Reflective Middleware for Location-Aware Application Adaptation*

Uzair Ahmad[1], Mahrin Iqbal[2], Uzma Nasir[2], Arshad Ali[2], Sung Young Lee[1]
Mudeem Iqbal[3,] Young-Koo Lee[1]

[1]Computer Engineering Dept. Kyung Hee University, 449-701 Suwon, Republic of Korea
[2]NUST Institute of Information Technology, [3]FAST, Pakistan
[1]{uzair, sylee, yklee}@oslab.khu.ac.kr,
[2]{54mahrin, 54uzma,arshad.ali}@niit.edu.pk

Abstract. Today mobile computing is pervasively taking over the traditional desktop computing. Mobile devices are characterized by abrupt and unannounced changes in execution context. The applications running on these devices need to be autonomous and thus dynamically adapt according to the changing context. Existing middleware support for the typical distributed applications is strictly based on component technology. Future mobile applications require highly dynamic and adaptive services from the middleware components i.e. context-aware autonomic adaptation. Traditional middleware do not address this emerging need of wide ranges of mobile applications mainly because of their monolithic and inflexible nature. It is hypothesized that such application adaptation can be achieved through meta-level protocols that can reflectively change the state and behaviors of the system. We integrate Component Technology with our active Meta Object Protocols for enabling mobile applications to become adaptive for different contexts. This paper implements the application adaptation service of this middleware. It specialize the concept of Meta Object Protocols for autonomic adaptation of mobile applications. ActiveMOP provide robust and highly flexible framework for autonomic component development for mobile applications. It proved to be a very simple and powerful way to programmatically develop location-driven applications based on autonomic components.

1 Introduction

Adaptivity is a distinguishing characteristic of emerging Ubiquitous Computing environments that clearly separates it from desktop computing. Software are supposed to be adaptive towards changing environments, ad hoc networks, multitude of communication protocols and most importantly user's current context.

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment)

Desktop computing is no longer sufficient to meet the ever-increasing information needs of the highly mobile world. Future computing environments promise to free the user from the constraints of this stationary desktop Computing. But mobile devices are characterized by abrupt and un-announced changes in execution context. The applications running on these devices need to be autonomous and thus dynamically reconfigure according to the changing context [1].

The conventional Middleware do not provide appropriate support for dealing with the dynamic aspects of this new mobile computational infrastructure. Due to this reason today the research focus is shifting from Static Design time services to run-time adaptive services. [2], [3], [4] and [5]. Application adaptation can be achieved through meta-level protocols that can reflect internal state and behaviors of the system.

1.1 Application Adaptation and Meta Object Protocols

Reflection is a discipline to achieve inspection and adaptability [4], [6], [5] and the protocols designed to achieve reflection are known as the "*Meta Object Protocols"*. MOPs can enable software to change itself e.g. alter its behaviors, reconfigure its settings, recover the damages, optimize its structures, and install new security mechanisms. As shown in Fig. 1, the meta object can be changed at runtime to get adaptive services.

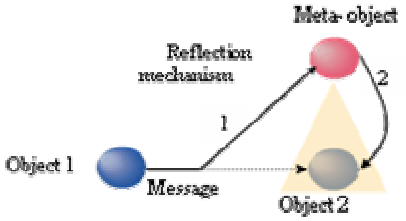


Fig. 1. MOPs at work

Meta Object Protocols are an abstraction of the computational process where the protocols governing the execution of the program are exposed. A *Meta Object* is bound to the base object and controls the execution of the object. The method calls going to base object are intercepted and diverted to the Meta Object. By changing the implementation of the Meta Object the object's execution can be adjusted in a principled way.

1.2 A Middleware using MOPs

A major advantage of using middleware to develop software is that it hides the details of the underlying layers and operating system specific interfaces. Developers of distributed applications can write code that looks similar to code for centralized applica-

tions; the middleware takes care of networking, method dispatching, scheduling etc. The code running on top of the middleware is easily portable and the programmer need not to worry about the internals of the operating system and of the middleware.
System and application code may use meta-interfaces to inspect the internal configuration of the middleware and when needed, reconfigure it to adapt to changes in the environment. Hence it becomes possible to choose networking protocols, security policies, encoding algorithms [3], and various other components to provide personalized service for different contexts and locations.
Our focus is on building a fully dynamic middleware thus providing development APIs for the development of Location-Driven applications so that at runtime behavior adaptive services can be provided depending upon the location of the mobile client. This research specializes the concept of Meta Object Protocols and introduces ActiveMOP that has been used in our Location Driven Adaptive Middleware (LDAM) for robust highly flexible application adaptation.

2 Existing work

Dynamic TAO [7] is an extension of the C++ TAO ORB [8], enabling runtime reconfiguration of the ORB internal engine and of applications running on top of it. But being built on top of the static code of TAO it does not provide much flexibility for offering adaptability. The Open ORB project [9] aims at the design of highly configurable and dynamically reconfigurable middleware platforms to support applications with dynamic requirements. It is built using components, component frameworks and reflection. But using many component frameworks increases the size of the middleware implementation; extra management functionality for managing reconfiguration exhausts the constrained resources of the mobile device. Moreover these existing systems like Dynamic TAO and Open ORB are built for application domains, such as multimedia and real-time only and do not address many other issues related to middleware in mobile computing. [5] Identifies that the key property in supporting mobile computing is the ability to seamlessly interoperate with the range of ubiquitous devices that are encountered by the mobile device as it changes location. Therefore, the Universal Interoperable Core (UIC) [5] has been developed; this reflective middleware is loosely based on the reconfiguration techniques of Dynamic TAO. The platform can change between different middleware personalities. But the implementation of UIC concentrates on synchronous middleware styles and does not implement all paradigm types that could be encountered in a ubiquitous environment. [10]

ReMMoC [10] also examines the use of reflection and component technology to overcome the problems of heterogeneous middleware technology in the mobile environment as in OpenORB. But, ReMMoC consists of two key component frameworks: (1) a binding framework for interoperation with mobile services implemented upon different middleware types, and (2) a service discovery framework for discovering services advertised by a range of service discovery protocols [10]. Hence it over comes the problem of exhaustion of the mobile devices resources. It uses OpenCOM [2] as its underlying component technology. OpenCOM uses a subset of Microsoft COM technology, hence is not an open source middleware. It needs an Active Space

(a physical space where mobile devices communicate via certain set mechanisms) called Gia for its execution.

To our best knowledge MOP are not used to specifically address the Location-driven Adaptation of applications for handheld devices and mobile clients. This problem faces a number of challenges in order to be operative for really interactive mobile applications. There is a need to extend component technology to take the benefit of MOP in order to support rich behavior adaptation. They should be supporting heterogeneous client handheld devices and independent of a particular wireless operating environment. Furthermore unlike existing MOP implementations, it should be self managing to support autonomic component development.

We propose a design pattern called the Autonomist Design Pattern that can be used to design Meta Object Protocols for dynamic adaptation of mobile applications. ActiveMOP has been built using Autonomist DP that supports behavior as well as application adaptation. Location Driven Adaptive Middleware (LDAM) is a middleware that implements ActiveMOP to provide development as well as run time adaptive services to location driven mobile applications across different environments in a principled manner.

3 Autonomist DP

As part of this research we have developed a design pattern, known as the Autonomist DP that supports design of Meta Object Protocols for application adaptation and construction of components that are Autonomic in nature. The design pattern offers adaptation at the behavioral as well as the application level. Its participants are shown in Fig 2. The Autonomic Component is at the base level, while the Autonomic Service and Service Template are at the meta level. The Meta Controller acts as a mediator between the base and the meta level and intercepts all the calls coming to the base level, sets the meta level with new behaviors and calls the base level that experiences new changed behaviors.

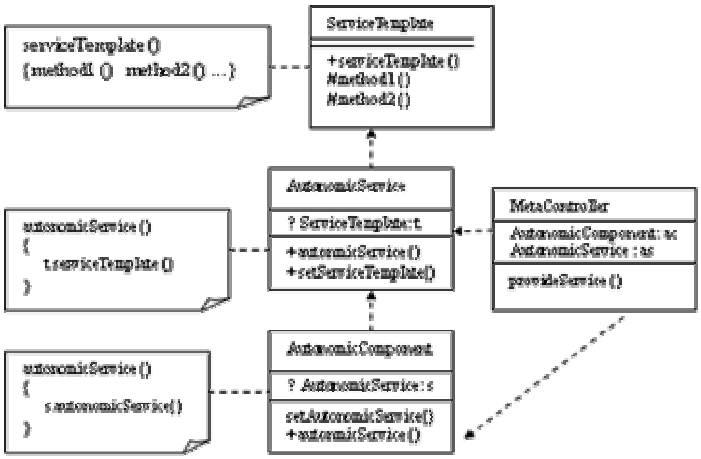
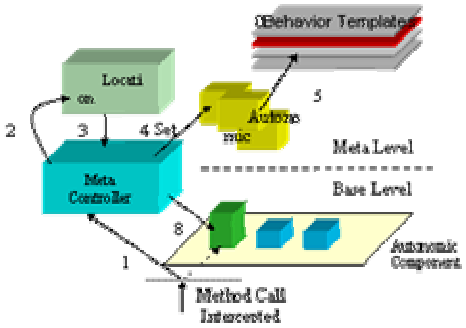


Fig. 2. Autonomist DP Participants

4 ActiveMOP

To show the strength of Autonomist DP, ActiveMOP was constructed. Fig. 3 shows ActiveMOP working. When a mobile client invokes the base level, the Meta Controller intercepts this call. The Meta Controller registers this client with the location service (which is peer research involving location sensing MOPs). After registration the Meta Controller is constantly updated about the clients location and sets the Autonomic Service with each update. In turn the Autonomic Service sets the Behavior Template based on the location value. Since location updation and behavior switching



is done actively, that is why this MOP is called the ActiveMOP.

Fig. 3. ActiveMOP working

5 LDAM

Location Driven Adaptive Middleware (LDAM) is a middleware that provides development as well as run time adaptive services for location driven mobile applications across different environments. ActiveMOP has been used to make LDAM capable of behavior level as well as application level adaptation specifically based on changing location.

5.1 System Architecture of LDAM

In the LDAM architecture the behavior adaptive services are realized through the components of ActiveMOP. All the components are part of a centralized Container that pools different components for instant execution.

- Container.

Container is the main holder of all the components. It controls all the activity of the reflective middleware. When the container is started up, it initializes all the Autonomic Components in a pool. It also initializes a Meta space that contains a pool of various template behaviors, the autonomic service object pool and the location service object pool.

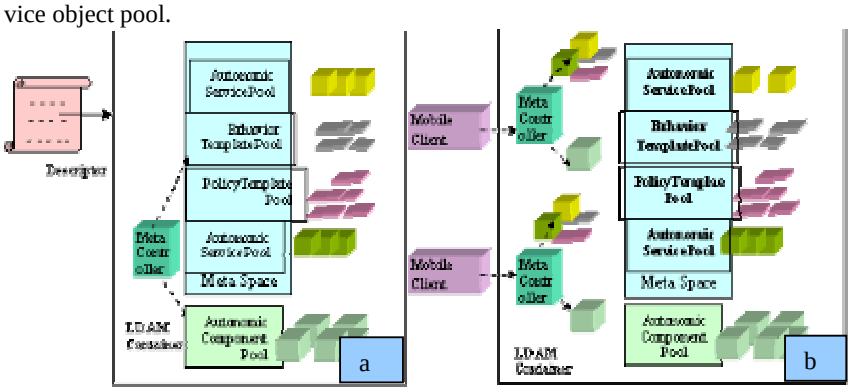


Fig. 4. (a) Descriptor defines LDAM container; (b) Dedicated Meta Controller for each mobile client

At the time of arrival of a client, the container fetches a behavior adaptive service (Autonomic Service) object and a location service object from the respective pools and binds it with a dedicated Meta Controller.

- Meta Controller

The Meta Controller acts as a mediator and interceptor of calls coming to the base level. It sets up the Meta level based upon the current location of the client and calls the base level method to execute the new behavior. The Meta Controller offers network transparency and hides the implementation details from the client.

- Autonomic Component

The Autonomic Component is a representative of the client application. It is the base level component containing an Autonomic Method, the execution of which is controlled by the Meta Controller.

- Behavioral Templates.

The behavioral templates are different template applications that are actually the adaptive behaviors for each particular location. Service Template is an interface that defines these behavior templates. These different BTs are the multiple implementations identifying a complete working component present against each Autonomic Component.

- Autonomic Service.

Autonomic service is setup by the Meta controller for each client. The location attribute is passed to the autonomic service by the Meta controller whenever behavior adaptation is required. The task of the Autonomic Service is to match the behavior with its corresponding Location. The resulting Template Behavior is set for execution

in the service residing in the Autonomic Component. The Meta Controller invokes the Autonomic Method to execute the new template behavior through this service.

- Location Service.

The second part of this research is to provide Meta Object Protocols (MOPs) for location monitoring, acquisition and updation. It facilitates the development of location sensitive applications by providing standard vocabulary in the form of APIs that encapsulate in them the description of location awareness thus shielding the programmer from many tedious and error prone aspects of location determination of an object representing a mobile device.

7 Implementation Results

The initial size of the client is very small, but without behavior adaptation the increase in behaviors causes a drastic increase in client size.

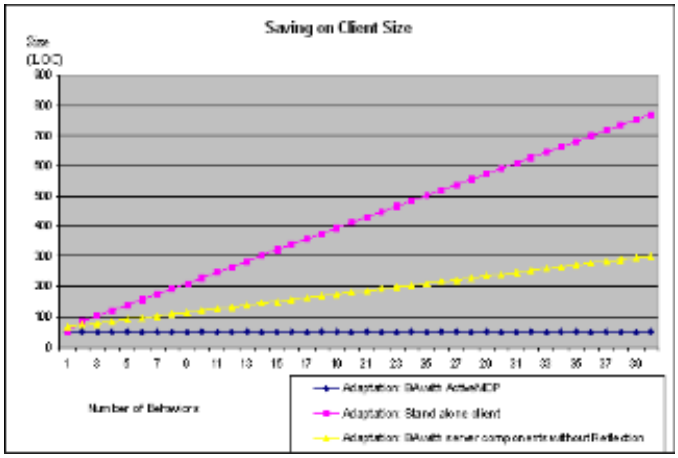


Fig. 5. Behavior Adaptation (BA) service size comparison

Through behavior adaptation in LDAM though, no code for the behavioral templates is present on the client. For as many behaviors there is no change in client size. This is shown in Fig.5.

Pooling was incorporated in LDAM after seeing the remarkable results found for component loading when they were already pooled. These results have been shown in Fig. 6. As the number of components increases the time for initializing them increases rapidly.

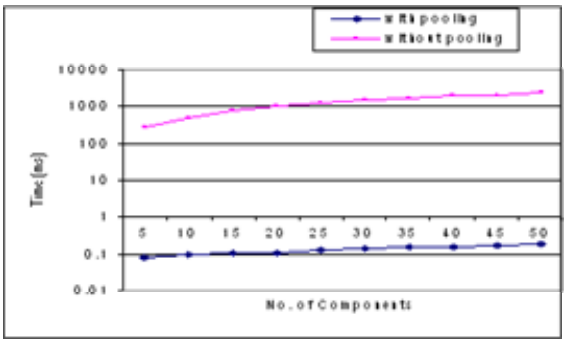


Fig. 6. Pooling vs. no pooling

The startup time for the client was noted to vary between 2ms to 20ms. This was the case when the Container initializes the Meta Controller by fetching components from each pool. Incase there is no pooling the components have to be instantiated each time a new client comes. This was noted to vary in range of 20 to 37 milliseconds.

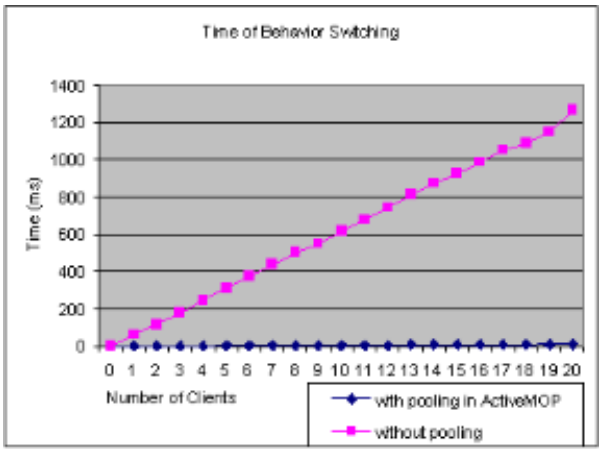


Fig. 7. Behavior switching time comparison

Pooling also resulted in fast execution of behavior adaptation process. At location change time the behavior switching was transparent for the client, i.e. it took the Meta controller nearly 0 ms to change the behavior. Whereas without pooling, when each time instantiation of components is required, it took 15 to 64 ms to switch a behavior. Pooled components save both startup as well as execution time of the client.

Fig. 8. Shows ActiveMOP comparison with OpenCOM for, the number of null method (method with no code) invocations possible in one millisecond. Through

ActiveMOP 4.219 calls can be made in one millisecond while in OpenCOM only 0.78 calls can be made in one millisecond.

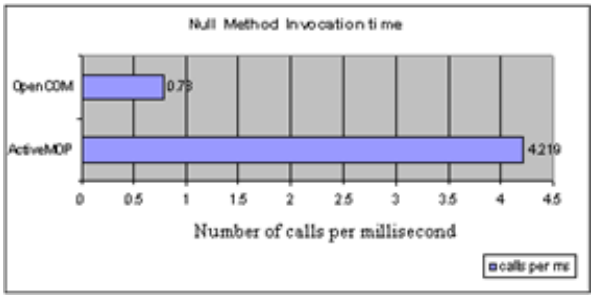


Fig. 8. ActiveMOP comparison with OpenCOM

All the tests for LDAM were conducted on Pentium III 800 M-Hz machine with 256MB RAM. The tests for OpenCOM had been performed on Dell Precision 410MT workstation with 256Mb RAM and Intel Pentium III processor 550Mhz. The operating system was Microsoft's Windows2000.

8 Conclusion

In this paper a generic architecture for a behavioral adaptive middleware targeting location driven applications is presented where emphasis has been laid upon dynamic behavior adaptation of the client device keeping in view the fact that mobile devices have a small memory footprint and the behavior adaptive services need to be light-weight and flexible. This middleware was named LDAM, Location-Driven Adaptive Middleware. It is a combination of component technology and Meta object protocols to be capable of providing adaptive services to the mobile client.

The Autonomist Design Pattern opens up new avenues for application programmers who want to design systems to implement adaptation at different granularity levels i.e. application, behaviors. ActiveMOP provides complete supports for location-based adaptation at the behavior level as well as the application level. This adaptation is transparent for the client since pooling the components makes behavior-switching time reduce to being negligible. The LDAM client enjoys a vast range of behavioral services with no burden on the memory.

References

1. Research.IBM.com/autonomous (2002)
2. Clarke, M., Blair, G., Coulson, G. and Parlavantzas, N. “An Efficient Component Model for the Construction of Adaptive Middleware”. In Proceedings of Middleware 2001, Heidelberg, Germany (November 2001)
3. Fabio Kon et al, The case for reflective middleware Communications of the ACM Volume 45, Issue 6 (June 2002)
4. L. Capra et al "Exploiting Reflection in Mobile Computing Middleware". In ACM, SIGMOBILE Mobile Computing and Communications Review.
5. Roman, M., Kon, F. and Campbell, R. H. “Reflective Middleware: From Your Desk to Your Hand”. IEEE DS Online, Special Issue on Reflective Middleware, 2001.
6. Thomas Ledoux. OpenCORBA: A Reflective Open Broker. In Proceedings of Reflection’99, number 1616 in LNCS, pages 197–214, St. Malo, France, July 1999. Springer Verlar.
7. Fabio Kon, Manuel Rom´an, Ping Liu, Jina Mao, Tomonori Yamane, Luiz Claudio Magalhaes, and Roy H. Campbell. Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB. In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware’2000), number 1795 in LNCS, pages 121–143, New York, April 2000. Springer-Verlag.
8. Douglas C. Schmidt and Chris Cleeland. Applying Patterns to Develop Extensible ORB Middleware. IEEE Communications Magazine Special Issue on De-sign Patterns, 37(4):54–63, May 1999.
9. Gordon S. Blair, Geoff Coulson, An-ders Andersen, Lynne Blair, Michael Clarke, F´abio Costa, Hector Duran-Limon, Tom Fitzpatrick, Lee Johnston, Rui Moreira, Nikos Parlavantzas, and Katia Saikoski. The Design and Implementation of Open ORB 2. IEEE Distributed Systems Online, 2(6), 2001.
10. Paul Grace, Gordon S. Blair and Sam Samuel. Interoperating with Services in a Mobile Environment. Distributed Multimedia Research Group, Computing Department. Lancaster University, Lancaster, LA1 4YR, UK., Global Wireless Systems Research, Bell Laboratories, Lucent Technologies, Quadrant, Stonehill Green, Westlea, Swindon, SN5 7DJ. lsamuel@lucent.com, p.grace@lancaster.ac.uk, gordon@comp.lancs.ac.uk