**Introduction**

Since the early 90’s, the field of mobile computing has witnessed tremendous research and technological advances. With wireless communications and mobile hand-held or wearable devices becoming a reality, new applications where users can have access to information anytime, anywhere are made possible. In the future IT society, terminal mobility and disconnections will be the rule and no longer the exception (cf. Figure 1). The emergence of the new field of pervasive computing as a successor to both distributed systems and mobile computing endorses this vision: environments will be “saturation of computing and communication capability, yet gracefully integrated with human users” [9].

The adaptation to the characteristics of mobile computing can be performed by the application (laissez-faire style), by the system (transparent style), 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.

**MDA**

In our collaboration approach for dealing with disconnected applications, developers specify the behaviour of the disconnection point while it is being disconnected in terms of cache management and reconciliation semantics.

**Modelling of cache management**

Figure 4 depicts the architecture with the middleware services: cache management, reconciliation management, and detectors. The first two of these work collaboratively with the components of the application, the last one is transparent to them.

**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 synchronize both entities at reconnection time. Reconciliation management is based on the SOCT4 algorithm [11] 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. Thus, a sequence is not causally consistent if it visits op1 before op2 is executed. The solution consists in transforming the operation before it is executed in order to accept the modifications generated by op2. 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 into 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 [10]. The approach is generic and can be applied to other models of software architectures and components.

**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 [10]. Connectivity detectors are entities dedicated to the estimation of local resources availability (battery, bandwidth, memory...) for wireless communication. The connectivity detector relies on a hyestraisis mechanism for smoothing variations in resource availability. The thresholds of the hyestraisis 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 (the properties of the consensus) can take disconnections into account in addition to failures. Figure 5 depicts the detectors’ relationships.

**Conclusion**

The development of DOMINT for OpenCCM is in progress. Structural reflection through disconnected component cache management and behavioral 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.