A Pervasive, Service-Oriented Architecture for Supporting Teamwork

by Schahram Dustdar and Hong-Linh Truong

At the heart of the EU 'inContext' project, the Pervasive Collaboration Services Architecture (PCSA) aims at providing a pervasive, SOA-based architecture for supporting various kinds of teamwork. This architecture comprises different kinds of Web services, loosely coupled in a dynamic environment that includes diverse underlying operating systems and networks, necessary for collaboration and teamwork. The goal of this platform is to reduce as far as possible human intervention in the support of collaborative work, by means of autonomic capabilities based on context information and interaction patterns.

Recent advances in computing devices and network technology are creating diverse, pervasive environments and opening various opportunities for human collaborative work on these environments. Nowadays, members of teams in collaborative processes may span different times and spaces. Depending on the context of a collaboration, a team may be set up through an Internet-based or mobile ad hoc infrastructure, using mobile devices and/or high-end computers, working together for only a short period of time or for up to several days or weeks, just to name a few cases. More than ever before, teamwork is highly dynamic and flexible in terms of both time and space; existing infrastructures in which teamwork is supported by fixed, tightly coupled systems that include dedicated services and portals but which do not interact with each other, thus fail to provide sufficient capabilities. Not only will an SOA-based platform allow us to integrate various existing collaboration services into dynamic, pervasive environments, but newly developed collaboration services for emerging teamwork could also be easily plugged into such environments.

The inContext Project

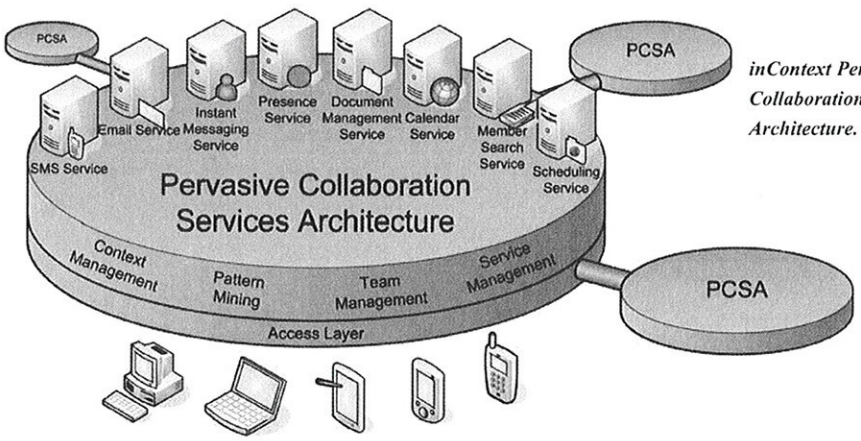
The inContext project aims at supporting highly dynamic forms of human collaboration such as Nimble (short-lived collaboration to solve emerging problems), Virtual (spanning different geographical places and involving diverse professionals) and Mobile (collaboration with mobility capabilities) teams.

These teams require different mechanisms for coordination, and in many cases also different services (eg document sharing, project management and instant messaging) and infrastructures (eg large-scale and Internet-based mobile devices, and mobile ad-hoc/P2P networks). SOA-based solutions thus offer greater advantages for inContext over other solutions, such as those that are portal-based.

The inContext PCSA

The inContext PCSA (Pervasive Collaboration Services Architecture) aggregates different types of services, all based on Web Services, to support collaborative team processes. These services are loosely coupled and can be deployed in various hosting environments.

As shown in Figure 1, this architecture comprises collaboration services and the inContext platform. In order to support basic team activities, we need plenty of common collaboration services, such as calendars, task management, instant messaging and document management. However, a variety of other services need to be developed for supporting emerging teamwork and autonomic capabilities. Hence, on the one hand we adopt and integrate existing common services by wrapping them, and by building and extending their Web service interfaces since not all services provide one. On the other hand, services in the inContext platform are able to manage context information, analyse interaction patterns and perform autonomic capabilities for the PCSA. This means the PCSA is able to fulfil the needs of different teams and to cope with changes in team forms as they evolve.



inContext Pervasive Collaboration Service Architecture.

The PCSA Network

Since team collaboration is conducted in a distributed system involving multiple organizations and spanning multiple locations, we have to ensure that team members are able to access all the serv-

ices in a transparent way. An efficient way to support these collaboration scenarios is to connect multiple PCSAs in a peer-to-peer network through which a team member can perform his/her collaboration without needing to know where and how services are deployed. As depicted in Figure 1, the inContext platform and in particular the Access Layer will route invocations to appropriate services in the PCSA network. In this respect, context information plays a key factor as it provides important information about the location and status of collaboration services.

Autonomic Collaboration Architecture

The inContext PCSA integrates and provides various services for human collaboration. Furthermore, by offering autonomic capabilities, it aims to reduce human intervention in managing the collaboration. Being autonomic requires an intensive use of context information and interaction patterns to adapt and provision services to the changes in and requirements of teams and their respective environments. Take the scheduling of a meeting as an example in which autonomic capabilities are important. In normal environments, the initiator of the meeting must select members based on roles and meeting objectives, determine possible dates based on their calendar, select potential days, send the selected days to members via e-mails or instant message or SMS, search for relevant documents needed for the meeting, create an agenda and so on. All of these steps are performed manually. With support of the inContext PCSA, this level of human intervention can be reduced. For example, the initiator may need to simply select a checkpoint in the project timeline and ask the PCSA to plan a meeting.

Based on context information about members' presence and location, human intervention in many steps like selecting members, checking calendars and sending confirmations can be completely removed. In the inContext project, autonomic aspects will be focused on work activities and the required service adaptation.

The inContext research is coordinated by the Vienna University of Technology and is conducted together with Softeco Sismat SpA, DERI Ireland, European Microsoft Innovation Center, Electrolux Home Products Italy SpA, Hewlett Packard Italiana, the University of Leicester, West Midlands LGA and COMVERSE Ltd.

Link:
http://www.in-context.eu/

Please contact:
Schahram Dustdar
Vienna University of Technology, Austria
Tel: +43 1 58801 18414
E-mail: dustdar@infosys.tuwien.ac.at

Engineering Services

by Mike P. Papazoglou

Service-oriented computing is not simply about deploying software: it also requires that organizations evaluate their business models and come up with service-oriented design and development techniques and support plans. At INFOLAB/University of Tilburg in the Netherlands we have developed an experimental methodology for service-oriented design and development that applies equally well to Web services and business processes.

Service orientation utilizes services as constructs to support the rapid, low-cost and easy composition of distributed applications. Key to this concept is the service-oriented architecture (SOA), which is a logical way of designing a software system to provide services to either end-user applications or to other services distributed over a network, via published and discoverable interfaces. A well-constructed SOA can empower a business environment with a flexible infrastructure and processing environment by provisioning independent, reusable automated business processes (as services) and providing a robust foundation for leveraging these services.

In their early use of SOA, many researchers and developers think that they can port existing components to act as Web services just by creating wrappers and leaving the underlying component untouched. Since component methodologies focus on the interface, many developers assume that these methodologies apply equally well to SOAs. Thus, the placement of a thin SOAP/WSDL/UDDI veneer on top of existing applications or components that implement the Web services is now widely practised in the software industry. Yet this is in no way sufficient to construct commercial-strength enterprise applications. Unless the nature of the component means it is suitable for use as a Web service – and most are not – it takes serious thought and redesign to properly deliver a component's functionality through a Web service.

While relatively simple Web services may be built with conventional development methodologies, a service-oriented development methodology is of critical importance to specify, construct, refine and customize highly volatile business processes from internally and externally available Web services.

Our service-oriented design and development (SoDD) methodology provides sufficient principles and guidelines to specify and construct business processes choreographed from a set of internal and external Web services. It takes into ac-count a set of development models (eg top-down, bottom-up and meet-in-the-middle), stresses reliance on reference models, and considers several service realization scenarios, including green-field development, outsourcing, and legacy wrapping in cases where services are assembled out of pre-existing components.

Our service-oriented design and development methodology is based on an iterative and incremental process that com-