our prior work on Policy-Based Management for autonomous and and autonomic systems [15]. The approach is based on a general MDA-based procedure for policy deployment, and allows us to add policies to autonomic systems that change the functionality of an operational system, add to its functionality, or some combination.

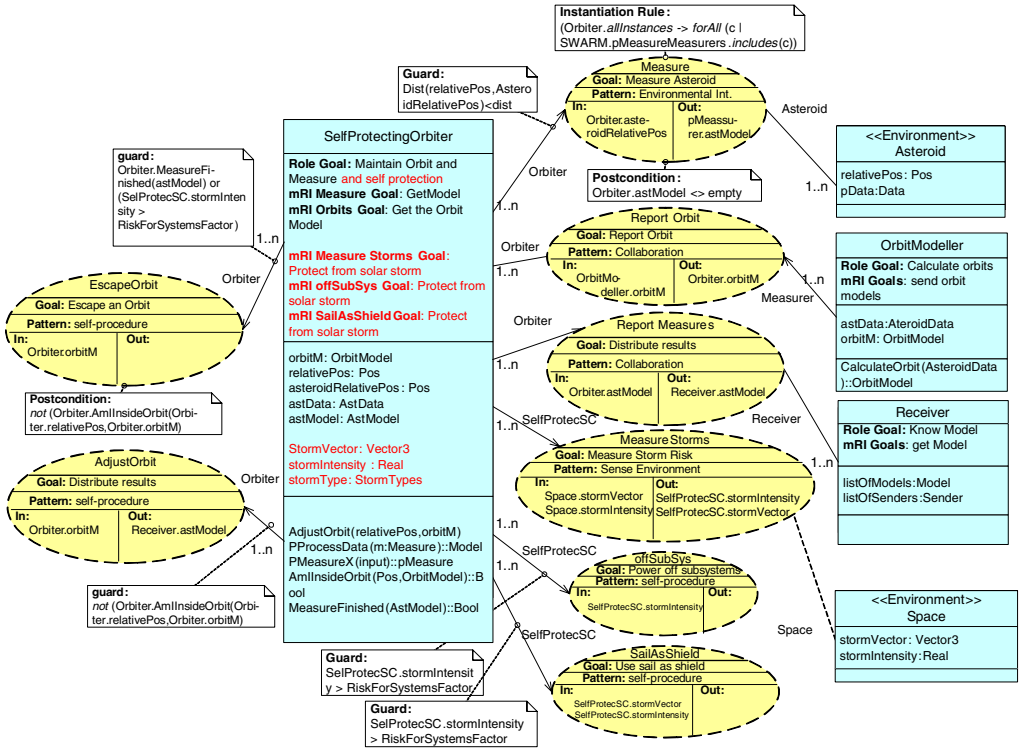

Figure 8. Composed Role Model

This paper has presented the initial two platform-independent models, and the corresponding transformation between them, and we have illustrated this with a simple example from a NASA concept mission. We have not entered into the details of further transformations, such as changing the structural organization of the system. However, given that such transformations are also based on role composition, we expect this work to extend easily to cover this. Also, although we have not presented the algorithms described more formally, implementations are available on the MaCMAS methodology website (http://www.tdg-seville.info/joaquinp/MaCMAS).

We believe that this work is more structured that the approached described in our previous paper [15] and that it is more consistent with standard approaches in the software industry (such as MDA and UML), which is necessary in order to support the industrial uptake of Policy-Based Self-Management.

References

[1] IEEE Task Force on Autonomous and Autonomic Systems, (TFAAS), June 2005. Available at http://www.computer.org/tab.

[2] S. A. Curtis, W. F. Truszkowski, M. L. Rilee, and P. E. Clark. ANTS for the human exploration and development of space. In *Proc. IEEE Aerospace Conference*, Big Sky, Montana, USA, 9–16 March 2003.

[3] C. He, W. Tu, and K. He. Role based platform independent web application modeling. In *PDCAT*, pages 411–415. IEEE Computer Society, 2005.

[4] J. Heflin and J. Hendler. Dynamic ontologies on the web. In *AAAI/IAAI*, pages 443–449, 2000.

[5] P. Horn. Autonomic computing: IBM perspective on the state of information technology. In *AGENDA'01*, Scottsdale, AR, 2001, (available at http://www.research.ibm.com/autonomic/).

[6] E. A. Kendall. Role modeling for agent system analysis, design, and implementation. *IEEE Concurrency*, 8(2):34–41, Apr./June 2000.

[7] B. Liskov and J. M. Wing. Specifications and their use in defining subtypes. In *Proceedings of the Eighth Annual Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA*, pages 16–28. ACM Press, 1993.

[8] S. Mellor, A. Clark, and T. Futagami. Model-driven development - guest editor's introduction. *IEEE Software*, 20(5):14–18, Sept. 2003.

[9] P. Mitra and G. Wiederhold. An ontology-composition algebra. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 93–116. Springer-Verlag, 2004.

[10] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic integration of knowledge sources. In *Proc. of the 2nd Int. Conf. On Information FUSION'99*, 1999.

[11] O. M. G. (OMG). Unified modeling language: Superstructure. version 2.0. Final adopted specification ptc/03–08–02, OMG, August 2003. www.omg.org.

[12] J. Peña, R. Corchuelo, and J. L. Arjona. Towards Interaction Protocol Operations for Large Multi-agent Systems. In *Proceedings of the Second International Workshop on Formal Approaches to Agent-Based Systems (FAABS 2002)*, volume 2699 of *LNAI*, pages 79–91, NASA-Goddard Space Flight Center, Greenbelt, MD, USA, 2002. Springer–Verlag.

[13] 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.

[14] J. Peña, R. Corchuelo, and M. Toro. Representing complex multi-agent organisations in UML. In *IX Jornadas de Ingeniería del Software y Bases de Datos JISBD'04*, pages 159–170, Málaga, Spain, 2004.

[15] J. Peña, M. G. Hinchey, and R. Sterritt. Towards modeling, specifying and deploying policies in autonomous and autonomic systems using an aose methodology. In *3rd IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2006)*, pages 37–46, Columbia, MD, 2006. IEEE Computer Society Press.

[16] R. Sterritt, C. A. Rouff, J. L. Rash, W. F. Truszkowski, and M. G. Hinchey. Self-* properties in NASA missions. In *4th International Workshop on System/Software Architectures (IWSSA'05) in Proc. 2005 International Conference on Software Engineering Research and Practice (SERP'05)*, pages 66–72, Las Vegas, Nevada, USA, June 27 2005. CSREA Press.

[17] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: the GAIA methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.