roar @UEL

research open access repository

University of East London Institutional Repository: http://roar.uel.ac.uk

This paper is made available online in accordance with publisher policies. Please scroll down to view the document itself. Please refer to the repository record for this item and our policy information available from the repository home page for further information.

To see the final version of this paper please visit the publisher's website. Access to the published version may require a subscription.

Author(s): Beydoun, G., Low, G., Mouratidis, H., Henderson-Sellers, B.
Article title: Modelling MAS-Specific Security Features
Year of publication: 2007
Citation: Beydoun, G. Low, G. Mouratidis, H., Henderson-Sellers, B. (2007) 'Modelling MAS-Specific Security Features' In: Erik, H.A., Halpin, T.A., Krogstie, J. (eds) "Proceedings of the Workshop on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD'07), held in conjunction with the 19th Conference on Advanced Information Systems (CAiSE'07)", Trondheim, Norway, Tapir Academic Press, Trondheim, Norway pp 179-188
Link to published version: http://www.via-nova-architectura.org/files/EMMSAD2007/Beydoun.pdf
ISBN: 9788251922456
Publisher statement: http://www.tapirforlag.no/forlag/english/

Information on how to cite items within roar@uel:
http://www.uel.ac.uk/roar/openaccess.htm#Citing

Modelling MAS-Specific Security Features

G. Beydoun[1], G. Low[1], H. Mouratidis[2], B. Henderson-Sellers[3]

[1]University of New South Wales, Australia
{g.beydoun, g.low}@unsw.edu.au
[2] School of Computing and Technology, University of East London, England
haris@uel.ac.uk
[3]University of Technology, Sydney, Australia
brian@it.uts.edu.au

Abstract. In this paper, we pursue a modelling approach to address security requirements for multi-agent systems (MAS). This will allow developers to account for both the system and agent-specific security requirements of a MAS during the requirements phase and throughout the whole Software Development Lifecycle of the system. We focus on autonomy, mobility and cooperation of individual agents and how these create additional security vulnerabilities to the system. In proposing a set of generic modelling primitives for these engendered requirements in the analysis of the MAS, we extend our recently proposed MAS metamodel.

1. Introduction

Agents are highly autonomous, situated and interactive software components. They autonomously sense their environment and accordingly respond as they are enabled. In a given Multi-agent System (MAS), coordination and cooperation between agents possessing diverse knowledge and capabilities facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation [1]. Autonomy of agents, cooperation amongst them and intensive communication between them incur additional security requirements for a MAS. Furthermore, agents are particularly useful in the engineering of open, distributed or heterogeneous systems. Indeed, agent technology is poised to exploit the pervasiveness of the internet and the availability of cheap computational power at its individual nodes and agents have already proved effective in varying tasks and in different organizational settings: automating management of information within businesses [2], e-commerce trading environments [3] and building computational models of human societies to study emergent behaviour [4].

We have developed and successfully evaluated a generic metamodel, called FAML (FAME[1] Agent-oriented Modelling Language), to describe features of any MAS [5, 6]. FAML was designed to support a set of general concepts relevant to model any MAS. Some problem-specific concepts including security and mobility were initially omitted. The chosen concepts are designated into two sets: run-time concepts and design-time concepts. Each set has two scopes: system-related or agent internals related scope.

---

[1] FAME (Framework for Agent-oriented Method Engineering) is the project name under which FAML has been developed.

We recognize that the security requirements of a given MAS are either generic (applying to any software), application-related (part of the functional requirement) or MAS-specific (due to their engendered properties). This paper describes an effort to complement our existing MAS metamodel, FAML, with the capacity to model the security requirements of any given MAS. We further recognize that these security requirements are of two kinds: system security requirements and agent-specific security requirements. The former applies to all agents in the system while the latter applies at the level of the individual agent. In this paper, we analyze the design tradeoffs in accommodating both the system and agent-specific MAS security requirements, but we focus more on modelling the former.

Our work is not the first to consider modelling of security issues of multi-agent systems, example of others are [7-10]. However, our work is unique in that it simultaneousely focuses on three important properties of a software agent: autonomy, mobility and cooperation. Although these properties are important for many agent systems, particularly those in situations where security is paramount, the above approaches do not consider all of them at the same time. Most related to our suggested approach is that of Mouratidis *et al.* [11] on the definition of an architectural description language (ADL) to specify secure multi-agent systems. This work proposes a set of design primitives conceptualized using the Z specification language to capture a "core" architectural model to build secure MAS architectures. However, this work differs from our approach in two ways: firstly, it does not consider mobility and, secondly, as its authors state, it lacks a suitable set of core abstractions, inspired by organizational metaphors, to be used during the design of the secure multi-agent system architecture. Therefore, we believe that our work complements [11] by providing the missing set of core abstractions.

The rest of the paper is organized as follows: Section 2 describes our analytic process used to identify the required security modelling units. Section 3 articulates the MAS-specific security concerns of any MAS and associates these with basic modelling primitives. It shows our existing MAS metamodel extended to incorporate the identified units. Section 4 discusses the proposed metamodel while Section 5 concludes with a discussion of future work.

2. A Security metamodel for MAS

We are advocating a software modelling approach to address security requirements of multi-agent systems, as suggested in recent work e.g. [12, 13]. This allows developers to account for security of a MAS early during the development of the system rather than as a costly afterthought. Our work in this paper is a step towards a coherent approach to provide methodological support for the creation of secure MASs. We systematically start the synthesis of a set of modelling units (classes in the metamodel) required to model the security of a MAS. Our systematic approach to the synthesis justifies why each unit is needed. In particular, our approach is to find a minimal set of modelling units to model the MAS-specific security requirements, identified by the MAS developer during design stage, and a number of security actions, performed by the MAS during run-time to satisfy the system's security requirement. We conduct our analysis along two dimensions: the inherent attributes of agents in a MAS and an increasing scale of security risk (see Table 1). Modelling unit sets associated with various attributes may overlap. The context complexity ranges from a single agent on a single machine to a fully mobile MAS with agents capable of roaming the net using dynamic multi-hop routing. An agent can be *strongly* mobile, where it carries its code, data and its state of execution (i.e. program counter and CPU registers); or it can be *weakly* mobile, carrying code and data only. We start our analysis in the least risky environment (a single agent on a single machine). We aim to extend this to overcome

its limitations in increasingly risky settings until we have a complete security model that can represent all security requirements and solutions for the riskiest setting.

We identified in total nine different levels of vulnerabilities as shown in Table 1. The agent vulnerability level increases as we move from level 0 to level 8. Consequently the need for protection such as encryption also increases the further we move away from level 0. In the case of Level 2, data is transported (for message passing) while, for Level 3, both data and code may be transported (for remote evaluation), which allows application level attacks. Beyond Level 3, distributed MAS systems solutions will no longer be viable. It becomes more complex to ensure safe transportation when states of execution are involved or when intermediate cooperation is not guaranteed (hostile hosts may be encountered - this is not the case for distributed systems). The scale of vulnerabilities, as shown in Table 1, delineates the research issues at each level of complexity and can be used as a research roadmap to fulfil MAS-specific security requirements of the most complex forms of MAS.

We seek to model features required for non-mediated decentralized security, without having to mediate interactions between agents by trusted secure agents. The mediated approach to security e.g. in [3] assumes two classes of agents, *internal* and *external,* and by-passes many security problems by having the internal secure agents mediate all interactions between external non-secure agents. The mediated approach limits the potential of a MAS and can secure MASs only up to Level 2 (Table 1), where internal agents reside in a central server (Level 1) and where external agents send messages to the central location of internal agents. Our approach will be more general than the mediated approach.

Table 1. Context Complexity of MAS

Vulnerability level	Description of Context Complexity Level
0	A single agent system on a single machine.
1	MAS running on a single machine– controlling access to agents' resources during cooperation
2	Distributed MAS (same as MAS if communication is secured)
3	Weakly mobile agents with single hops
4	Strongly mobile agents with single hops
5	Weakly mobile agents with fixed multiple hops
6	Weakly mobile agents with dynamic multiple hops
7	Strongly mobile agents with fixed multiple hops
8	Strongly mobile and roaming agents

We note that agent security requirements for communication channels are equivalent to normal requirements for confidentiality, integrity, authentication and availability in a typical software application [14]. MAS-specific security problems do not include security of communication channels. Our security framework strives for authenticated communication between agents (including mobile agents), where any receiving agent can ascertain the identity of the sender and can choose to block an agent as it sees fit. Any agent can also deny/offer access to its owned resources (including its internal state) at any time it wishes. Resources that need to be protected can be local (owned by a single agent) or global (owned by many agents), distributed or reside on a single device.

Our security framework is integrated into our generic MAS metamodel, FAML which can be used to represent workproducts of any MAS methodology. We extend FAML in the next section to ensure that any secured workproduct (one which takes into account security requirements) can also be represented.

3. General MAS Security Vulnerabilities

In this section, we establish the analytic process to identify modelling units that software engineers can use to describe security requirements that are common to all multi-agent systems (and to any agent within any multi-agent system). We overview the general security requirements associated with the features of agents of cooperation, mobility and autonomy. We note basic modelling units to express these requirements. Our concerns are authentication and resource access control. More specifically, we are concerned with modelling features of agent interactions and with additional data structures to provide secure access and detection as well as recovery in the case of a security violation. For our purpose we assume messages are transmitted over secured communication channels, since interception problems are not MAS-specific. Our analysis is a first step to extending our existing MAS metamodel [6]. The extended metamodel, FAML, will retain the same structure. It will still have two layers: design-time and runtime layers and each layer will still have two scopes: agent-related and system related. However, each of the four layers+scopes will be extended to accommodate the general security requirements associated with the engendered properties of agents: *cooperation, autonomy* and *mobility*. In the rest of this section, we identify security modelling units associated with the each of those properties. The integration of these units into FAML is presented later in Section 4.

3.1. Cooperation, autonomy and security of agents

Cooperation between agents in a MAS requires mutual agreement between agents to share resources and, in some instances, to share access to their internal states [1]. To simplify our current analysis, we assume that agents' resources are localized with the agent and that any resource is owned by only one agent. The broader the mutually agreed access to resources and internal states is, the richer the functionality of the system potentially becomes. It is essential for agents to be able to trust other agents in order to provide a broad set of interactions. To implement an agreement to share access to resources and internal states between agents, i.e. to cooperate, it is critical for agents to discern changes in their states or resource arising from a legitimate access, and from changes arising from malicious intrusion. A way to achieve this is for each message to be uniquely associated with an agent identity and time stamped. We include *agent_identity* as a basic modelling unit to describe an agent uniquely. This identity can be a function of the system to which an agent belongs, serving as a fingerprint of any agent created in any MAS. A higher level security concept, *signature,* can be implemented using *agent_identity*. A signature is to be known by trusted parties and producible by a relevant identity [16]. Our extended metamodel will also describe the relation between agents and their resources through *ownership* and *usage* relationships.

To maintain its autonomy, an agent should be resilient and able to recover from an interaction that gives access to resources when it shouldn't. In other words, maintaining some state description of its past interactions is required. Each agent should have an *interaction_history* log to allow recovery. Such a log is only accessible to the agent itself This concept along with *ownership* secures the internal state of agent, which can then cooperate to reinstate its state as well as the state of the MAS if and when needed. Mediated systems e.g. [17] use a system interaction history, but this requires centralized access, which we avoid since it places limits on the mobility of agents. Our decentralized framework can model a mediated solution by having a single agent designated to mediate and to interact with all other agents. The interactions log of the 'mediator-designate' will then be a log of all interactions within the system, as in [17].

3.2. Mobility and security of agents

Mobility allows agents to replace remote procedure calls, saving on bandwidth and allowing computations that otherwise are not possible. In comparison with traditional distributed systems, mobility allows additional functionality for a MAS but also incurs additional security requirements. Mobility of agents varies, as noted in Table 1. For *weakly* mobile agents, discerning intrusion is easier, since the agent starts execution from scratch when it reaches its host. The essential problem with mobile agents is compounded by the fact that the host may require access to agent execution states [18]. Hence, if the host is malicious, intrusion recovery is needed.

Assuming that an agent is transmitted safely along a channel, threats to an agent (mobile or non-mobile) resources come from interacting with at least the following sources: (i) individual or group of agents in its environment - for example, one or more mobile malicious agents designed to steal or corrupt data; (ii) the host: the host that controls its execution could mistake the identity of the agent or it may simply be a malicious luring host; (iii) users themselves: unauthorized users might attempt to corrupt or steal an agent's data, or attempt to infiltrate the functionality of the MAS.

The extent of the vulnerability of a mobile agent depends on the freedom it has with respect to its *mobility*. Mobility can be single hop (from host to another without any intermediate hosts) versus multiple hop (one or many intermediate hosts) or fixed (travel path is fixed and statically decided) versus dynamic (path is decided by agent as it travels, it is said to be roaming) (see Table 1). Mobility requires the *location* modelling unit so that an agent is able to reason about its movements. Combined with its *interaction_history* (history of hops as well for mobile agents), it can reinstate itself into a safer location if it is under attack.

4. Proposed Metamodel

We recognize in this paper that MAS-specific security requirements are either generic, application-related or MAS-specific. In Section 3, our focus was on the third kind engendered by the inherent characteristics of agents and MASs. This suggested a number of new modelling units, which we integrate into FAML concepts in this section. We first give the complete set of the new security FAML concepts, together with their corresponding strict definitions. The new security design-time concepts with their definitions are:

- *Agent-specific Security Requirement:* Security requirements specific to an individual agent
- *Location Specification:* Information about places where an agent can reside within the system
- *Plan Resources Specification:* Specification of resources used in the Plan Specification
- *Private Resources Specification:* Specification of resources only visible and available to the individual agent
- *Recover Action Specification:* Specification of an action undertaken for the specific purpose of recovery from an intrusion or security breach event. For example, an agent can use the interaction log to reinstate its state as well as part of the state of the MAS if an interaction gives access to resources when it shouldn't
- *Relocate Action Specification:* An agent can move to another location based on information contained in the interaction log and agent-location if an interaction gives access to resources when it shouldn't

- *Resource Specification:* Something that has a name, may have reasonable representations and which can owned. The ownership of a resource is connected with the right to set policy on the resource
- *Security Action Specification:* Action that an agent takes if an interaction gives access to resources when it shouldn't
- *Security Requirement:* A desirable security-related property that constrains functional requirement(s)
- *System Security Requirement:* Security requirements that are true for all agents of the MAS

The new security run-time concepts and their definitions are:
- *Location:* Information about places where an agent can reside within the system
- *Plan Resources:* An organised collection of resources used in the Plan Specification
- *Private Resource:* Resources only visible and available to the individual agent e.g. agent's history log of hops as well as interactions with other mobile agents
- Public Resource: Resource that is used by one or more agents in the system e.g. database resource
- *Recover Action:* An agent can use the interaction log to reinstate its state as well as the state of the MAS if an interaction gives access to resources when it shouldn't
- *Relocate Action:* An agent can move to another location based on information contained in the interaction log and agent-location if an interaction gives access to resources when it shouldn't
- *Resource:* Something that has a name, may have reasonable representations and which can owned. The ownership of a resource is connected with the right to set policy on the resource. (http://www.w3.org/TR/ws-arch/#resource)
- *Security Action:* Action that an agent takes if an interaction gives access to resources when it shouldn't

It is worth mentioning that we distinguish between run-time security modelling units and design-time security modelling units (shown above). In terms of security, this means that we effectively differentiate between security requirements, which are modelled by the software developer during the design stage, and security actions, which are performed by the multi-agent system during the run-time to satisfy the security requirements. Such differentiation allows, on the one hand, the development of a security-aware platform independent design; and, on the other hand, the realization of a working security-aware multi-agent system for a specific platform.

We present the extended FAML in four diagrams to integrate the security modelling units into the same existing areas of concern for FAML (Fig. 1-4): design-time system related, runtime system-related (environment), design-time agent-internals and runtime agent-internals area of concern. Note that each agent assumes responsibility for its security; this is modelled with Recover Action Specification (Fig. 2). An example recovery action may be what the agent does to use the interaction log to reinstate its state and perhaps assist in reinstating of the MAS if an interaction gives access to resources when it shouldn't. This is a refinement of a more general modelling unit, Security Action Specification. FAML is extended with modelling units to express mobility behaviour of agents with the units Relocate Action Specification and Location Specification. Relocating is a restricted action that requires access to secured resources of the agents (i.e. only the agent can relocate itself).
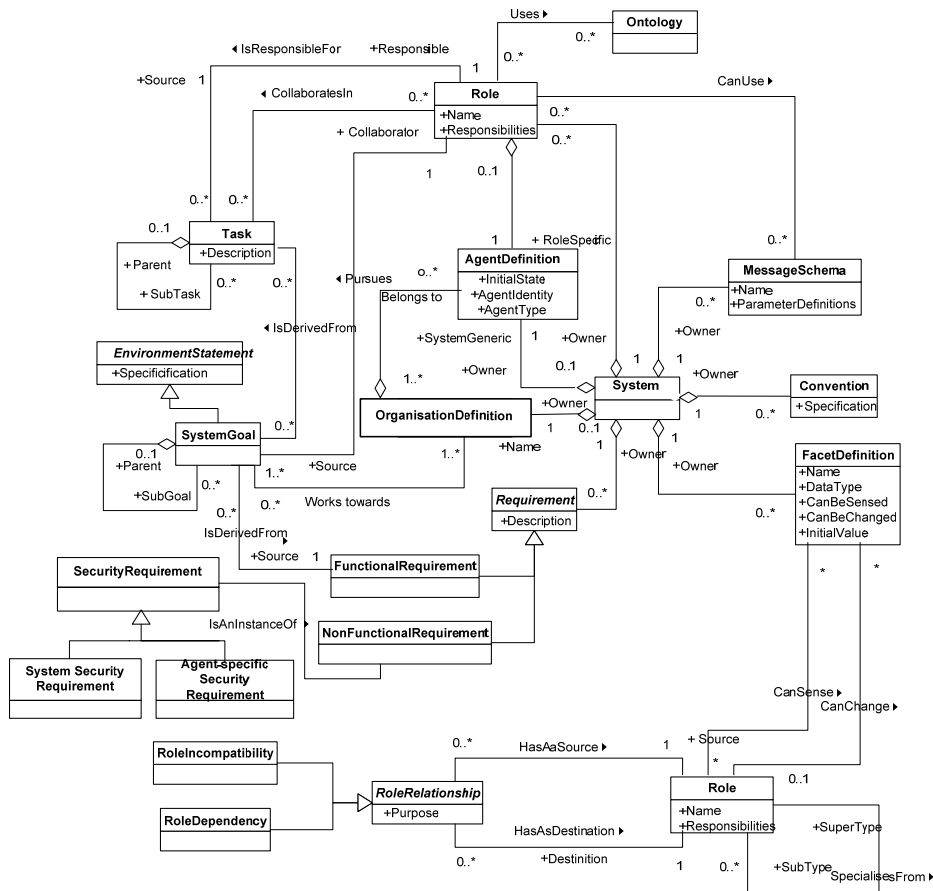
Fig. 1. System-level classes (Duplication of the Role is only to simplify the layout)
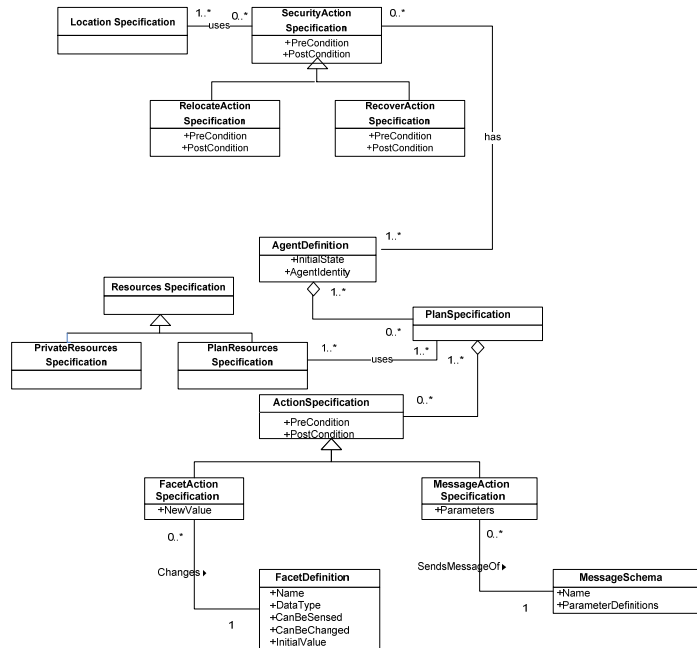
Fig. 2. Agent definition-level classes at design time

Our security framework is underpinned by recognizing and modelling the status of access to resources during the development. In the extended FAML, the modelling units Private Resource and Public Resource are added (Fig. 3). Private resources are agent-specific including logs of hops and interactions with other mobile agents.
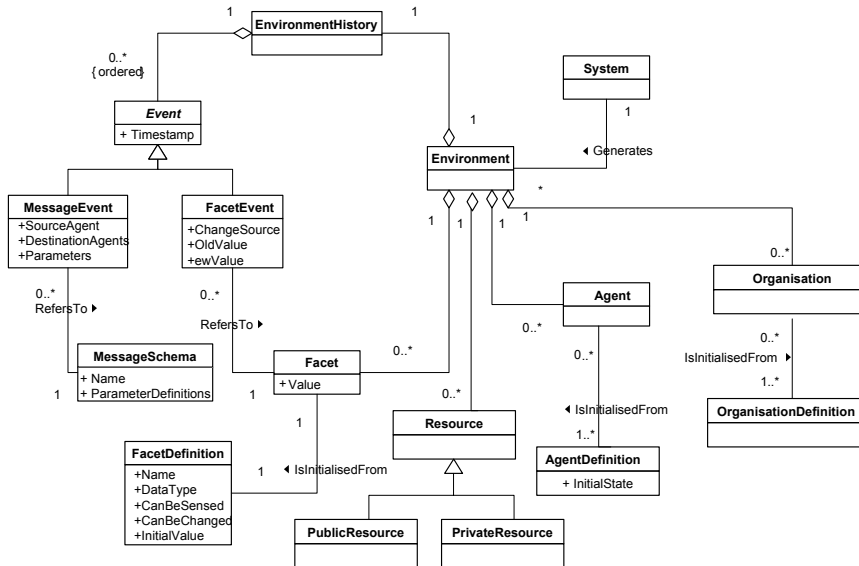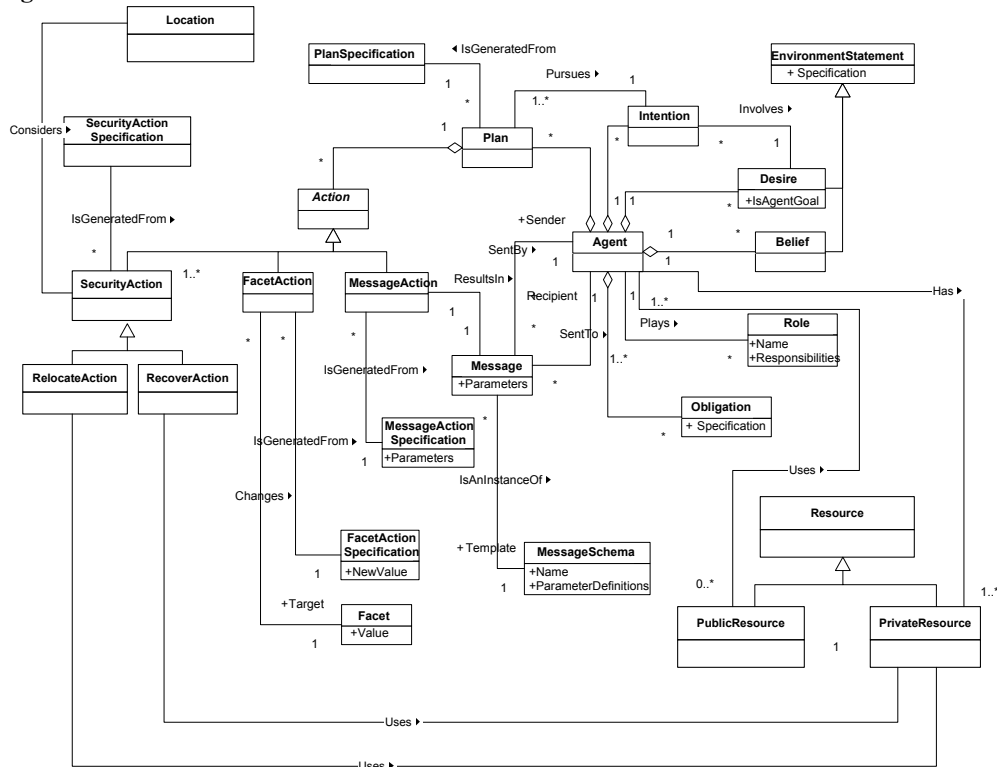

Fig. 3. Environment-level classes

Fig. 4. Agent-level classes at run time

At runtime, central to agent security are authentication and recovery when that fails. Therefore at the agent-runtime scope (shown in Fig. 4), FAML is extended with the

modelling units to represent the various classes of resources and their access (the same as agent-design time taxonomy of resources). In addition, it is extended to enable modelling of restricted actions.

5. Conclusions and Future work

FAML to any MAS is similar to UML to any OO system. It is a product metamodel allowing the representation of any workproduct during the development of a MAS. Our original FAML did not accommodate the security requirements of the system, and it did not allow description of security solutions such as the ones discussed in this paper. The extended FAML presented in this paper will allow software developers to describe security solutions and produce secured workproducts as the system is developed. In extending it to represent secured workproducts, we preserve its structure.

In our security framework, managing the MAS-specific security requirements is decentralized and is relegated to the individual agents forming the system. There are still a few outstanding questions with respect to modelling units so far identified e.g. can agents themselves be *owned?* Further work is required to complete our set of security modelling units. For example, we still need to identify what (if any) additional modelling units are required for mobile agents with dynamic routing. It should be noted that these units alone (even when completed) would not suffice to model all security features, although out of these basic units, and clever programming, more complex concepts can be derived such as access level, authority levels, passports, signatures, worth of resources.

When our MAS existing metamodel [6] is completed with most of its required security features, we will start verifying it by applying it to the analysis of security requirements of a community-based peer-to-peer web search engine. We envisage that fulfilling the highest levels of complexity will be particularly challenging.

Hence, we will overlap verification with the development of the security-enhanced metamodel described here. Our future work will also link the development of secure MASs and their related access policies with risk management standards (e.g. ISO17799, ISO7498/2) as applied at an organizational level. This would guide MAS developers in making inevitable tradeoff decisions of security versus cost and functionality. Authentication, intrusion detection and recovery require more computation and storage of relevant features of the involved interactions and resources by each agent. Diverting too many resources (e.g. storage, computation) towards security may indirectly limit the functionality of the system. Investing in reducing security has a point of diminishing returns, most of the benefits being reaped with the first chunk of the cost. In future work, we will guide developers in tackling such complex tradeoff decisions.

References

1.   Wooldridge, M.: Multi Agent Systems, Vol. volume 3. Wiley, Chichester (2002)
2.   Horlait, E.: Mobile Agents for Telecommunication Applications (Innovative Technology Series: Information Systems and Networks). Kogan Page, Portland (2004)
3.   Rodriguez, J.A.: On The Design and Construction of Agent-Mediated Electronic Institutions. Artificial Intelligence Research Insitute. UAB - Universitat Autònma de Barcelona, Barcelona (2003)
4.   Tidhar, G., Heinze, C., Goss, S., Murray, G., Appla, D., Lloyd, I.: Using Intelligent Agents in Military Simulation or " Using Agents Intelligently". 11th Conference on Innovative Applications of Artificial Intelligence Papers(IAAI-99), Vol. 1. MIT Press, Orlando,Florida (1999) 829-837

5. Beydoun, G., Gonzales-Perez, C., Low, G., Henderson-Sellers, B.: Towards Method Engineering for MAS: A preleminary validation of a generic MAS Metamodel. 17th Software Engineering and Knowledge Engineering Conference (SEKE2005) (2005)

6. Beydoun, G., Gongalez-Perez, C., Low, G.C., Henderson-Sellers, B.: Synthesis of a Generic MAS Metamodel. International Conference on Software Engineering (ICSE05) Workshops (SELMAS2005). ACM Digital Library, St Louis, USA (2005) 27-31

7. Liu, L., Yu, E., Mylopoulos., J.: Analyzing Security Requirements as Relationships Among Strategic Actors. 2nd Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina (2002)

8. Huget, M.-P.: Nemo: An Agent-Oriented Software Engineering Methodology. OOPSLA Workshop on Agent-Oriented Methodologies, Seattle, USA (2002)

9. Mouratidis, H.: A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of Health and Social Care Needs of Older People in England. Department of Computer Science, Vol. PhD. University of Sheffield (2004)

10. Odell, J., Van Dyke Parunak, H., Bauer, B.: Extending UML for agents. In: G. Wagner, Y.L.a.E.Y. (ed.): Agent-Oriented Information Systems Workshop, 17th National Conference on Artificial Intelligence, Austin, TX, USA (2000) 3-17

11. Mouratidis, H., Kolp, M., Faulkner, S., Giorgini, P.: A Secure Architectural Description Language for Agent Systems. 4th International Joint Conference on Autonomous Agents and MultiAgent Systems. ACM Press, Utrecht- the Netherlands (2005) 578-585

12. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Requirements Engineering meets Trust Management. Second International Conference on Trust Management (iTrust 2004) (2004)

13. McGraw, G.: Software Security. Addison-Wesley, Indiana (2006)

14. Borselius, N.: Security in Multi-Agent Systems. In: Mun, Y., Arbania, H.R. (eds.): International Conference on Security and Management (SAM02). CSREA Press, Nevada (2002)

15. Beydoun, G., Gonzalez-Perez, C., Henderson-Sellers, B., Low, G.C.: Developing and Evaluating a Generic Metamodel for MAS Work Products. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.): Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications, Vol. LNCS 3914. Springer-Verlag, Berlin (2006) 126-142

16. Roth, V.: Programming Satan's agents. International Workshop on Secure Mobile Multi-Agent Systems, Montreal, Canada (2001)

17. Esteva, M., Cruz, D.d.l., Sierra, C.: ISLANDER: an electronic institutions editor. International Conference on Autonomous Agents & Multiagent Systems (AAMAS02). ACM, Italy (2002)

18. Yee, B.: Monotonicity and Partial Results Protection for Mobile Agents. 23rd International Conference on Distributed Computing Systems, Rhode Island (2003)

19. Mouratidis, H., Giorgini, P.: Integrating Security and Software Engineering. Idea Group Inc, Hershey, PA, USA (2007)