ARMAdA: Creating a Reflective Fellowship (Options for Interoperability)

Edward Curry
Department of Information Technology
National University of Ireland
Galway, Ireland
edward.curry@nuigalway.ie

Desmond Chambers
Department of Information Technology
National University of Ireland
Galway, Ireland
des.chambers@nuigalway.ie

Gerard Lyons
Department of Information Technology
National University of Ireland
Galway, Ireland
gerard.lyons@nuigalway.ie

ABSTRACT

With the development of numerous adaptive and reflective middleware platforms, inter-platform interoperability is a desirable next step. At present, little or no interoperability is possible at the meta-layer of reflective middleware. The emergence of an open standard for meta-layer interaction is imperative to support the development of next-generation middleware that can express their needs and capabilities to platforms with which they interact. In this paper, we describe the foundations of the ARMAdA interaction standard for adaptive and reflective middleware platforms.

Categories and Subject Descriptors

D.2.12 [**SOFTWARE ENGINEERING**]: Interoperability; D.2.0 [**SOFTWARE ENGINEERING**]: General – Standards; D.2.11 [**SOFTWARE ENGINEERING**]: Software Architectures

General Terms

Management, Standardization.

Keywords

Adaptive and Reflective Middleware, Interaction Standard, Domain Specific Languages.

1. INTRODUCTION

Adaptive and reflective middleware systems have been developed in order to cope with the demands of current and next generation computing environments. Such systems are capable of adapting or self-adapting to meet changing user or environmental requirements. A number of reflective middleware platforms have been developed to provide such capabilities including Open ORB [3], DynamicTAO [1], RAFDA [2], QuO [10], mChaRM [4], etc.

Current research has focused on the reflective capabilities of a system to examine its own internal operation: - internal reflection. In reality, the vast majority of software solutions comprise of a number of interacting propriety implementations. In such environments, internal reflection is not enough; systems need to examine themselves and their interactions with other systems.

Standards and protocols such as IIOP, RMI, and the Web service stack (HTTP, SOAP, UDDI, etc.) have provided a common infrastructure to enable propriety software implementations to interact. This regulated infrastructure facilitates the exchange of information (request, reply, etc.) between implementations. As reflective platforms are deployed in the field they will inevitably encounter and interact with other reflective capable systems. These infrastructure standards enable the base-level of a reflective platform to interact. However, no standard or protocol is available to allow the meta-layers of the platforms to interact.

Interoperability within the meta-layer is one of the key challenges for future reflective platforms. The emergence of an open standard for meta-layer interaction is imperative to support the development of next-generation middleware that can express their needs and capabilities to the platforms with which they interact. Such standards need to define adaptive and reflective capabilities in a neutral format, increasing interoperability across proprietary implementations.

Due to the large number of domains in which adaptive and reflective techniques may be applied, any standard will need to be adequately simplistic to avoid a high cost of entry while remaining sufficiently expressive to describe each domain. With such strenuous demands, one generic standard may not be enough; a number of standards may be required.

It is critical that any such standards be widely accepted by researchers and practitioners in the field; it is therefore important that the creation of any such standard be a communal effort. In this spirit, we see this paper as a catalyst designed to generate discussion and interest in the area with the hope of fostering a cooperative standardisation effort.

1.1 Paper Overview

This paper puts forward the case for meta-layer interaction standards for adaptive and reflective middleware. Section 2 presents a motivational scenario; sections 3 and 4 provide the concept and implementation details of one potential solution. In Section 5 usage scenarios are discussed and section 6 highlights related works. In section 7 an outline of future plans and directions is provided.

2. MOTIVATIONAL SCENARIO

In order to demonstrate the motivation of this work, we present the hypothetical scenario of an online multimedia broadcasting service. This service broadcasts audio streams using an open standard, an MP3 stream, as its distribution format. Utilisation of such an open format enables multiple proprietary media players to seamlessly connect to the service. In this environment, as illustrated in Figure 1, any media client that supports the MP3 format is capable of receiving the multimedia (MP3 stream) broadcasts. With this configuration, clients connect to a MP3 stream of fixed quality.

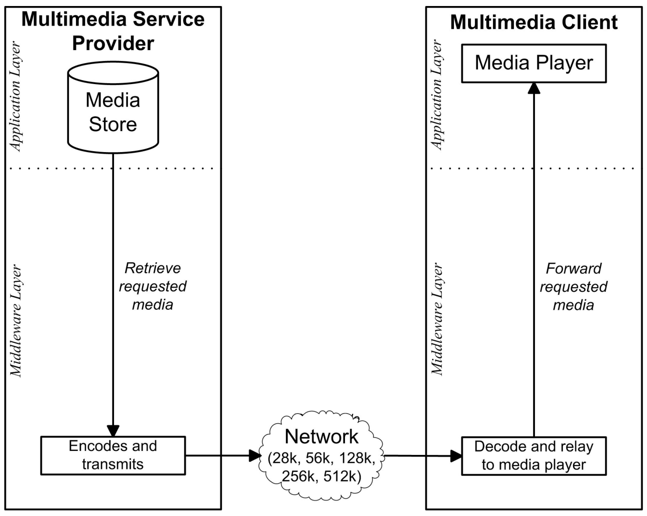

Figure 1: Non-reflective media broadcast service

With the use of current reflective research, (i.e. Open ORB, etc.) it is possible to build such a service with the ability to self-adapt to the client capability (bandwidth, latency, connection reliability etc.). Such techniques improve the Quality-of-Service (QoS) provided by the multimedia broadcasting service by altering the process in which it serves media to its clients. For example, attempting to send a high-quality live audio stream to a client on a low-bandwidth connection will result in a poor QoS for that client. Ideally, the service should recognise the current network capability and transmit a more suitable lower-quality stream to the client. Altering the media encoding and services infrastructure can easily achieve an increased QoS.

As illustrated in Figure 2, current reflective research is proficient at implementing a service with self-adaptive capability. In this reflective broadcast service, it is important to note that the same reflective implementation (i.e. Open ORB) is present in both the server- and client-side middleware stacks. When compared to the stack of the first solution, the reflective capabilities have greatly improved the QoS provided by the service. However, utilisation of such capabilities removes the independence provided by the open MP3 format, resulting in lock-in to a particular propriety reflective implementation. This problem is not unique to Open ORB and would be experienced if any propriety technology were to be used.

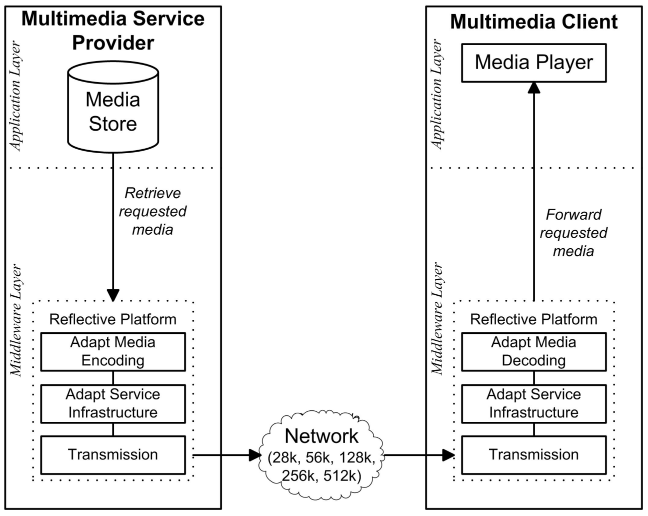

Figure 2: A propriety reflective broadcasts service

Ideally, we would like to keep the adaptive capabilities without the resulting lock-in to any implementation. To this end, we propose the use of a standardized interaction mechanism for adaptive and reflective technologies. Such a standard would decouple the adaptive and reflective capabilities from a given implementation.

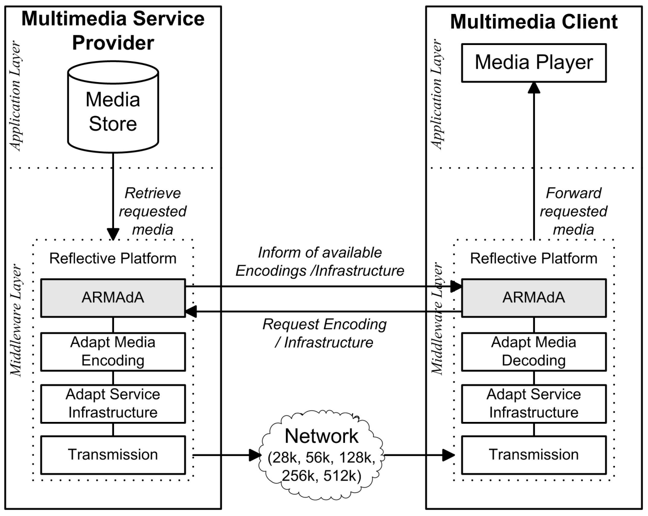

Figure 3: A standardized reflective broadcast service

In Figure 3, a standardized reflective broadcast service is presented. In this middleware stack a new meta-layer interaction protocol is introduced. This layer allows for communications between the server- and client-side meta-layers. When a new client joins, it is able to discover what “capabilities” or “formats” the service can distribute its multimedia in, i.e. what adaptations are available/provided by the service. The client is now able to request the service to deliver the media in a format that best suits the client’s current operating conditions. It could also request additional adaptations if its infrastructure was to change (improve or worsen), for example to alter the quality of the audio encoding to best suit its current network connectivity.

When compared to the previous propriety reflective approach, the service adapts to improve its QoS in the same manner. However, the additional meta-interaction protocol alleviates the problem of propriety lock-in. Any client that supports this standard can benefit from the service's reflective capabilities and request adaptations while maintaining implementation independence.

3. PROPOSED SOLUTION

Critical to the development of interoperable heterogeneous systems is the definition of standards that define the interface and interactions among participating systems. To initiate debate in this area we propose the utilisation of a high-level abstract architecture to facilitate the interaction and cooperation between reflective and adaptive platforms.

Researchers within the Agent community encountered similar issues at an early stage of investigation into open multi-agent based systems. After a number of years the Foundation for Intelligent Physical Agents (FIPA) [6] emerged as a standards body for the agent community. FIPA is an international organisation dedicated to promoting the industry of intelligent agents by openly developing specifications to support interoperability amongst agents and agent-based systems. FIPA defines a number of standards and specifications that include architectures to support inter-agent communication [8], interaction protocols [9] between agents, as well as communication and content languages [7] to express the messages of these interactions. With the use of these standards, any FIPA-compliant platforms, and their agents, are able to seamlessly interact with any other FIPA-complaint platform.

The FIPA model illustrates a community-centric approach to standardization. FIPA serves as an example for a similar effort within the adaptive and reflective middleware community. To initiate discussion on such standards, we purpose the use of a derivative of the FIPA model to enable interoperability among reflective meta-layers.

4. ARMAdA

The Adaptive and Reflective Middleware Abstract Architecture (ARMAdA) is a conceptual view of how meta-layers can interact with one another. Within ARMAdA, each participating entity (platform, system, service, etc.) is part of a community (group). The ARMAdA standard does not attempt to describe how to implement an adaptive or reflective participant, nor does it attempt to specify the internal architecture of a participant. It simply regulates how they interact.

Communication through ARMAdA is based on a model of semantically grounded communication that is pre-defined, semantically rich and well understood by all parties. The basic building block of communication between participants is through the ARMAdA Management Language (AML). This protocol defines a number of generic interaction commands based on the "Request-Reply" paradigm. Each interacting command contains an associated message. The content of this message are expressed in a domain specific language. These languages are responsible for describing the application/domain specific details of the request (i.e. security, hardware, multimedia, telecoms, flight control, education, UI, etc.); these languages may also contain their own interaction commands. Interactions between systems are achieved with a combination of the relevant interaction command with an associated message expressed in a domain specific language.

4.1 Architecture Overview

There is no single definition of what constitutes a participant within ARMADA; participants may be an ORB, event service, transaction service or any middleware platform. The minimum requirements an ARMAdA compliant participant needs is to support the core ARMAdA Interaction Commands, detailed in the next section, and the ability to receive and respond to AML requests from other participants using the relevant domain specific languages. Figure 4 illustrates a potential architecture for such a participant; this architecture is based on a traditional reflective implementation of a base-layer controlled by a meta-layer.

The novel element of this sample architecture is an additional layer on top of the meta-layer. This new layer handles ARMAdA interactions for the participant, acting as a gateway to the platforms meta-layer. The ARMAdA-layer consists of an Inbox, a Language repository, and a description of the capabilities offered by this platform. With these three elements, the participant is able to perform all the basic interaction needed to participate within an ARMAdA community.

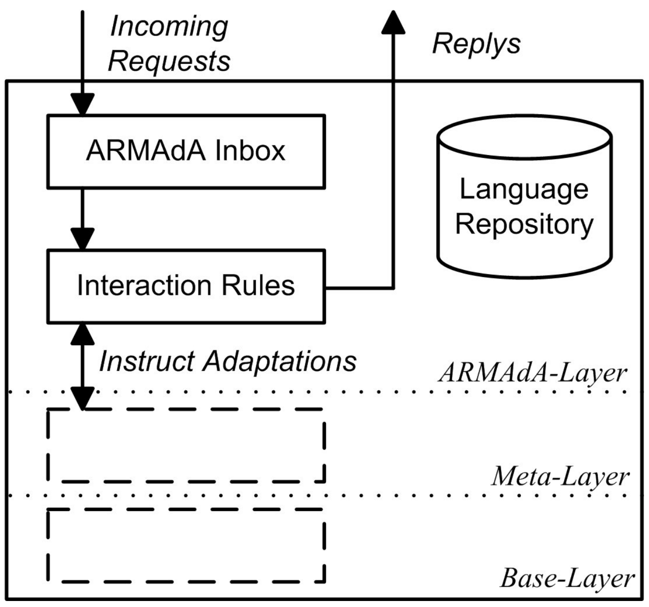

Figure 4: Sample ARMAdA Compliant Architecture

4.2 Interaction Commands

The goal of the Interaction Command (IC) protocol is to describe entire conversations between ARMAdA participants to achieve some action (i.e. an adaptation) or outcome (i.e. resource allocation). ICs provide the context in which the associated messages should be interpreted. The core ICs are illustrated in Figure 5.

There are also general ICs that are not part of the core, such as auctioning, issuing a Call for Proposals or Participation (CFP), negotiation, brokerage services, subscription-

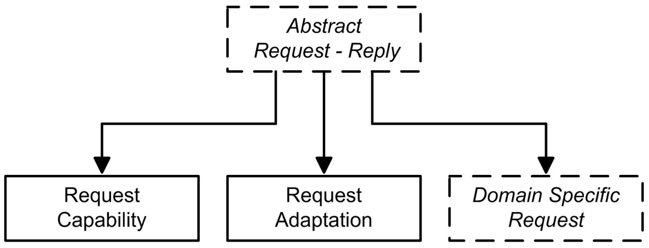

Figure 5: ARMAdA Interaction Commands Hierarchy

based services, etc. In addition, domain specific languages may also define their own specific ICs such as, monitoring, event systems, brokering, multimedia etc.

4.3 Domain Specific Languages

With the use of ARMAdA, two independently developed platforms are able to interact with one another. In order to achieve this communication, ARMAdA uses Domain Specific Languages (DSL) to reach a shared common understanding to enable interoperability between participants. For minimal ARMAdA compliance, platforms are only required to understand one such language, the ARMAdA Management Language. This language is used to bootstrap further domain specific interactions.

4.3.1 ARMAdA Management Language

The ARMAdA Management Language (AML) has two roles. First, it is designed to capture a common understanding of the basic elements that comprise a conversation between two participants. These generic elements can then be reused in other domain specific languages. Secondly, the AML describes a minimal set of interactions needed to initiate a conversation with another ARMAdA participant. A draft AML is detailed in table 1.

Command	Purpose
Request – Reply	Generic request reply command, used as the basis for communication
Request Capabilities	Retrieve a participant's adaptive capabilities and the domain specific languages it uses to describe them
Request Adaptation	Request a participant to perform one of its supported adaptations.

Table 1: ARMAdA Management Language

4.3.2 Domain Specific Languages

Domain Specific Languages (DSL) are used to describe possible adaptations within a specific environment The division of languages allows highly specialised languages to be defined to describe specific adaptations within a domain. This allows ARMAdA to be very comprehensive in its descriptions of environments without resulting in a bloated single language; such an approach would result in a high cost of entry to the community by requiring conformance to a large specification. The use of multiple DSLs reduces the cost of entry to the relevant DSLs that describe the domain and its adaptive capabilities. DSLs can be defined for a number of areas such as network connectivity, security, transactions, notification services, etc. Table 2 details a partial DSL for multimedia platforms.

Name	Purpose	Contents
Media Type	Type of media available	Audio Only / Video Only / Audio and Video
Encoding Format	Formats media can be encoded into	Media Format (aac, avi, mp3, mpg, ram, ogg, wav, wma, etc)
Bit Rate	Quality of the encoding	Bit Rate (28kbps, 56kbps, 128kbps, 256kbps, 512kbps)
Delivery Mechanism	Mechanism of deliver	Streamed / Download

Table 2: Sample Multimedia Domain Specific Language

With the use of the above multimedia language it is possible for a platform to describe the possible adaptations it can perform; a sample description is provided in Table 3.

Name	Settings
Media Type	Audio Only
Encoding Format	aac, mp3, ram, ogg, wav, wma
Bit Rate	28kbps, 56kbps, 128kbps, 256kbps, 512kbps
Delivery Mechanism	Streamed, Download

Table 3: Sample Multimedia Service Definition

This service definition details possible adaptations within a service. With this information, we can request changes to the format/compression and delivery mechanism of a particular audio source provided by the service. With the definition provided in table 3, we can request audio to be sent as a 128kbps MP3 encoded stream.

4.4 Walkthrough

To fully illustrate the interaction mechanism proposed within ARMAdA we provide a systematic walkthrough of a conversation between two participants.

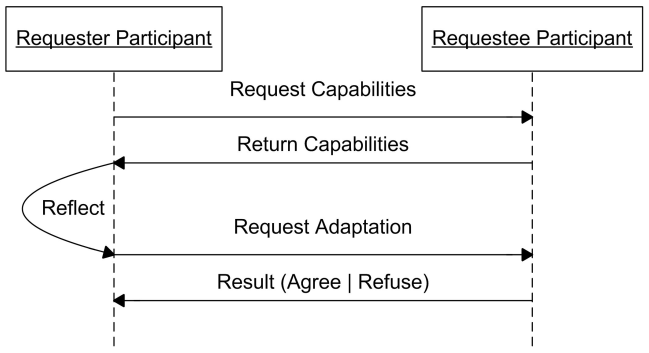

Figure 6: ARMAdA interaction sequence

- ***Discovery*** – there is no restriction on participant discovery within ARMAdA. Participants can be discovered via a directory service, user prompting, system configuration or from other participants.

- ***Capability Handshake*** – post discovery, the first act is to exchange capability information using the *Request Capabilities* command. This command will return a list of available adaptations described in a relevant domain specific language.

- ***Adaptation Requests*** – participants can now use the *Request Adaptation* command to request any of the supported adaptation.

5. ARMAdA USAGE SCENARIOS

With an ARMAdA-like standard in place, a new "participant" in the community can see what adaptations are available from other participants within the community. With this information, it can request adaptations from participants it interacts with to best suit its current situation.

5.1 Security Example

In order to illustrate the generality of the ARMAdA approach an additional usage scenario from the security domain is provided. In this example, we have created a minimalist DSL to describe a basic security domain for user authentication and encryption protocols. For clarity sake we assume a public key infrastructure is in place. The Security DSL is described in Table 4.

Name	**Purpose**	**Contents**
Authentication Protocol	Authentication options	RADIUS / S/Key / TACACS / CHAP / Kerberos / IPSec (AH) / EAP
Hashing Algorithm	Optional hashing algorithm used with authentication protocol	MD5, SHA-1, etc
Encryption Protocol	Protocol used for encryption	CIPE / SSL / SHTTP / SSH / SSH2 / IPSec
Encryption Cipher	Cipher used with encryption protocol	Triple DES / RC4 / RC5 / AES / IDEA

Table 4: Sample Security Domain Specific Language

The security DSL allows a platform to describe the security protocols it supports, a sample description is provided in Table 5.

Name	**Settings**
Authentication Protocol	Kerberos / IPSec (AH) / EAP
Hashing Algorithm	MD5, SHA-1
Encryption Protocol	CIPE / SSL / SHTTP / SSH / SSH2 / IPSec
Encryption Cipher	Triple DES / RC4 / RC5 / AES

Table 5: Sample Security Service Definition

In this description we can request authentication to be achieved using the Extensible Authentication Protocol (EAP) with a SHA-1 hash, and for information to be encrypted using Advanced Encryption Standard (AES) and sent using the SSH encryption protocol. DSL's can be used in conjunction with one another to describe different aspects of a service, for example the security DSL could be used with the multimedia DSL to describe a secure multimedia broadcast service.

5.2 Relationships

Within ARMAdA, participants are able to reflect on their current requirements and request adaptations from others. However, it is important to note that the requestee is under no obligation to carry out the request, they can refuse to adapt: - they have a choice. With no standard definition of relationships, they can refuse to adapt, accept the adaptation or promise to perform a best effort. A number of communities with defined participant relationships may also be set-up:

- Cooperatives – to achieve mutual goals
- Market Places
- Master-Slaves
- Commons – deregulated control of common resources

Such environments are heavily influenced by group dynamics. From a reflective perspective, community or group reflection provides a number of interesting areas for investigation, where participants desire to co-operate but may not have mutual goals; co-operation and game theory may be applied to the community.

Participant relationships may also extend beyond simple adaptation requests to the sharing of information between participants. When a new member arrives into an environment, it may request an individual or group of participants to inform it of their history within the environment. This history sharing could enable the service to predict possible future requirements or patterns within the environments, enabling it to rapidly gain experience of its new environment. Such arrangements could also be extended to exchange real-time environmental information; this can reduce the overhead and burden of reflection by normalising and sharing common monitoring activities between participants.

5.3 Reflection^2 (Reflecting on Reflection)

Within ARMAdA, participants inform the community of their capabilities and maintain control over the capabilities they advertise. This enables a participant to easily change their adaptation offerings to best suit their current run-time operating conditions. Such ability is of particular use during peak-usage periods. As a case in point, a news broadcasting service may offer to stream its news bulletins in a number of different qualities 28kbps, 56kbps, 128kbps, 256kbps, and 512kbps. During off-peak-hours the service may offer its broadcasts in all of these rates, however during its peak-times when demand is greatest (morning, lunchtime, and evening news bulletins) it may only offer the lower rates of 28kbps, 56kbps, and 128kbps in order to improve its scalability and the overall QoS it delivers. This "limitation" of its adaptive capabilities is easily achieved by altering its capability advertisements. ARMAdA participants are able to subscribe to these advertisements, using a subscription-based IC, to maintain an accurate description of currently available adaptive capabilities.

6. RELATED WORKS

Numerous communities have encountered the issues covered in this paper over the years. As previously highlighted, the agent community developed the FIPA standards to enable interoperability between agents. The networking community developed standards such as MIB / SNMP for network management. These standards have made network devices easier to control through a common administrative view.

Similar management standards have also reached the software management domain with efforts such as Java Management Extensions (JMX). JMX enables the integration of Java application into existing network management solutions, simplifying the management of software applications. The vision of GRID computing has been made possible thanks to the development of the Open Grid Services Architecture (OGSA) which defines the standard interfaces and behaviours of a Grid service, building on a Web services base.

Other domain also face the challenges that we currently face, for example the vision of autonomic computing will involve mass standardization for interoperability on almost every level of software and hardware.

7. FUTURE PLANS

Future work in this area falls along two lines, development of standards, and research opportunities that arise from the development of such standards.

7.1 Standards Development

In order to develop the concept of ARMAdA or any standardization any further, there is a need for consensus. If any standardization effort is to be successful, the adaptive and reflective community need to participate in the formation of an international group to develop such standards in an open collaborative environment. Any such group should consist of a mixed representation of international companies and academic institutions in order to reach a balanced common specification. Standards developed by such a group would benefit from the risk reduction afforded by an internationally coordinated and managed effort. This reduces the risk of individual research efforts by creating useable standards that are implemented and supported by others.

A number of standards will need to be developed by this group, including standardisations for ARMAdA message encoding formats, transportation protocols, DSL message formats and the definition of DSLs and relevant ICs for each domain.

7.2 Research Opportunities

One of the most interesting research challenges with open communities is in the area of co-operation and co-ordination within the community. Participants will need to deal with both friendly (mutually beneficial) and hostile (competitive) environments.

Middleware platforms may provide different levels of services depending on environmental conditions and resource availability and costs. John Donne said 'No man is an island' likewise no adaptive or reflective middleware platform, service or component is an island and each must be aware of both the individual consequences and group consequences of its actions [5]. Next-generation middleware systems must coordinate/trade with each other in order to maximize the available resources to meet the system requirements.

Such challenges present a number of interesting research areas such as negotiation-based adaptation, resource definition (capabilities and an assurance of the Quality-of-Service), and resource trading. Resources may be traded with simple barter between two participants or complex auctions with multiple participants, each with their own tradable resource budget.

8. CONCLUSION

Interoperability between adaptive and reflective platforms is an important next step for our research community. An international consensus is needed on the interfaces and protocols used to interact with these platforms. In this paper, we present a potential candidate to standardize such interoperability. Our solution, ARMAdA, is an approach based on the notion of a minimal set of interaction commands to enable the enquiry of adaptive capability offered and to request adaptations. Adaptive capabilities are defined in domain specific languages. These languages allow highly specialized descriptions to be created in order to define each domain's relevant adaptations.

9. ACKNOWLEDGMENTS

The authors would like to thank Enda Ridge for his improvements to the readability of the paper.
The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

10. REFERENCES

[1] *Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB*, New York, 2002.

[2] *A reflective approach to providing flexibility in application distribution*, Rio de Janeiro, Brazil, 2003. Springer-Verlag Heidelberg.

[3] G. S. Blair, G. Coulson, A. Andersen, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N. Parlavantzas, and K. Saikoski. The Design and Implementation of Open ORB 2. *IEEE Distributed Systems Online*, 2(6), 2001.

[4] W. Cazzola and M. Ancona. mChaRM: a Reflective Middleware for Communication-Based Reflection. *IEEE Distributed System On-Line*, 3(2), 2002.

[5] E. Curry. Adaptive and Reflective Middleware. In Q. H. Mahmoud, editor, *Middleware for Communications*, pages 29–52. John Wiley and Sons, Chichester, England, 2004.

[6] FIPA. FIPA - Foundation for Intelligent Physical Agents.

[7] FIPA. RDF Content Language Specification, 2001.

[8] FIPA. Agent Message Transport Service Specification, 2002.

[9] FIPA. Request Interaction Protocol Specification, 2002.

[10] I. M. W. o. R. Middleware, editor. *The Quality Objects (QuO) Middleware Framework*, New York, USA, 2000.