An Adaptive Middleware to Support Context-Aware Knowledge Sharing

Roberto Boselli, Federico Cabitza, Flavio De Paoli, Marco Loregian
Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano Bicocca
via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
{*boselli, cabitza, depaoli, loregian*}*@disco.unimib.it*

Abstract

Due to the diffusion of ubiquitous computing environments, the need for adaptive applications to effectively support knowledge sharing, and improve context awareness without interfering with habitual work practices has gained importance. This paper presents the design of an application architecture that exploits an ontology-based knowledge management framework and a reflective middleware to support multi-channel applications. The development of the middleware is carried on in parallel with the development of a real application, SWIRLS, that aims to support the work in a hospital setting. SWIRLS is a case study with real technology and organizational problems that involves innovative devices, seamless interaction via different channels and domain-dependent knowledge management issues.

1 Introduction

In this paper we outline a software architecture designed to support ubiquitous applications by supplying the ability both to dynamically adapt to current context and to properly manage knowledge. Context includes the technological environment and the performed activities. Knowledge is typically expressed by the semantic content of documents that users feed into the system. Our middleware-based proposal exploits lower-level services provided by two consolidated and mature projects, MILK [13] and MAIS [12] and represents an ongoing endeavor to integrate them to support real work settings and habitual user practices. The work is carried on in parallel with the development of a demonstrator, SWIRLS [6], to test and validate the proposed solutions.

SWIRLS (*Supporting Wards with Interactive Resources and Logic-based Systems*) is an application envisioned with a twofold aim: on the one hand it is designed to support hospital practitioners without disrupting their habitual practices and by augmenting the natural interaction with paper-based forms through context-adaptivity of situated whiteboards; on the other hand, SWIRLS supplies a real application-level domain that is complex enough to verify the architectural hypothesis of our novel integrating middleware. For this reasons a scenario, emerging from our field studies and thus derived from a real situation, will be given to illustrate the application functionalities and enabling technologies.

In the rest of the paper, the background of the presented project is described, MILK and MAIS projects are briefly outlined. Moreover, as an example of the proposed architecture, a demonstrator is described. In the last sections, some related works are discussed, followed by implementation issues and the conclusion.

2 Conceptual background

The problem we are dealing with, which is the creation of an adaptive middleware architecture that enables knowledge sharing in an open mobile environment, presents issues mainly regarding two topics: context-aware interaction and technical development.

The interaction area has two main foci: (1) what people do in terms of activities and contents, and (2) how people accomplish a given task. The former deals with questions like what people need (or want) to do with systems that go beyond traditional mouse-monitor-keyboard devices; what "allowing for seamless transition between devices and contexts" means; how private spaces integrate with social shared spaces; and so forth. The latter deals more with user interfaces and physical interaction: how to design the graphical presentation of contents properly; how to use effectively new devices (e.g., digital pens); how people interact with the system to accomplish certain tasks in a multi-modal environment; how to make people aware of the context (environment and other users); and so forth. In short, we need to define what the requirements for this new class of applications are. It is worth saying that this is not (or not only) a technical problem, therefore analysis from

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 **IEEE**

different perspectives is necessary. Starting from the experience acquired in previous projects, we are designing experimental applications that allow for further investigations.

As a consequence of widening interaction possibilities, contents and ways to represent them need to be rethought and redesigned to fit the new delivery channels. A naive approach is to simply resize the displayed information (e.g., to shrink an interface to fit the tiny screen of a cell phone). It is well known that it does not work. Therefore there is the need of supplying different representations of the same information depending on the kind of device and connection in use. In addition, also the situation affects the way information is displayed. These two aspects – technology and context – should always drive design, management and retrieval of information. Another key aspect is the interleaving of social and private interaction. In the former case the system should supply functions and information depending on generic parameters (in accordance with some domain and community practices). In the latter, the system should allow a certain degree of personalization and should support the user in his/her current activity. Representation management, correlation of information and filtering are already supported by the MILK Metadata Management System that will be described later (see 3.2). The MILK-MAIS integration we are now pursuing is a great opportunity for exploiting these features in realistic experiments.

To better achieve a successful support to complex interactions, a clear understanding of technical development issues is necessary. "The reflective middleware model is a principled and efficient way of dealing with highly dynamic environments yet supports development of flexible and adaptive systems and applications" [10]. The MAIS project indeed promotes the idea of systematic development of multi-channel adaptive applications based on a middleware approach. The approach is in-line with the evolution of computer science: evolution by abstraction. In the 60s evidences emerged of the need to handle the most common and shared activities performed by programmers in a more efficient and practical ways: the development of operating systems was the answer. In the 80s and early 90s a similar need emerged for distributed system programming: the answer was CORBA and the object model. Modern middleware such as the so-called "application servers" are evolution of that answer. Even today there are people claiming: "I do it better with sockets. I get more flexibility, more performance, why should I bother with component models, EJBs and so forth?". As for the long-running discussion between C and higher level languages, there is some truth in these claims, but since most of the people are now convinced that the more structured a language is, the more effective in long terms; most of multi-device, multi-channel, ubiquitous, pervasive applications' programmers will agree that systematic development supported by an infrastructure facilitates the development of *real* applications. The question is: what are the requirements for such an infrastructure? for such a middleware? Nobody has the right answer today, but it is worth investigating to identify problems and needs.

3 Project background

In this section the key features of MAIS and MILK projects, that are the cornerstones for our current work, are presented.

The goal of the MAIS project is the deploy of a platform according to models and methodologies enabling the creation of adaptive and flexible information systems. To this end, MAIS defines a reflective architecture [2] supporting the creation of a system that is able to observe and control its own structure and run-time behavior. In the MAIS architecture, reflective knowledge is expressed in terms of Quality os Service (QoS). QoS means that, given a certain service (e.g., localization service, displaying service, etc...) the middleware lets the application access and possibly control its characteristics (e.g., the bandwidth, the size and resolution of the screen, etc...).

In our context, the location is of particular interest. The position of each device can be static or dynamic. In the former case (e.g., for a desktop PC), the location is fixed and can be entered at setup time and assumed valid at any time; therefore, no special location technology is required. In the latter case (e.g., for mobile phones and WiFi devices), extra effort is required to track the current position by exploiting a fixed or mobile infrastructure with different degrees of precision. In recent past, there have been many research efforts on wireless positioning of mobile devices. As a result, it is now possible to determine the geographical position of users with sharp approximation (e.g., by UMTS or GPS technologies), or coarse approximation (e.g., by triangulation techniques of telephone cells for the GSM protocol). The reflective architecture is in charge of detecting such properties and updating them as soon as relevant changes in the environment occur.

The MILK project has delivered a framework, named MMS, to support knowledge management and knowledge

sharing among people belonging to an organization. The framework is potentially accessible in multi-modal and multi-channel fashion. In the MILK project we have developed a PC-based interface [1], ad-hoc solutions to support both social interaction, via interactive shared screens, and private interaction via cell phones. An open issue is still the development of an integrated and adaptive support for seamlessly moving among different environments. The ongoing work is focusing on the integration of the MILK system in a reflective architecture, given by the MAIS project. Our goal is now to exploit the features of the reflective architecture to support multi-channel access to MILK, identifying further requirements for multi-modal interactions in MAIS and enhancing the knowledge on requirements when supporting seamless transition between interaction devices, modes and contexts, and consolidating at the same time the MILK MMS features by experimental verification.

MILK interaction approach. MILK exploits a new approach to interact with users. With the traditional searching features, the MILK system provides the user with a discovery feature that fetches and shows information that is related to the current user activity and hence to the context. Discovery is addressed by the *view with context*, a metaphor realized in a profiling mechanism that binds together elements of different nature [5]. On the screen, the element a user has retrieved (e.g., a document) is displayed surrounded by other elements that are similar to it (e.g., related documents, expert people and projects on the same subject). In such a way, users become aware of what is available and can discover novel information. In the following scenario (see 4.1) this presentation technique will be simply addressed as "workspace". Other details on the PC interfaces have been thoroughly discussed in [1].

Metadata Management System. The MMS is the component accomplishing knowledge organization. The MMS is composed of a set of modules that manage the information associated with documents, people, communities and projects, which are collectively referenced as "elements" within the proposed model. The physical repository of these elements (as well as of any other MILK element) is the Document Management System (DMS). MMS elements are described by means of metadata that capture basic information (e.g., names, references, dates, ownerships, etc.), content information (e.g., free or user-defined keywords and system or classifying keywords), qualifying information (e.g., relevance, frequency of use, user rating, etc.) and finally relations with other elements within and outside the system. Relations are key issues for a knowledge management system since they support advanced information retrieval and information discovery. Besides relations explicitly maintained in element profiles (such as bookmarks and threads of discussion), the MMS can compute and filter relationships to address issues related to the context of use (e.g., user profiles and environment). The context-based content adaptation is a fundamental issue in MILK, and it is accomplished through the concept of representation. In the MMS, the "classical" document is replaced by an artifact, i.e. a compound object constituted by various files, each of them being a different representation of the same content. Different representations are needed to support different contexts and different channels. In fact, when the system delivers information, it has to provide data in a format accessible according to the context of the user activity. A good reason to introduce different representations is related to technology: some representations are not suitable for certain technology. For example, it is not so desirable to watch a presentation or read a technical paper on a mobile phone. The reflective system is in charge of detecting the available channels and suggesting the best representation available in terms of both technology and context of use (QoSs).

4 SWIRLS: the reference application

In this section, the SWIRLS application, which is the test bed of our project, is presented.

The hospital ward represents a very challenging field of work for ubiquitous applications since its domain is characterized by high mobility and spread of heterogeneous actors playing different roles over time. Moreover hospital ward work is organized in a delicate balance between a strict planning, which is driven by quite well established protocols and practices, and a constant and swift adaptation to ever changing contingent needs so as to guarantee assistance and care in any way, regardless of the scarce resources. Although computer technology offers several kinds of support to this highly dynamic environment by enhancing actors' capabilities of storing, fetching, and communicating information according to current context, hospital practitioners have always been quite reluctant in adopting computer applications that propose a more or less accentuated dismissal of paper and traditional artifacts [18]. As a consequence, the level of digitalization in hospitals is still quite low and different kinds of healthcare stakeholders ask for new approaches to design tools that reach an acceptable compromise between natural human-computer interaction and context-aware knowledge management.

SWIRLS is our endeavor to address these compelling research issues: by designing SWIRLS we intend to develop a sample system that, by relying on a stable distributed architecture and by adopting the general guidelines of ubiquitous computing paradigm, supports practitioners in their habitual hospital *knowledge work* and *articulation work*. As such, SWIRLS can be considered a suitable reference application for the middleware we are developing: it conceives

computation as spread throughout the environment, across small devices, either mobile or embedded in the background infrastructure.

In the SWIRLS application domain the focus is on how practitioners can be supported by information technology without disrupting their habitual practices and through a process of augmentation of habitual technology and devices. For this reason, following a first effort towards the identification and analysis of artifacts used daily by practitioners, we detected that paper-based forms and wall-mounted whiteboards are among the most useful artifacts [3]. We have found that generally paper-based artifacts can be augmented by the use of the Anoto technology, whereas whiteboard-based interactions can be enhanced by the use of interactive touch-sensitive large screens. Although we think that these enabling technologies do address interaction requirements in a novel and promising way, that notwithstanding we also think that only within a comprehensive solution (encompassing a middleware able to provide context adaptivity and advanced knowledge management) these tools could really support workers even in one of the most demanding domain as hospital work can be. In the next section a scenario envisioning SWIRLS application, integrating MILK and MAIS in healthcare domain, is presented.

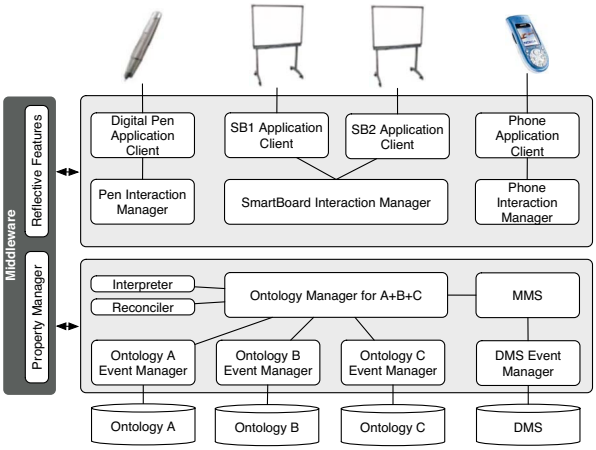
Dr. Pierce, an orthopedic physician, is in the practitioners room at the hospital; he is prescribing a blood examination that Mrs. R.D. — one of the inpatients he is accountable for — should undergo the next day by filling a form by means of a digital pen. The prescription form is a preprinted checklist in which all the most common examinations are reported; each item on this list has two boxes aside: one for the physician's mark affixed at prescription time and the other for the nurse's sign, that she will write once the blood samples are ready to be processed by the lab; when the physician puts her/his initials at the bottom of the order form, all the marked examinations are intended as prescribed and booked at the lab, at the same time the corresponding tube stickers are also printed.

In order to publish the prescription, Dr. Pierce must place the digital pen into its cradle, so that the patient prescription is downloaded; the cradle is placed aside the Smart whiteboard and, once he approaches it, his RFID badge is sensed by an RFID tag reader at the board; as a consequence on the screen the system brings into focus the elements that were blurred for privacy reasons, highlights ward information that relates directly to Dr. Pierce and opens his personal workspace where personal information and data (such as calendar and to-do lists) are displayed.

The moment also Dr. Forrest enters the room, she is also identified by the system by means of her Bluetooth-enabled mobile phone and when she approaches the Smart whiteboard, also her RFID badge is sensed by the board RFID tag reader. Now also Dr. Forrest workspace is available on the screen; except for private documents and confidential messages (marked as such by their sender) that Dr. Pierce should not be allowed to read, work notes, public announcement and messages left for Dr. Forrest by other practitioners are promptly displayed within her workspace, so that she may read, delete or possibly store them in her smartphone or palmtop. When the board senses Dr. Forrest personal badge, the system highlights with a different color the patients' names that she has in common with Dr. Pierce; likewise the icons corresponding to each of these patients are represented by a cluster of related elements that refer to data regarding their medical trajectory. By such information filtering and presentation modulation (highlighting), the system attempts both to prevent information overloading and to facilitate face-to-face interaction among the two physicians by providing them with a convenient way to recall significant and up-to-context information about patients. With such a reminder, Pierce and Forrest end up by considering together a particular patient case, Mr. G.D.. Dr. Pierce takes from G.D.'s electronic patient record the image of a computed tomography (CT) scan that the patient underwent the day before, and begins browsing the attached report while commenting it with Dr. Forrest. The report is a perfect copy of the original hard copy but it is also enriched by many hyperlinks both towards other G.D.'s health data and to both corporate (LAN) and worldwide (Internet) medical knowledge resources.

5 The application architecture

As depicted in the scenario above, the SWIRLS application exploits a large number of devices of various kinds, interacting in a peer-to-peer fashion. Each component (i.e. each software client in the system) is essentially powered by three different logical layers (application, middleware, services); each of those is decoupled from the others and composed of different tiers (see Fig. 1 for an example). The upper part of Fig. 1 is the actual application layer. This layer is split in two tiers: the Application Clients and the Interaction Managers (IM). The specific implementation of these components depends on the device (see Fig. 1) and supports the human-computer interaction, as they are responsible for processing interactions and contents presentation. The layer components, and IMs in particular, are located with the computational resources of the devices. On the other hand, according to the approach we have chosen, all other system components (i.e. service providers) may be spread over the environment, or even accessible through the Internet.



Instead of an RPC approach, we have chosen a document-based interaction style, meaning that there is no direct invocation of methods from a component to another. Form-like XML documents are exchanged to decouple the internal behavior of components and the services they supply. Resources are specified by URI's, so they might be locally resident as well as located on some remote machine acting as a server, and those resources (i.e. documents) are parsed by target entities. In the following the main elements of the architecture are presented.

Application Client. In order to properly work, each of the devices adopted in the application framework needs to be enabled by a specific software client. This component is responsible for the final presentation of the documents, or generally of information coming in XML from the underlying Interaction Manager, and thus for enforcing interaction. Different technologies will require different solutions. For example, devices with a small amount of memory or computational power (i.e. mobile phone in Fig.1) can only carry very lightweight clients. Other device, like digital pens and Smart whiteboards, cannot run software and therefore need to rely on PCs. To better explain what the tasks of ACs are, consider for example PCs, which can host complete solutions: their client may encompass both a visualization manager, such as a web browser, and a presentation engine, such as Flash interpreter. XML documents will then be parsed by the presentation component and then visualized in a way that is suitable for the current context.

Interaction Manager. The IMs context information and build a suitable representation of documents fetched from the MMS. The aim of this task is to feed ACs with tailored information. The MMS is not described in details in this section because it has already been with respect to MILK. IMs are thus the behavioral core of the applications. They are responsible for finally managing both the inputs (coming from application clients) and the outputs (gathered through the middleware from the underlying web services). We actually chose a peculiar strategy for processing inputs and outputs with respect to the context. At the moment we rely on logics that are managed by an inferential engine [7, 9]. The innovation in the process is that the knowledge inferential rules come from distributed ontologies, that are interpreted by a specific component of the OM (see next section).

Middleware. The middleware basic role is to provide communication means between applications and services but, most important, the middleware layer also supplies features to support adaptation and the knowledge management capabilities of the system. Context information may be directly or indirectly gathered through reflection mechanisms. For example it is possible to utilize network reflective information can be exploited to build topological references with respect to a given model and therefore understanding user positions in terms of proximity. Moreover, if a device is within the sensitivity range of one or more other devices then it is possible to infer its location with coarse accuracy. For example, in the scenario depicted above, Dr. Pierce was sensed to be approaching to the Smart whiteboard when he enters into the RFID tag reader range, while his presence in the room was already known according to the detection of his Bluetooth-enabled Nokia digital pen. Main components of the middleware layer are thus the reflective features and the methods for handling them, that are handled by the so-called Property Manager. Knowledge management features deal with element profiling, computing of relations among elements and organizing actual elements (i.e., documents and files). Such capabilities are addressed by dedicated tiers that exploit another tier — the OM — that supplies terms and descriptions of the domain addressed by an instance of the system.

The lower part of Fig. 1 depicts the reasoning infrastructure for the applications, meaning the services required for providing the application level the essential material to be manipulated: documents to be displayed (already in a suitable format) or just facts and rules (extracted from distributed ontologies) to feed the inferential engine in the IM.

Ontology Manager. What we generically call our knowledge base is then a set of shared ontologies that is managed by the Ontology Manager plus the documents stored in the Document Management System. Ontologies are used to represent the knowledge of a given domain through formalization of concepts and definition of relations among the

terms of the domain. An ontology generally appears as a taxonomic tree of hierarchically organized concepts or entities. Such a hierarchy is built with respect to a given relation (e.g., the *is-a* relation), but other relations (e.g., *is-part-of*) may also be explicitly represented to capture the different dependencies between concepts. The OM manager actually uses several kind of ontologies that cover different aspects of the domain and the interaction. For example we adopted ontologies for addressing and correlate *contents*, for describing the possible *contexts* and the *devices* in the domain of interest, to describe the actors in the domain in terms both of *organizational structure* and of people *roles*.

The Ontology Manager, which inherited its name and some functionalities from a former MILK OM, is in charge of managing the distributed ontologies, presenting them as a single entity to the upper level. The OM is responsible for simple tasks, such as granting access to the knowledge base (e.g., to the MMS to correlate documents), and for complex tasks, such as keeping distributed ontologies aligned and consistent. The OM has to gather different pieces of ontologies together and adopt mechanisms to create links and interconnect them. The OM is powered by two subcomponents that are devoted to each of the above mentioned tasks: the Ontology Reconciler and the Ontology Interpreter. The Reconciler encompasses strategies and algorithms to keep the ontologies and their instances updated and aligned with the system's desired behaviors and with the knowledge model in the knowledge base. It might be useful, for example, when a device is back online after being offline, to realign to the local ontology that may have been changed in the meantime. Moreover, the Reconciler has the task of merging ontologies that may have been created on specific devices and situations. This problem has been widely studied in literature. For example, the works of Pinto [15,16] focus on the problem of integration. She gives a deep analysis of what the problems of integrating ontologies are and in [15] she carefully deals with the *merge* operation. Noy and Musen [14] also studied and implemented PROMPT, an algorithm that provides a semi-automatic facility for merging and aligning ontologies.

In our approach, ontologies are used to represent knowledge, and procedural knowledge (which is also "how to adapt to context changes") in particular, in a declarative form. The Ontology Interpreter is tightly connected with IMs and is responsible for translating such procedural knowledge into rules that may be interpreted by the inferential engine associated with IMs. It is then by this component that the IM is able to gather information (i.e. facts) upon which inferences are made and it is also through this element that the rule base might be extended or dynamically adapted to any desired system behavior (i.e. matching to context-related ontology in the knowledge base). According to the relation through which the Context Ontology is structured, new rules can be generated by this component and then they may be dispatched to the IM.

Event Managers. The architecture also provides standard components for managing information flows from resources that are logically not encompassed by the system (e.g., repositories). Since there are different kind of resources, such components have to be highly customizable, in order to provide a reliable interface to repositories (which are by this mean turned to effective web services). Information and control can flow either top-down or bottom-up. In the former case the application requires services to the underlying system. In the web services model we adopted this simply maps to a request for accessing a resource. The EMs are the components that enable the "publication" of those resources. Moreover, since the same resource may be shared among different services, they may also push information to other system components. For example, when a new document is inserted in the DMS, an event is notified to the MMS so that a new profile can be created or an existing one may be updated so as to include the new entry.

6 Related works

Middleware technology, acting as a mediator between the application and the operating system, should make it easier to the programmers to develop applications that are able to adapt to highly dynamic environments. Reflection, and reflective middleware in particular, seem a suitable solution for dealing with the proliferation of computation-enabled mobile devices. Our middleware is supported by the reflective layers of the MAIS platform [2] and hence its features have not been re-implemented in our middleware. Several other frameworks providing similar functionalities may be found (e.g. OpenCORBA [11], DynamicTAO [10]). Among other solutions relying on a reflective platform to support device context-awareness it is also worth mentioning the works in the CARISMA project [8] and their use of metadata for achieving the separation between middleware functionalities and their implementation. Policy-based approaches, as the one used for CARMEN [4], have the same goal, that is the *separation of concerns*, even if they differ on how this separation is achieved. Mobile agents have also been investigated in this scope, the SALSA framework [17] also investigates our application domain, since it addresses information retrieval in healthcare. We are not going to relate further, for this paper, on literature regarding ontologies and knowledge management issues.

7 Implementation issues

At the moment this paper is written the SWIRLS application is being developed as a case study for the middleware

proposal we are describing. We are leading field studies in an Italian hospital to collect precise requirements to support the application development. In the meanwhile, consolidation work on the MILK MMS is taking place. The MMS is a web service application to decouple the DMS and client-side software. For the same reason, data are exchanged in XML format. The MAIS reflective architecture is defined through a series of reflective classes, which allow the application to observe and control in real time the QoS expressed by the system components. The current version of the IM exploits the inference engine Jess™ [9], but a distributed version of the same engine, namely DJess [7] is under development for a test deployment within the SWIRLS application. We are also investigating the possibility of adopting a (BPEL) workflow engine within the application scheme.

8 Conclusion

The middleware solution we have outlined was conceived with the purpose of supporting adaptive applications for ubiquitous computing environments. Apart from the specific demonstrator we have introduced, our aim is to develop middleware solutions that grant applications comprehensive context awareness. By "comprehensive" we mean that both environmental context, *ˆ la UbiComp*, and social context have to be taken into account to support and provide the user with effective accessibility to available services. Besides the specific technologies we have exploited in our case study, the exploitation of ontologies combined with reflective capabilities ensures openness toward a real multi-channel interaction by employing different kinds of devices. A novelty of the proposed solution lays on the development of an integrated interaction manager that is: *self adaptive*, thanks to the underlying reflective architecture and the Context Manager capabilities; *content sensitive* in presenting the content according to the context; *easy* to feed and maintain, by means of remotely-fetched distributed ontologies; *independent* of QoS property *extraction* mechanisms and of any specific user interaction channel. The preliminary results are encouraging and make us confident that our approach will accomplish the expectations.

Acknowledgements. The work presented in this paper has been supported by FIRB MAIS [12] project. The authors thank the anonymous reviewers for they precious comments that lead to major changes with respect to the submitted, longer but less complete, version of this paper.

References

[1] A. Agostini, S. Albolino, R. Boselli, G. De Michelis, F. De Paoli, and R. Dondi. Stimulating knowledge discovery and sharing. In *Proceedings of the Conference on Supporting Group Work - GROUP'03*, pages 248–257. ACM Press, 2003.

[2] F. Arcelli, C. Raibulet, F. Tisato, and M. Adorni. Architectural reflection in adaptive systems. In *Preceedings of SEKE 2004*, 2004.

[3] J. E. Bardram and C. Bossen. Interwoven Artifacts – Coordinating Distributed Collaboration in Medical Care. Technical report, 2004.

[4] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. Context-aware middleware for resource management in the wireless internet. *IEEE Trans. Software Eng.*, 29(12):1086–1099, 2003.

[5] R. Boselli, R. Dondi, and F. De Paoli. Knowledge organization and retrieval in the MILK system. In *Proceedings of SEKE 2003*, 2003.

[6] F. Cabitza, M. Loregian, and M. Sarini. Supporting wards with interactive resources and logic-based systems, in ubihealth2004: The 3rd international workshop on ubiquitous computing for pervasive healthcare applications, nottingham, england, september 7, 2004.

[7] F. Cabitza and B. D. Seno. Djess – a knowledge-sharing middleware to deploy distributed inference systems. In *Proceedings of ENFORMATIKA'05 Joint Conferences*, Istanbul, Turkey, February 25-27 2005.

[8] L. Capra, G. S. Blair, C. Mascolo, W. Emmerich, and P. Grace. Exploiting reflection in mobile computing middleware. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):34–44, 2002.

[9] E. J. Friedman-Hill. Jess, the java expert system shell, 2001.

[10] F. Kon, F. Costa, G. Blair, and R. H. Campbell. The case for reflective middleware. *Commun. ACM*, 45(6):33–38, 2002.

[11] T. Ledoux. Opencorba: A reflektive open broker. In *Reflection '99: Proceedings of the Second International Conference on Meta-Level Architectures and Reflection*, pages 197–214. Springer-Verlag, 1999.

[12] MAIS. Multichannel adaptive information systems, http://www.mais-project.it/, Italian FIRB project.

[13] MILK. Multimedia interaction for learning and knowing, http://www.milkforum.com, IST Project 2001-33165.

[14] N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000.

[15] H. Pinto, A. Prez, and J. Martins. Some issues on ontology integration, 1999.

[16] H. S. Pinto and J. P. Martins. A methodology for ontology integration. In *Proceedings of the international conference on Knowledge capture*, pages 131–138. ACM Press, 2001.

[17] M. Rodríguez and A. Preciado. An agent based system for the contextual retrieval of medical information. In *AWIC*, pages 64–73, 2004.

[18] S. Timmons. Resistance to computerised care planning systems by qualified nurses working in the uk nhs. *Methods of Information in Medicine*, 42:471–476, 2003.