The Collective: A Common Information Service for Self-Managed Middleware

Edward Curry
The Department of Information Technology
The National University of Ireland, Galway
Ireland
EdCurry@acm.org

Enda Ridge
The Department of Computer Science
The University of York
United Kingdom
ERidge@cs.york.ac.uk

ABSTRACT

As the deployment of self-managed reflective middleware platforms increases, the process of collecting and examining information used within the reflective process becomes ever more complex. The quality of such information is vital to ensure the successful outcome of the self-management process. However, the cost associated with the collection of this information plays a major role in influencing the success of a self-managed system.

Within typical deployment environments it is not uncommon for multiple self-managed systems to be deployed, each collecting information for use within their respective reflective computations. In many cases, these systems will collect the same information, replicating the effort required to retrieve the information. Such replication could be avoided by sharing information between systems to reduce the overall cost of collection within the deployment environments.

Current self-managed systems lack adequate support for information collection and sharing. This work proposes the use of an independent information service to assist in the collection and management of information within self-managed middleware systems.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; D.2.11 [**Software Engineering**]: Software Architectures

Keywords

Information Management, Self-management, Adaptive and Reflective Middleware

Keywords

1. INTRODUCTION

Within self-managed systems, one of the most important factors that influences a system's ability to self-manage is the quality of information available to the reflective process. A number of information sources may be used within reflective computational logic to analyse the current operating environment including resource utilisation, performance metrics and usage patterns. The sources that produce such information are wide-ranging from performance interceptors to Quality-of-Service (QoS) feedback from clients.

In general, the higher the quality of information given to the reflective process, the greater the chance of a successful self-management action. However, the task of collecting this information is one of the more costly activities within the self-management process. As systems increase in scale, the quantity of potential information sources will also increase, further complicating the task of information retrieval.

Given the increase in the range and quantity of information sources, a need exists to provide support mechanisms to assist in the management and analysis of such information within the reflective process and to minimise the cost associated with its collection. This work investigates the use of an independent common information service to support the collection and dispersion of information within groups of self-managed middleware systems.

1.1 Paper Overview

This paper puts forward the case for an independent service to assist in the collection, aggregation, maintenance, and dispersion of heterogeneous information sources within self-managed middleware. Section 2 presents a motivational scenario; Section 3 provides the purposed design for an independent information service. Section 4 highlights related works. In Section 5 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 two self-managed middleware services that use similar information within their reflective computations. The two services in question are:

- **A Multimedia Service:** used for the provision of multimedia content (such as streamed video and audio)
- **A Video Conferencing Service:** provides real-time audio and video between multiple locations.

Both of these services can be deployed on a variety of networks with diverse operating conditions such as an office

LAN, office wireless network, GPRS mobile connection or home dial-up connection. In order to provide a suitable QoS over this range of network connections, these services can be enhanced with self-management techniques found within reflective platform such as OpenORB [1]. These techniques provide adaptive capabilities to alter the services infrastructure to provide suitable connectivity within each of these environments.

The *local* and *remote* information collection processes of both of these services will now be examined.

2.1 Local Information Collection

Within the reflective process of both of these services, similar information may be used to assess the QoS received by a client of the service. This information includes network activity (connection latency, reliability and bandwidth), user feedback and performance metrics. Based on this information an appropriate infrastructure will be chosen to provide the best QoS for that client.

An example deployment scenario using current self-managed reflective techniques is provided in Figure 1.

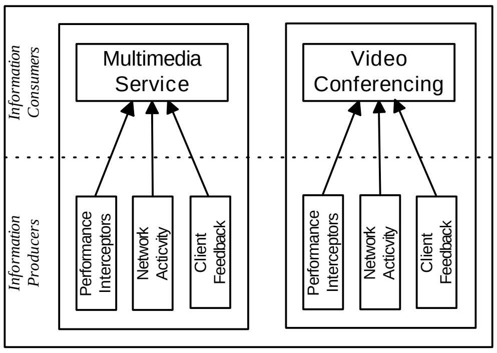

Figure 1: Current Information Collection Process

Using current techniques, both of these self-managed services gather information for QoS assessment independently. The effort required to collect information that is common to both of these services is replicated by each service; resulting in unnecessary and costly overheads.

2.2 Remote Information Collection

The example scenario presented within Figure 1, is a simple straightforward deployment environment. However, within real world deployments, self-managed middleware platforms encounter considerably more complex environments with greater diversity of information sources distributed over larger scales. Within such environments, middleware platforms are often deployed in interconnected groups to service the environment, as illustrated in Figure 2. Each Broker or node within this deployment could contain multiple local middleware platforms as illustrated in Figure 1.

When examining the role of the information gathering and sharing within such deployments, similar issues are shared with the collection of information as expressed within the node-level local information collection process. However, the level of overhead within these environments is much greater as multiple systems may replicate similar information collection tasks.

The key obstacle here is the effective collection and dispersion of information over large distributed deployments.

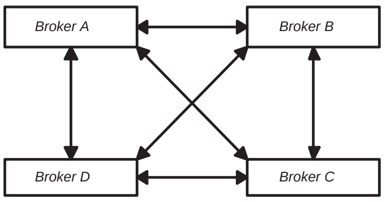

Figure 2: Sharing Information between Brokers

Current self-managed systems do not provide adequate support for information collection and dispersion. Providing an effective solution to these problems is a key challenge to gain the maximum benefits from self-managed middleware platforms.

3. THE COLLECTIVE: A COMMON INFORMATION SERVICE

Ideally, common information should only be collected once to minimise the cost of the reflective process (Equation 1) and so maximise the potential benefits of using self-management. This can be achieved with the use of an independent information service that is responsible for the collection and distribution of such information.

$$\begin{aligned}\text{Cost} = {} & \text{Monitoring Cost} \\ & +\text{Reflective Computation Cost} \\ & +\text{Adaptation Cost}\end{aligned} \tag{1}$$

The Collective is an independent information service for self-managed middleware platforms. The service provides an effective support service for the collection and management of reflective information, minimising the cost associated with these activities. The Collective is designed to reduce the effort needed to collect information from an environment and to allow this information to be shared between reflective systems.

Given the role of The Collective, it has been designed as an independent entity within the middleware stack that is easily accessible by multiple services. The Collective is introduced into the motivational deployment within Figure 3.

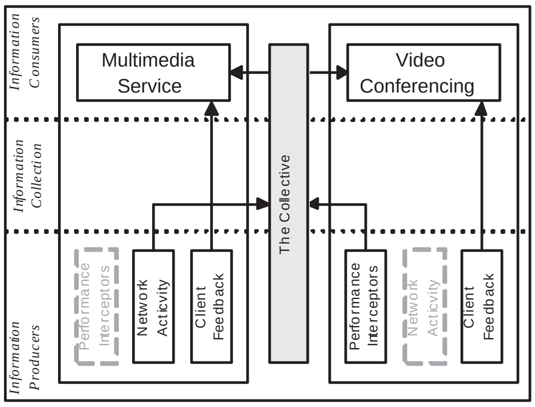

Figure 3: The Collective within the Information Collection Process

By relocating the task of information collection to an independent service, the effort required to gather information is minimized. Information collection tasks that are common to both platforms are assimilated into The Collective. This reduces the effort required to implement self-managed systems, as the duplication of these common information services is no longer required.

It should be noted that not all tasks are appropriate for common collection and sharing, this point is illustrated within the scenario using client feedback monitoring; such information is often highly specialized to the relevant middleware service.

With the role of the service defined, the next issue is to identify the mechanism used by the service for the effective collection of information.

3.1 Overview

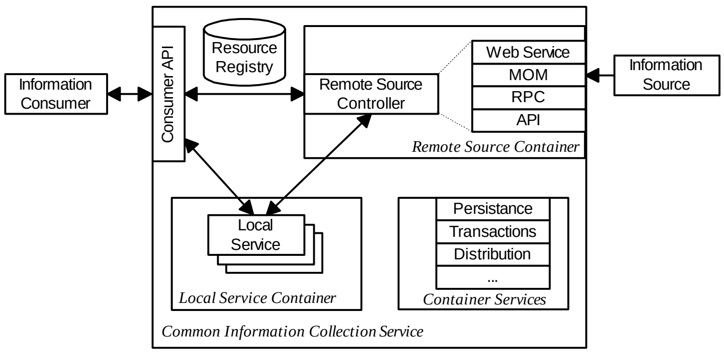

Figure 4: The Collective Architecture

As illustrated in Figure 4, the internal architecture of The Collective is broken into the following parts:

- Resource Registry – Listing of available information sources
- Local Service Container – Information collection and analysis entities located within the platform
- Remote Source Container - Offers facilities to hook-in external and legacy information sources to the platform
- Container Services – Support services for both local and remote containers within The Collective. Services include persistence, transaction and distribution.

3.2 Information Services

The Collective can be used to achieve a number of information requirements within a self-managed middleware platform and is designed to allow the creation of new information services and to provide support for legacy information sources. In order to support these capabilities two types of information service are provided within The Collective, *local* information services and *remote* information sources. It is envisaged service administration may be directed via a configuration file or runtime API.

3.2.1 Local Information Service

Local information services are primarily executed locally within The Collective platform. Local information services provide a platform for the construction of information-only collection services. This opens the possibility for the development of a wide variety of information services that can assist within the self-management process. It should be noted that 'local' refers to the location of the service execution with respect to The Collective, the information collected is not limited to local sources.

One interesting use for a local information service is as a mechanism to perform operations on other information sources within the platform. Such actions could include information source aggregation, information formatting, or analysis. By locating such activities within The Collective, the cost of common information processing tasks may also be reduced.

3.2.2 Remote Information Sources

Remote Information Sources provide a mechanism to hook-in information produced from remote locations or from legacy information collection within current reflective platforms. Remote information sources may interact with The Collective in a number of manners, from a direct API call to an asynchronous Message-Oriented Middleware (MOM) or Web Service interaction. Flexibility within the interaction mechanism is vital to allow diverse information sources to interact successfully and seamlessly with The Collective.

Remote Information Sources comply with a simple service interface to allow The Collective to pull information from the remote source or for the source to push information to The Collective. Each remote information source has a controller present on The Collective platform that can be used to access the information source. The controller can also enhance the information source with post collection operations such as information formatting or analysis activities. The controller may also utilize container services to improve the QoS of the information source such as using the local persistence service as a cache to improve the responsiveness of the remote source.

It should be noted that contributing information to The Collective does not need to affect the current role of a legacy information source; the source should be able to continue to supply its current consumers directly and not interfere with the operation of the donating platform.

3.3 Information Consumer Interaction

Information consumers retrieve information from the service using the *Information Consumer API.* This API will allow information to be retrieved from the service in both a synchronous and asynchronous manner. The basic information retrieval operations needed from the API are described in Table 1.

In addition to these operations an information selection mechanism is required to allow the information consumer to be selective about the information it receives from a service. A filtering capability is vital to avoid information overload for consumers. Within the Message-Oriented Middleware domain the Java Message Service (JMS) specification [10] uses message selectors, a form of attributed-based filtering, to filter the messages received by a message consumer. The message consumer will only receive messages whose headers and properties match the selector. Message selectors consist of a string expression based on a subset of the SQL-92 conditional expression syntax.

A similar techniques could be utilised within The Collective, however, in order such an approach to be useful a

Action	Description
RETRIEVE (BLOCKING)	Retrieve information from the service. If no information is currently available, the call will block until information is available.
RETRIEVE (NON-BLOCKING POLL)	Retrieve information from the service. If no information is available, do not block.
LISTEN (NOTIFY)	Allows the information service to inform the client of the arrival of a information using a call-back function on the client. The call-back function is executed when new information arrives.

Table 1: Information Consumer API

standardised structures for information available within The Collective would need to be developed. The five generic message types within the JMS are currently under consideration for this role.

Additional issues the consumer API will need to deal with include controlling the rate of information flow from services to consumers and the Quality of Service (QoS) related to the information flow.

3.4 Deployment Options

The Collective can be deployed in a number of fashions. The most straightforward deployment is as a standalone service to manage information consumers and producers within self-managed systems. Within such a deployment, as illustrated in Figure 5, The Collective acts as a medium to exchange information between platforms providing a single location to send and receive information.

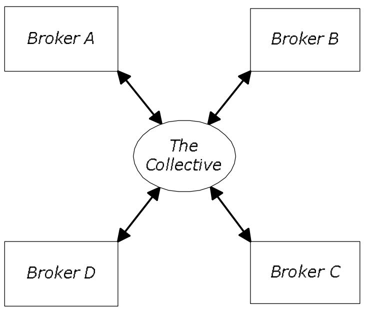

Figure 5: A The Collective within a Broker Network

While The Collective is useful within such an environment, the true power of such services are revealed once the deployment environment scales to multiple physical sites each containing large collections of self-managed systems. A key benefit of the service is its capability to interconnect to share information between multiple services and locations. Within such environments, The Collective may be interconnected to provide a streamlined exchange of information between inter-site brokers, self-managed systems at each site can retrieve information from sources at each site within the deployment. This is illustrated in Figure 6.

A Collective network can be connected in a hierarchical and peer-to-peer topology to meet the needs of the deployment environment, allowing the creation of information networks to exchange information relevant to the self-management process.

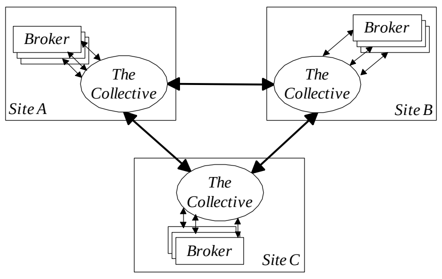

Figure 6: A Collective Network

4. RELATED WORKS

While no self-managed or reflective platform claims to possess an independent information service, a number of projects do possess comprehensive information capabilities and represent and manage it in a variety of manners.

K-Components [4] is a component framework designed to support autonomic components with the use of reflective techniques. K-Components is one of the first platforms to investigate the coordination of decentralized reflective systems with the use of Collaborative Reinforcement Learning (CRL). K-Components also facilitates communication between decentralized runtime deployments by exploiting the event-communication paradigm.

The sharing of information between K-Component runtimes is possible by subscribing to feedback events from remote runtimes; these are known as remote feedback events [3]. Remote feedback events demonstrate a basic ability to share information between systems. However, the type of information that can be exchanged between K-Component runtimes is limited to feedback events and the transfer of known component definitions with no support available to extend this mechanism.

OpenORB 2 [1] is an adaptive and dynamically reconfigurable Object Request Broker (ORB) supporting applications with dynamic requirements. Open ORB utilizes a resource meta-model to allow access to underlying system resources, including memory and threads, via resource abstraction, resource factories and resource managers [1]. This model provides control and accounting facilities for a resources to simplify QoS management [12]. OpenORB does not provide any support for sharing information within its meta-level with external entities.

Platforms such as OpenORB provide a good example of systems that could benefit from integration with The Collective. The resource meta-model could provide a promising candidate for a remote information source within The Collective, allowing its information to be shared with other self-managed systems that share its environment.

The Quality Objects (QuO) framework [15] is designed to assist in the development of QoS stringent distributed systems. Within QuO “System condition objects (sysconds) provide interfaces to monitor and control low-level details of the system.” [5]. SysCond objects can expose the QoS state in a number of ways from simple values probes to periodical polls and sliding window counters. Once an QuO application is running its runtime kernel is responsible for the coordination and evaluation of contracts and monitoring of SysCond objects. SysCond objects provide an example of shared information between two reflective systems, in this case the client and servant for QuO. While not a first class information service in themselves, SysCond objects do illustrate the potential for sharing information within an environment. In a similar fashion to the OpenORB resource meta-model, the information from certain SysCond objects may also be a candidate for integration with The Collective.

The Simple Network Management Protocol (SNMP) is a straightforward protocol used to interact with the management aspects of networked devices (bridges, hubs, routers, etc). Many similarities exist between the role of SNMP for network management and the role of The Collective for information management within self-managed systems. In a similar fashion to The Collective's separation of information infrastructure from self-managed, SNMP also separates management activities from the architecture of network devices.

SNMP is commonly used in conjunction with a Network Management System (NMS). The goal of a these systems is to provide a single point of monitoring and administration of all SNMP enabled devices, often providing a console through which the network administrator performs network management functions. The consolidation of network management activities achieved by NMS and SNMP is analogous to the goal of self-management information consolidation of The Collective.

5. FUTURE PLANS

Immediate plans for this research focus on the implementation and evaluation of The Collective information service. A number of research opportunities also exist with the development of common information services at both the *infrastructure* and *application* levels. Future research directions within both of these areas are now briefly discussed.

5.1 Empirical Evaluation

Once the implementation of The Collective is complete, an extensive evaluation process is needed to verify and validate the benefits of an independent information collection service when compared to their traditional alternative. Benchmarks within the empirical evaluation must be performed under conditions as close to a production environment as possible. This is vital to find out the actual savings achieved with the use of a common information service.

5.2 Infrastructure Standardisation

In order for services like The Collective to be universally accessible, a standard mechanism is needed to govern interaction with the service. Service interaction can be broken down into two parts, the *interface* used to interact with the service and the *semantics* of the information exchanged through the service. Standardisation of interfaces and semantics is needed to achieve streamlined accessibility by information consumers.

5.2.1 Interface

Definition of a standardised consumer API is an important step in the development of a universal information service. A standardised interface similar to the Java Message Service Specification [10] or JDBC Specification [6] would streamline the integration process, allowing information consumers to easily connect to multiple information sources in a consistent manner. It is important that any interface does not constrain potential information sources or limit their inclusion within The Collective.

5.2.2 Semantics

The second obstacle with a standardization effort is the need to reconcile semantic differences between diverse information producers and consumers. Successfully integrating two systems requires that the semantics of both systems must be reconciled or bridged. Within a large-scale heterogeneous integration effort, this is one of the greatest challenges faced. In order to achieve a successful integration, all participants must have a common conceptualisation of the problem domain.

Semantics have an affect on a number of areas of application development. Most prominently from the perspective of this work, they effect the definition of information within the service. In order for a common information service to work, it is vital that all participants understand what the information means and how it is expressed. One promising approach for this problem is the use of ontologies and semantic technologies such as Resource Description Framework (RDF) [8] or OWL Web Ontology Language [11].

5.3 Information Services

With the infrastructure in place, a great deal of potential exists for the development of application-level information services to improve the quality and reduce the cost of information available to self-managed systems. Two such services are highlighted to illustrate the research potential within this area.

5.3.1 Global Information Services

Global Information, one that relates to the entire deployment environment, is an area of particular interest within self-managed middleware platforms [9]. Limited strategies exist for the collection, analysis and management of information that provides a global perspective a wide-scale deployment.

The Collective allows for the exchange of information between middleware platforms within a large-scale deployment. However, many open research issues exist with the handling of this information. How can this information be collected and presented in a simple and straightforward manner?

Pheromone trails within natural environments, such as ant and bee colonies, have been successfully transferred to provide a simple and effective communication between large numbers of participants within decentralised environments [14]. To this end, work is underway on the development of a nature-inspired pheromone-based information service to provide an effective mechanism to gather and compose global information in a simple concise format.

5.3.2 Publish / Subscribe Services

The publish/subscribe messaging model is a very powerful mechanism used to disseminate information between anonymous message consumers and producers. This one-to-many and many-to-many distribution mechanism allows a single producer to send a message to one user or potentially hundreds of thousands of consumers. A publish subscribe service routes the messages to consumers based on the topics to which they have subscribed as being interested in.

Given the increasing amount of information available to self-managed systems, publish/subscribe offers a powerful mechanism for the dissemination of information between information sources and information consumers. Integration of a publish/subscribe engine [7], [2], [13] within The Collective would not only improve the capability of the service to provide consumers with relevant information, but also provide an effective mechanism to interconnect Collective networks as discussed in Section 3.4.

6. CONCLUSION

The quality of information available to a self-managed system is a vital factor in ensuring successful self-management activities. However, a significant cost associated with self-management is the collection of such information.

Current self-managed systems do not provide adequate support for information collection and sharing. Given the broad range of available information sources and their increasing quantity, a need exists to provide support mechanisms to assist in the management and analysis of information within the self-managed systems.

This work proposes the use of The Collective, a common independent information service to assist in the collection, aggregation, maintenance, and dispersion of heterogeneous information sources within self-managed middleware, minimising the cost associated with these tasks.

Common information collection tasks within self-managed systems are assimilated into The Collective, reducing the effort required to implement self-managed systems by removing the need to duplicate common information infrastructure.

7. ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their constructive comments

8. REFERENCES

[1] 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.

[2] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. Phd, Politecnico di Milano, 1998.

[3] J. Dowling. *The Decentralised Coordination of Self-Adaptive Components for Autonomic Distributed Systems*. Phd, University of Dublin, Trinity College, 2004.

[4] J. Dowling and V. Cahill. The K-Component Architecture Meta-model for Self-Adaptive Software. In A. Yonezawa and S. Matsuoka, editors, *Metalevel Architectures and Separation of Crosscutting Concerns, Third International Conference, Reflection 2001*, volume 2192 of *Lecture Notes in Computer Science*, pages 81–88. Springer, Kyoto, Japan, 2001.

[5] G. Duzan, J. Loyall, R. Schantz, R. Shapiro, and J. Zinky. Building adaptive distributed applications with middleware and aspects. In *3rd international conference on Aspect-oriented software development*, pages 66 – 73. Lancaster, UK, 2004.

[6] J. Ellis, L. Ho, and M. Fisher. JDBC 3.0 Specification, 2001.

[7] L. Fiege, F. C. Grtner, O. Kasten, and A. Zeidler. Supporting Mobility in Content-Based Publish/Subscribe Middleware. In M. Endler and D. C. Schmidt, editors, *ACM/IFIP/USENIX International Middleware Conference (Middleware 2003)*, volume 2672 of *Lecture Notes in Computer Science*, pages 103–122. Springer, Rio de Janeiro, Brazil, 2003.

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

[9] K. Geihs. Middleware Challenges Ahead. *IEEE Computer*, 34(6), 2001.

[10] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout. Java Message Service Specification v1.1, 2002.

[11] D. L. McGuinness and F. v. Harmelen. OWL Web Ontology Language Overview, 2004.

[12] N. Parlavantzas, G. Blair, and G. Coulson. A Resource Adaptation Framework for Reflective Middleware. In *2nd Workshop on Reflective and Adaptive Middleware, Middleware 2003*. Springer-Verlag Heidelberg, Rio de Janeiro, Brazil, 2003.

[13] P. R. Pietzuch. *Hermes: A Scalable Event-Based Middleware*. Ph.d., Queens' College, University of Cambridge, 2004.

[14] E. Ridge, D. Kudenko, D. Kazakov, and E. Curry. Moving Nature-Inspired Algorithms to Parallel, Asynchronous and Decentralised Environments. In *Self-Organization and Autonomic Informatics, Frontiers in Artificial Intelligence and Applications*. IOS Press, 2006.

[15] R. Vanegas, J. A. Zinky, J. P. Loyall, D. Karr, R. E. Schantz, and D. E. Bakken. QuO's Runtime Support for Quality of Service in Distributed Objects. In *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*. The Lake District, England., 1998.