Towards Self-managing Unmanned Autonomous Vehicles

E. Asmare, N. Dulay, E. Lupu, M. Sloman
Imperial College London, Department of Computing, 180 Queen's Gate, London SW7 2AZ, UK

Abstract

This paper presents a mission management architecture for Unmanned Autonomous Vehicles. An outline of a mission specification scheme is also presented. Elements of the mission management architecture and the mission specification scheme are described and illustrated using a reconnaissance scenario.

Keywords: Mission-management, Policy-based robots, Self-management

Introduction

Unmanned Autonomous Vehicles (UXVs) are often used in applications where it might be dangerous or difficult for humans to enter the area. For example searching collapsed buildings, areas where there may be explosives or chemicals. Typically UXVs are self-propelled, with on-board computers for processing, communication, and data storage and may include infrared sensors for obstacle detection, video cameras and possibly specialised sensors to detect explosive materials, radiation or chemicals. They may also include some form of autonomous navigation capability, adaptive mission planning, target recognition or identification and situation assessment.

UXVs need to adapt their behaviour to current context -location, activity, available resources such as battery power and available services such as quality of communications link. They should be self-managing in that they have to recover or adapt to component failures and optimise performance to best utilise available resources. Additionally, a team of UXVs can cooperate to achieve a particular mission such as surveillance of a specific area or search for specific targets. Individual UXVs may have particular sensors or communication devices required to fulfil specific roles in an application and the UXVs typically form a mobile ad-hoc wireless network for communication amongst themselves and with human controllers.

In this paper we describe a distributed event-based self management architecture for UXVs which makes use of policies to define adaptive strategy for both individual UXVs and groups of cooperating UXVs. First we give an overview of the Self-Managed Cell concepts and policy-based management then we present our policy-based mission management framework. Finally we conclude by summarising the paper and pointing out future works.

Background

The Self Managed Cell

A Self Managed Cell (SMC) [8] is an architectural pattern for combining the services and components needed for self-management of a set of components ranging from a body-area network to large-scale distributed applications in which multiple devices collaborate to achieve a common goal. The SMC provides an extensible framework for supporting self-organisation, self-healing, self-protection and self-optimisation.

The management services in an SMC interact with each other via an asynchronous event bus. As a minimum an SMC should contain a policy service to implement a feed-back loop adaptation, a discovery service to discover the components which form the cell and a publish-subscribe event service to implement the event bus. The basic architecture of an SMC is shown in Figure 1 and the core services are briefly described in the following.

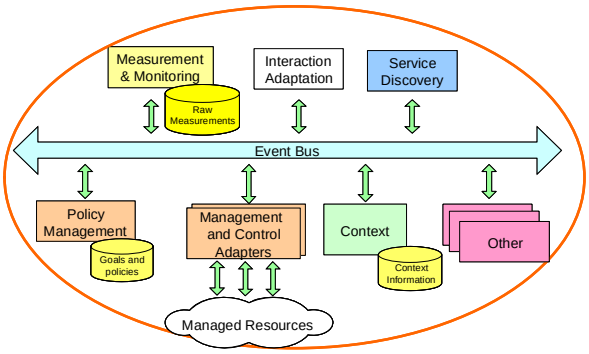

Figure 1: Architecture of a SMC

The Policy Service is the means of specifying the adaptive behaviour of an SMC in terms of obligation policies or event-condition-action rules as well as authorisation policies to control what service and resources can be used by specific entities.

The Discovery Service discovers components which are in range and capable of being members of the SMC.

The Event Bus is responsible for asynchronous notification of events to different management services of the SMC. Event notification is a crucial element of an SMC because adaptation, protection and other self-management actions are specified in terms of obligation policies triggered by events. An obligation policy may perform an action which modifies the behaviour of a single component of an SMC or it may enable or disable other policies to change the overall behaviour of the SMC.

The use of an event bus also allows maintaining loose coupling between the various SMC services which can react concurrently and independently to event notifications. However all communication is not constrained to be over the event bus and simple messaging or object invocation may be used as appropriate.

The Measurement and Monitoring Service is responsible for keeping track of the SMC's operation and generates events when actions need to be taken. Events may indicate situations such as degradation or failure of components. The Context Service uses sensors to determine information such as current location, weather conditions, or what obstacles are in the vicinity as the SMC behaviour will adapt to current context.

A group of cooperating SMCs may compose or federate to form a single self managing system (SMC) in order to collaborate to achieve a particular mission.

Policy-based Management

Policies are rules defining choices in behaviour. We use obligation policies (event-condition-action rules) to trigger specific task to be performed when an event occurs such as a component fails or a specific chemical is detected. Policies can be changed dynamically without shutting down a system so they facilitate very flexible management. Authorisation policies specify what services and resources within an SMC can be accessed by other SMCs performing a specific role.

Policies are interpreted; as a result they can be dynamically loaded, enabled or disabled at run-time without shutting down a system in order to adapt the management strategy being used within an SMC. Thus policy-based management enables flexible and adaptive automation of systems management for realizing self management. [2,9,10].

We use the Ponder policy framework in our mission management architecture. Ponder [3,4] is an object-oriented policy-specification language developed at Imperial College. It provides a common language which enables administrators to specify a policy, hence unifying the concepts and models of various policy related research. It supports a family of policies named Access Control Policies which are used for security management and Obligation Policies which are used for service management.

Ponder2 [1] is the latest version of the Ponder policy framework. It is a generic object management system which allows dynamically loading, unloading, enabling and disabling of managed objects. Generally a Ponder2 managed-object is an active object that is capable of receiving action commands and performing actions.

Ponder2 is comprised of three components, the domain service, the obligation policy interpreter and the command interpreter. The domain service provides hierarchical structure for managing objects. The obligation policy interprets event-condition-action rules. The command interpreter accepts and interprets a set of commands, directed to managed-object in the domain structure, in the form of XML documents.

Policy-based Mission Management Architecture

A mission is a set of sequential or concurrent tasks which must be performed in order to achieve an overall goal. A planning process is needed to generate the tasks from the goal. The tasks may also be sub-missions as we allow for nested missions. The planner may generate more than one strategy for achieving a goal or the context may change such that the strategy for achieving the goal has to adapt to the current situation. The implication of this is that the UXV mission execution architecture should allow adaptation of missions. We are using a policy based approach for developing a self-managing mission framework for UXVs.

Arkin et al. [7] define mission, in the context of robots, as a composition of tasks, and mission specification as the process in which step-by-step instructions are generated to guide one or more robots to accomplish a set of tasks. We use a similar notion of mission. Hence, a mission, in our system, is either a simple set of tasks or a composition of sub-missions which by themselves could be a composition of other sub-missions.

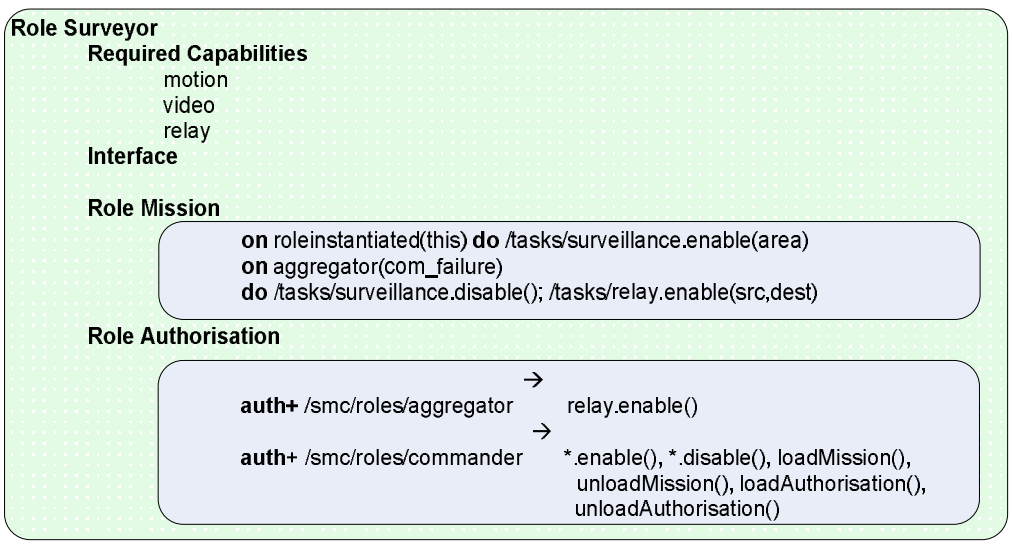

Figure 2: Surveyor Role Definition

As indicated in Figure 2, a mission can be hierarchically structured. We use the term Overall mission-specification (Mission-SMC specification) to refer to the overall specification of policies and roles relating to a group of cooperating UXVs which perform the roles within a Mission-SMC.

The Mission-SMC is an SMC formed by a group of UXVs as a result of interpreting the overall mission-specification. A role is defined within a Mission-SMC and consists of the set of tasks and policies which will be loaded onto a UXV assigned to the role, and effectively specifies the behaviour of the role, i.e., the UXV executing the role.

Role

A role is specified in terms of tasks to be performed by entities assigned to the role and policies which control tasks and interaction of the role with other roles within the Mission-SMC.

The tasks to be performed by a role are described in terms of obligation policies and this description is a specification of the role's mission. When an entity such as a UXV is discovered, it is assigned to a role based on its capabilities, i.e., the inherent functions it can perform with the devices it has and the credentials it provides. It will then be provided with a role specification containing policies and tasks to control how it behaves within the Mission-SMC.

The policies within the role's mission may be triggered by events to cause switching between tasks. A role can also use services provided by other roles, as permitted by the authorisation policies.

Our approach for role definition is illustrated using a simple reconnaissance scenario.

In this scenario there are four UXVs and the objective is to collect data regarding the layout and contents of a house to determine whether it is safe for humans to move in to the house or not.

First, we describe possible roles in the scenario and then show a role definition for one of the roles.

We can identify the following roles in this scenario:

- *Commander*: this is a manned vehicle with a range of communications equipment. It is responsible for managing the mission.
- *Surveyors*: the surveyors collect data (for instance video) while exploring the house. They send the collected data to the aggregator which relays it to the commander.
- *Aggregator*: produces a map of the whole space and distributes it back to the surveyors so that they can use it for the remaining reconnaissance. It will also send all the aggregated information to the command centre.

These roles act as 'placeholders' for the purpose of specifying the required behaviour of the UXVs assigned to roles. This behaviour is specified in the form of policies which define the missions which the UXV will perform (in terms of role-mission, see Figure 2) and privileges with respect to accessing services provided by other roles or shared resources within the SMC.

Figure 2 shows the definition for the Surveyor role which comprises the required capabilities to perform this role, the role interface, the role-mission, i.e., the set of tasks needed to perform the role and the policies which will activate or deactivate the tasks when triggered by events, to achieve adaptive behaviour. The role-authorisation, specified in terms of authorisation policies, specify how other roles are permitted to interact with this role.

The required capabilities indicate the basic functionality that a UXV must have in order to perform this role. This information is used during role assignment. The role interface is comprised of the management interface, tasks which can be invoked, list of cooperating roles, as well as events generated and received. The management interface is a set of actions which can be invoked on the role to manage role-missions and authorisation policies.

A task will typically cause a UXV to perform an action such as follow a path, search for a container, or start transmitting video so may result in interaction with another role. A task is usually initiated by an obligation policy. Co-operating roles take part in the overall (team) mission. If there are actions the cooperating roles make visible to their team members then those actions are considered part of the role interface. Authorisation policies can be used to selectively permit and deny action invocation by cooperating roles thereby effectively defining which actions from tasks of a role are visible to a cooperating role.

The role-mission can invoke actions from local tasks as well as cooperating-role tasks. The set of all the above actions and the events the role can send and receive give rise to the role interface. Figure 3 shows the details of the role interface.

Relationship between Roles and Capabilities

The capabilities of a UXV indicate its potential in terms of the basic functionality that is expected by the tasks and polices which are loaded on to the UXV. It is analogous to a virtual machine interface available to the policy/task programming level, whereas the role of a UXV defines a specific behaviour within the Mission-SMC in terms of the tasks to be performed and how the role interacts with other roles.

A role specifies the duties of a UXV as part of the Mission-SMC. It effectively defines an interface which specifies how it interacts with other roles in terms of events generated and received and actions which other roles can invoke upon it e.g. to support services it provides to other roles.

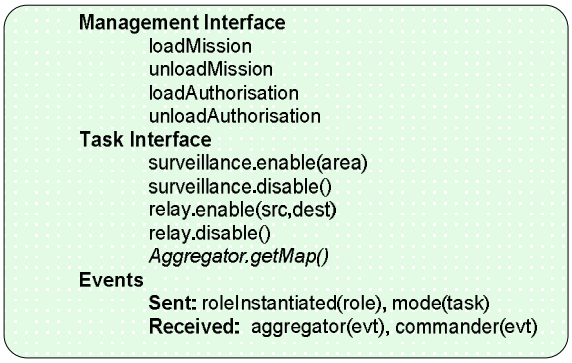

Figure 3: Surveyor Role Interface

A UXV could perform multiple roles at the same time, such as a communication relay function as well as a video search function. A role can be suspended, removed and new roles downloaded at runtime so roles are very dynamic, however the capabilities of a UXV, often relate to specific sensors or devices on the UXV, which do not change frequently so most of the capability is comparatively static, other than properties such as available power, memory, or spare processing capacity.

A role definition specifies the capabilities required to support the role and so only a UXV with the required capabilities should be assigned to the role. Typically assignment to roles takes place when a UXV is discovered or if a UXV fails, another may need to be reassigned to take over, assuming the role mission can be loaded onto it by the commander or from a nearby UXV which has the role mission specification.

Role Assignment and Mission-SMC Formation

A team of UXVs form a Mission-SMC to cooperate to accomplish the mission with each UXV being assigned to one or more

roles within the Mission-SMC. We assume that the Mission-SMC specification is preloaded onto a command vehicle or one of the UXVs which has the capabilities to perform the commander role. Figure 4 shows the main roles for the reconnaissance scenario Mission-SMC. The commander role is responsible for assigning other UXVs to appropriate roles as defined by role assignment policies specified within the Mission-SMC specification (as explained in section 'Mission-SMC Specification').

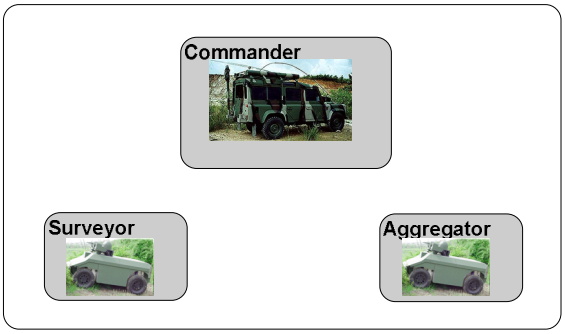

Figure 4: The Mission SMC and Roles

The assumption is that that the UXVs to be assigned to surveyor and aggregator roles may be discovered as they come into radio range of the commander during the course of the mission.

```
1.oblig on discovered ( uxv, credentials, resources)
  do /smc/roles/commander.assign(uxv)
  when authenticate (credentials) and resources.comms = "longRange"

2. oblig on discovered ( uxv, credentials, resources)
   do /smc/roles/surveyor.assign(uxv)
   when authenticate (credentials) and hasCapabilities(motion, video, relay)

3. oblig on discovered (uxv, credentials, resources)
   do /smc/roles/aggregator.assign (uxv)
   when authenticate (credentials) and capabilities.processor > medium
```

Figure 5: Role Assignment Policies

A discovered UXV would first have to be authenticated and its credentials (certificates) verified in order to ensure that the UXV belongs to an allied force. The UXV will then be requested for its capability description and based on its capabilities it will be assigned to a role. For instance in the reconnaissance scenario, if a UXV has the capabilities motion, video and communication relay it would be assigned to a surveyor role and if it has the required processing and storage it would be assigned to an aggregator role.

The policies shown in Figure 5 specify how a newly discovered UXV can be assigned to the appropriate role, after the credentials have been successfully verified, based on the UXV's capabilities. Encoding the role assignment as policies enables us to change the strategy of this assignment during the mission without interrupting its functioning.

Mission-SMC Specification

We specify Mission-SMC in terms of roles using policies in order to enable mission adaptation by utilizing the dynamic nature of policies. As stated previously, the Mission-SMC specification defines a composition of role specifications (see section 'Role').

In this section we will look at the Mission-SMC specification which contains the types of roles required to perform a mission, and a shared knowledge base which might contain certificates, overall-mission constraints etc. For each role type, a role

assignment policy, a role-mission and authorization policies are specified.

An outline of a mission specification in terms of roles and policies is illustrated in Figure 6. The UXV on which the Mission-SMC is formed is responsible for interpreting the overall Mission-SMC specification. That would be the commander UXV in our scenario.

Figure 6: Example Mission-SMC Specification

Prototype Design and Implementation

We have started designing and implementing the policy-based UXV mission-management system. The implementation extends the core SMC implementation developed in the AMUSE project [6] to realise UXV Self Managed Cells (UXV-SMC). A Teleo-reactive based wall following robot task is implemented as a way towards implementing a surveillance mission. Figure 7 shows the architecture of our prototype design and implementation.

A UXV-SMC provides the core functionality on a UXV to allow it to be discovered, authenticated, assigned to roles, interact with other SMCs and to download and execute roles, i.e. it supports role management. In the next section we explain the components of this architecture in detail.

Our research environment is comprised of three Koala robots. The Koala robot [5] is a mobile robot which has 16 infrared proximity sensors, around the body of the robot, and a camera. It has a Motorola 68331, 22MHz onboard processor, a 1Mb ROM and 1Mb RAM. The robot is extensible in that various modules can be added. Our robot has a KoreBot module [5] which has an ARM PXA255 400MHz processor, 64Mb SDRAM, 32Mb Flash and a Wi-Fi card. The KoreBot runs Linux.

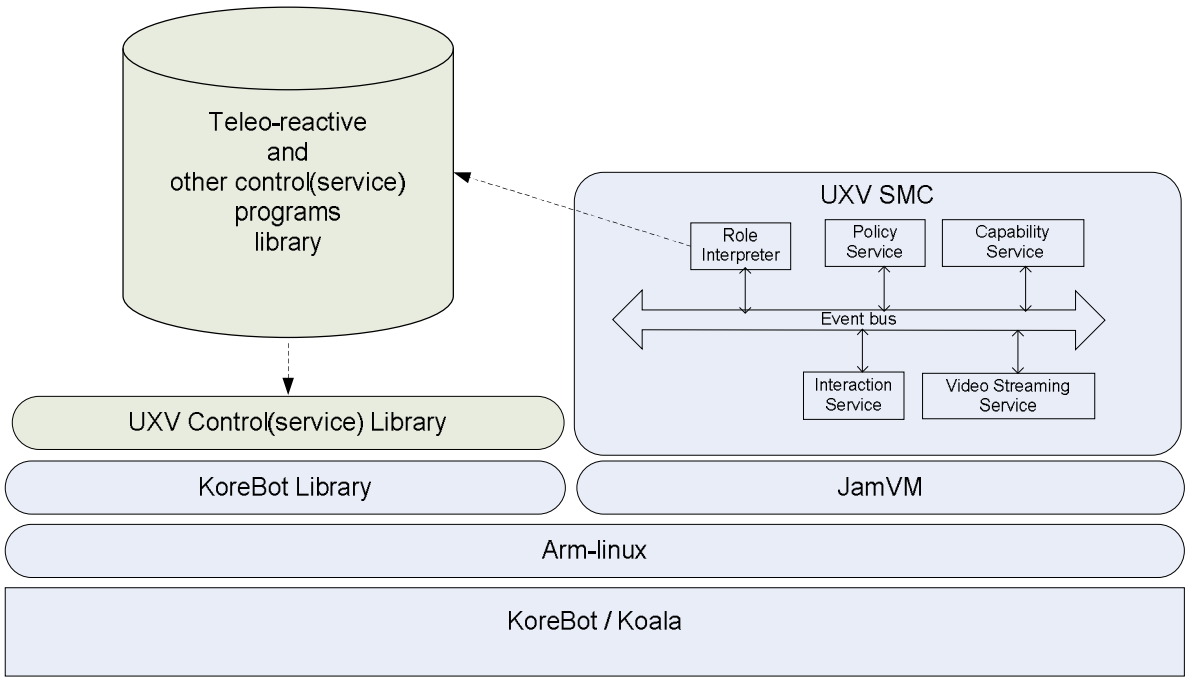

Figure 7: Architecture of UXV Mission Management

The UXV Self Managed Cell (UXV-SMC)

The basic SMC architecture is comprised of an event bus, a policy service and a discovery service [6]. The event bus serves as an asynchronous means of communication between different modules of the SMC. But, not all communications should necessarily pass through the event bus. The policy service interprets event-condition-action policies and it also maintains a domain structure which stores managed objects in a hierarchy. The managed objects are policies, references to role instances and various objects in the SMC. The discovery service is used to discover new components which might join the SMC.

The Self Managed Cell architecture is designed in such a way that it can be extended to support different management components by adding new ones or extending the existing ones. To realise a UXV-SMC we extended the basic SMC by adding two new services called the capability advertisement service and interaction service. In addition we extended the basic discovery service to form a UXV discovery service which is instantiated in the commander UXV's SMC. Figure 8 shows the UXV-SMC.

The role interpreter is responsible for downloading role definitions from a library of definitions and interpreting the role-missions that are contained in the role definition. Typically, it selects the appropriate task from a task library and activates it. This is illustrated in Figure 7. The role interpreter is implemented as an SMC module in that it can interact with other modules through the event bus of the SMC. In addition the role interpreter provides access to the role interpreter of other SMCs in the mission as shown in Figure 8.

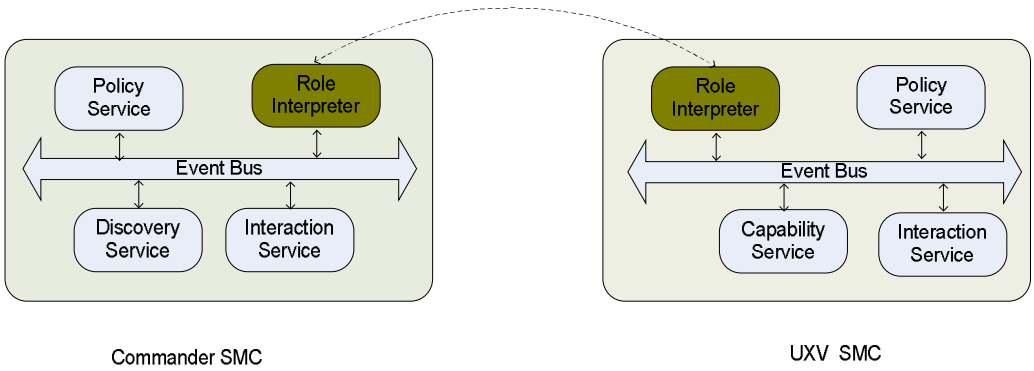

Figure 8: UXV and Commander SMCs

The capability service is created during the SMC initialisation and is responsible for providing the capability description of the UXV when requested. The capability description includes static information such as types of sensors, motion commands, processor type, total physical memory etc. as well as parameters which change over time, but give useful information about current processing capabilities which may be important in role assignment e.g. available battery power, average CPU and memory utilisation. Currently only static information is used in the role assignment process. The use of variable system parameters will be investigated in the future.

The interaction service is the basic means of inter-SMC communication. This service relays events and other messages to and from the commander SMC (Mission-SMC) or among UXV-SMCs. In effect it enables communication among role instances, and the commander. Figure 9 shows the architectural pattern of the interaction service.

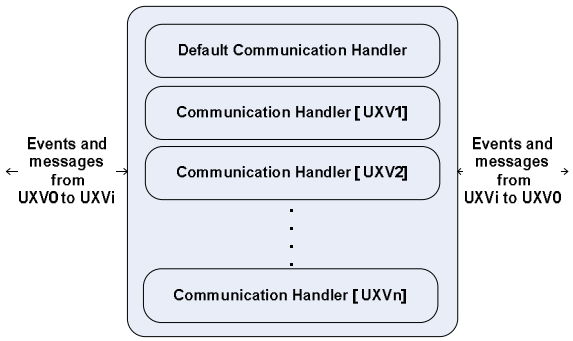

Figure 9: Interaction Service Architectural Pattern

Conceptually, the interaction service keeps a list of communication adaptors to all the SMCs the UXV-SMC communicates with. These adaptors are indexed by the SMC identification for easy access from within the UXV-SMC. Initially the interaction service will have only the default communication handler; this module will be responsible for all communications until the UXV is assigned to some role. When a UXV is assigned to some role it essentially means that the UXV has become part of a Mission-SMC, during this time a communication adaptor is created to glue the UXV-SMC to the Mission-SMC. In essence, the communication adaptor will connect the event bus of the UXV-SMC to the commander-SMC. The events relayed to the Mission-SMC are specified by the role definition. The adaptor also enables message passing between the two SMCs. Should the UXV-SMC need to share its events with another UXV (other than the commander), another communication adaptor is created. This approach allows the UXV-SMC to be a member of more than one Mission-SMC.

The Commander Self Managed Cell

The commander Self Managed Cell (commander-SMC) is similar to the UXV-SMC. The only difference being, the absence of the capability advertisement service in the commander-SMC and that the commander initiates the discovery service whereas the UXV-SMC only

responds to discovery requests but does not try to form new Mission-SMCs.

The discovery service advertises the identification (ID) of the commander UXV periodically. When a new UXV comes in to communication range of the commander it will be able to receive the broadcast. The new UXV then replies to the broadcast if it is willing to take part in a mission.

Upon receiving the reply the commander will request the new UXV for its capability description. The commander then uses the role assignment policies provided by the Mission-SMC specification and the capability description to deicide whether the UXV can support one of the roles. The UXV is informed to which role(s) it has been assigned and if necessary the mission-roles are downloaded. Finally the UXV will instantiate the proxy and the commander will instantiate the adaptor. Figure 10 illustrates the UXV discovery protocol.

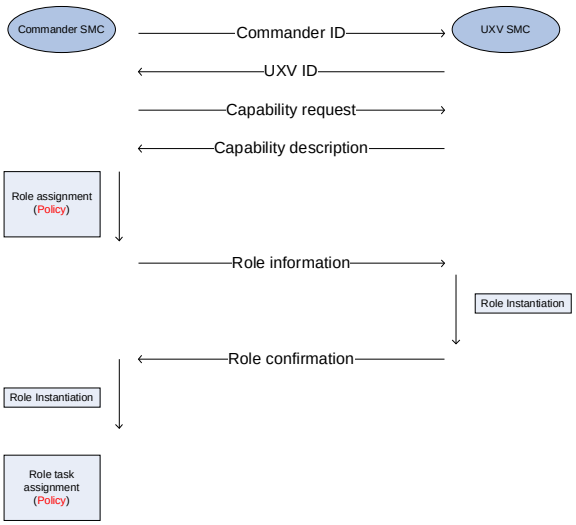

Figure 10: UXV Discovery Protocol

Role Tasks

We use simple robot tasks to test our policy based mission management approach. A teleo-reactive wall following task, which directs the koala robot to follow the wall on its right side, is implemented and tested with the policy based mission management system. An outline of the wall following teleo-reactive program is shown in Figure 11. This task is compiled as a shared library; when a UXV is assigned to a surveyor role, the role proxy loads this task and activates it.

```
all_clear à moveForward()
front_right_obstacle à turnLeft()
front_left_obstacle à turnRight()
right_wall_too_near à moveAwayFromRightWall( )
right_wall_too_far à approachRightWall( )
NOT aligned à align( )
NOT wall_on_the_right à findRightWall( )
```

Figure 11: Teleo-reactive Program for Wall Following

Conclusion

In this paper we have presented a policy based mission specification scheme and mission management architecture for UXVs. These elements will enable UXVs to manage themselves to accomplish a mission. We have also reported our progress in designing and implementing the UXV mission management framework.

The mission management framework is a policy-based way of specifying and assigning missions to be accomplished by a set of UXVs. By using policies and roles to specify missions we enable dynamic mission assignment and adaptation.

Future work will focus on communication and team management as well as refinement and integration of the mission management framework with the rest to produce a UXV self management framework.

References

[1] *Ponder2*. http://ponder2.net/.

[2] M. Dam, G. Karlsson, B. S. Firozabadi, and R. Stadler. *A research agenda for distributed policy-based management.* Report, The Royal Institute of Technology (KTH), 2002.

[3] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. *The Ponder policy specification language.* In Proceedings of Policy 2001, Workshop on Policies for Distributed Systems and Networks, pages pp. 18–39. Springer-Verlag LNCS, 2001.

[4] N. C. Damianou. *A Policy Framework for Management of Distributed Systems.* PhD thesis, Imperial College of Science, Technology and Medicine, February 2002.

[5] *k team.* http://www.k-team.com.

[6] E. Lupu, N. Dulay, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh, and A. Schaffer-Filho. *Amuse: Autonomic Management of Ubiquitous e-Health Systems.* To be published in Concurrency and Computation: Practice and Experience, 2007.

[7] D. C. MacKenzie, R. C. Arkin, and J. M. Cameron. *Multiagent mission specification and execution.* Auton. Robots, 4(1):pp. 29–52, 1997.

[8] J. Sventek, N. Badr, N. Dulay, S. Heeps, E. Lupu, and M. Sloman. *Self-managed cells and their federation.* In Workshop Proceedings of the 17th Conference on Advanced Information Systems Engineering. Springer-Verlag LNCS, 2005.

[9] D. C. Verma. *Simplifying network administration using policy-based management.* IEEE Network, 16(2):pp. 20–26, Mar/Apr 2002.

[10] S. R. White, J. E. Hanson, I. Whalley, D. M. chess, and J. O. Kephart. *An architectural approach to autonomic computing.* In Proceedings of the International Conference on Autonomic Computing, pages pp. 2–9. IEEE Computer Society, May 2004.

Acknowledgments

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.