DOMINT: Disconnected Operation for Mobile INternetworking Terminals

Denis Conan, Sophie Chabridon, Lydialle Chateigner, Nabil Kouici, Nawel Sabri, and Guy Bernard
GET / INT, CNRS UMR SAMOVAR   9 rue Charles Fourier, 91011 Évry, France
firstname.surname@int-evry.fr

1. INTRODUCTION

In the future IT society, mobility and disconnections will be the rule and no longer the exception. The adaptation to the characteristics of mobile computing can be performed by the application (*laissez-faire* strategy), by the system (*transparent* strategy), or by both the application and the system (*collaboration* strategy) [8]. As surveyed in [4], there is much work dealing with mobile information access that demonstrates that the *laissez-faire* and the transparent approaches are not adequate. Our collaboration approach is threefold. We define a development process called Mobile Application Development Approach (MADA) that is model-driven, architecture-based, and component-oriented, and which follows the Model-Driven Approach (MDA) of the Object Management Group (OMG). Application developers are responsible for specifying specific elements of their distributed applications in terms of meta-data driving cache management [5] and operation transforms performing reconciliation [2]. The "default" behaviour specified by developers may be dynamically adapted by users and middleware through reflexion. Then, we design and implement a middleware platform called DOMINT that hides as much as possible the details of the hardware, the operating system, and the telecommunication protocols from application developers and users. DOMINT is integrated in the component-oriented middleware OpenCCM [7] conforming to the CORBA Component Model (CCM) of the OMG. Next, users and middleware services can rely on several detectors (included in DOMINT too) well suited to mobile environments: namely, connectivity, disconnection, and failure detectors [9]. The approach is generic and can be applied to other models of software architectures and components.

2. MADA

In our collaboration approach for dealing with disconnected operation, developers specify the behaviour of the distributed application while being disconnected in terms of cache management and reconciliation semantics.

2.1 Modelling of cache management

In Coda, Odyssey, and Rover [4], non-optimal or bad choices can be made when users do not fully understand the application, and the application can then freeze. We propose three types of meta-data to select the entities to work with while being disconnected. The disconnectability meta-data contain information indicating whether a "server" object on a remote host can have a proxy on the mobile terminal. This proxy called a disconnected entity will be used during disconnection. Then, a necessary entity is a disconnectable one whose corresponding disconnected entity on the mobile terminal must be present for the disconnected mode. Finally, since memory space on a mobile terminal is scarce, priorities are assigned to disconnected entities.

In the software architecture, use cases identify application functionalities (services) which are accessed by users through a GUI linked to a local component acting as a "Façade" (design pattern). Architects specify which use cases are disconnectable (provided) and necessary during disconnection, and their corresponding priority. Thanks to other views [6], architects tag interfaces, components, *etc.* Figure 1 depicts the rest of the development process: the Platform Independent Model (PIM) is transformed into a Platform Specific Model (PSM) following a CCM profile for disconnection management.

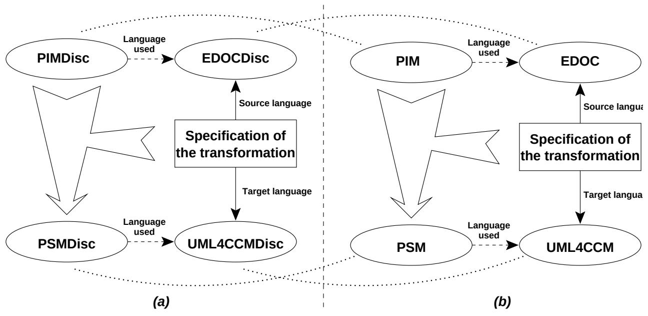

Figure 1: MDA view of MADA

2.2 Reconciliation management

Potential disconnections of mobile terminals can lead to divergence between the disconnected entity and the remote one remained on the server side. This calls for reconciliation mechanisms in order to synchronise both entities at reconnection time. Reconciliation management is based on the

SOCT4 algorithm [10] from the operation transforms technology. This algorithm exploits operations semantic properties in order to serialise them and to maintain the shared object consistency. We now briefly present the main characteristics of SOCT4. Two operations op1 and op2 are causally dependent if op2 depends on the effects of op1 or inversely. Two operations are called concurrent if they are not causally linked and performed from the same initial state on two distinct sites.

To guarantee data consistency at the reconciliation step, the operation transform technique requires the satisfaction of three conditions: 1) Causality preservation - A simple way to ensure causality preservation is to have operations be sequentially ordered using a global sequencer. 2) User intention preservation - Two concurrent operations can be executed in different order on different sites. However, in order to preserve user intention, if a site performs op1 before op2, the effect produced by op1 has to be taken into account when op2 is executed. The solution consists in transforming the operation before it is executed using a forward transposition. 3) Copies convergence - In order to obtain the consistency in all cases, the forward transposition used in SOCT4 has to verify that the execution of op1 followed by the execution of op2 which takes into account the modifications generated by op1 produces the same result as the execution of op2 followed by op1 applied on op2.

3. DOMINT

3.1 Architecture

We designed a first platform dealing with disconnected CORBA objects [3]. The concepts of component, container, factory, and required and provided interfaces allow for a cleaner design (well-defined interception points, separation of concerns...). Figure 2 depicts the architecture with the main middleware services: cache management, reconciliation management, and detectors. The first two of them work collaboratively with the components of the application, the last one is transparent to them.

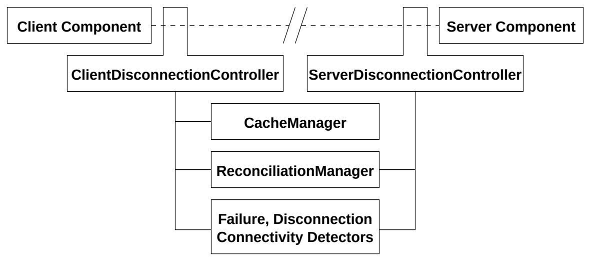

Figure 2: Architecture of DOMINT

3.2 Detectors for context management

Disconnections and failures need to be detected so that the middleware itself or the application can perform preventive and corrective actions. In addition to unreliable failure detectors [1], we introduce connectivity and disconnection detectors [9].

Connectivity detectors are entities dedicated to the estimation of local resources availability (battery, bandwidth, memory...) for wireless communication. The connectivity detector relies on a hysteresis mechanism for smoothing variations in resource availability. The thresholds of the hysteresis are configurable by application users, allowing them to define what is strong, weak, or null connectivity. Disconnection detectors execute a distributed algorithm that tries to notify neighbouring entities just before disconnection and that detects the disconnection of remote entities. When notification messages cannot be transmitted, the disconnection may be seen as a failure, thus preserving safety properties. Hence, the semantics of the distributed applications (*e.g.*, the properties of the consensus) can take disconnections into account in addition to failures.

4. CONCLUSION

The development of DOMINT for OpenCCM is in progress. Structural reflection through disconnected component cache management and behavioural reflection through container interception mechanisms allow adaptation to application needs. The platform is under evaluation on an application scenario for crisis management with groups of mobile users.

5. RÉFÉRENCES

[1] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *JACM*, 43(2), Mar. 1996.

[2] L. Chateigner, S. Chabridon, and G. Bernard. Service de réconciliation pour la synchronisation de copies. In *Journées Mobilité et Ubiquité*, Nice, France, June 2004. In French.

[3] D. Conan, S. Chabridon, O. Villin, and G. Bernard. Disconnected Operations in Mobile Environments. In *Proc. 2nd IPDPS Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Apr. 2002.

[4] J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM CSUR*, 31(2), June 1999.

[5] N. Kouici, N. Sabri, D. Conan, and G. Bernard. MADA, une approche pour le développement d'applications mobiles. In *Journées Mobilité et Ubiquité*, June 2004. In French.

[6] P. Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, 12(6):42–50, Nov. 1995.

[7] ObjectWeb. OpenCCM home page. http://www.objectweb.org/openccm, 2003.

[8] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proc. 15th ACM PODC*, May 1996.

[9] L. Temal and D. Conan. Détections de défaillances, de connectivité et de déconnexion. In *Journées Mobilité et Ubiquité*, June 2004. In French.

[10] N. Vidot, M. Cart, J. Ferrié, and M. Suleiman. Copies convergence in a distributed real-time collaborative environment. In *Proc. ACM CSCW*, Dec. 2000.