Building a context-aware service architecture

Keith Jones 12 December 2008

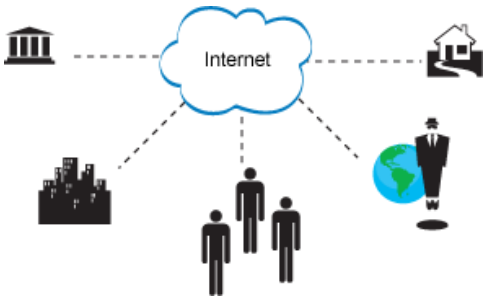
Mobile computing has ignited the idea that the physical and logical context of users can influence the behavior of services they call for. This article reviews some approaches to architecting context-aware services, including context delivery and enrichment, dynamic context-driven service discovery, and invocation.

Context-aware services

For several years the IT industry has been excited by the prospects of both mobile computing and ubiquitous computing. Laptops, Notebooks, PDAs, smart phones and other devices have already freed us from the technological shackles of workstations and desktops. Large scale experimental computing “clouds” and “grids” have provided service to an extraordinary number of computer users.

An interesting aspect of mobile computing is the decoupling of function from the location where it is performed, as shown in Figure 1. Users can play games, maintain accounts, exchange e-mail, access business data, submit product orders, track shipments, and interact with coworkers in real time from home, on the beach, at the airport, in the office, on the shop floor, and so on.

Figure 1. Mobile computing decouples function from location



Until relatively recently, the functions mentioned above were performed at workstations in fixed locations. The decoupling of function from location would not be possible without near-ubiquitous Internet communications.

Certain mobile computing applications have taken advantage of changing user location. For example, geospatial databases are now routinely used to answer queries from mobile computer users for the nearest post office, best realtor, directions to the airport, and so forth. They all take

advantage of information about current location that is available to the mobile application, mostly from sensors reacting to local wireless network towers or GPS satellite signals.

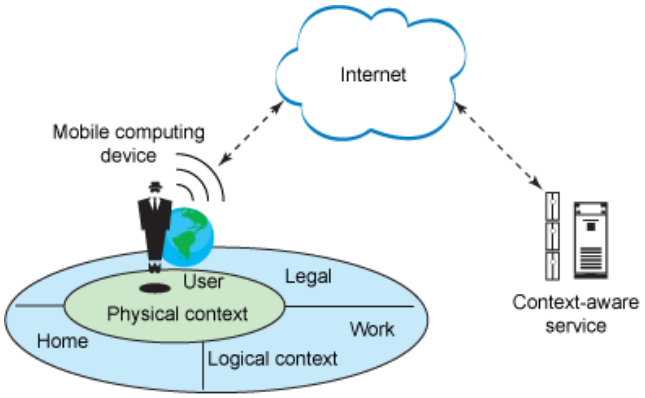
In this article, learn about context awareness, and about architectural approaches for implementing context-aware scenarios.

Becoming context-aware

Information about the current location of a mobile user (or more precisely, about the device being used) is an example of information taken from the physical “context” of that user and their device, as shown in Figure 2.

Recent trends in mobile computing are extending the concept of context to include many other facets of the user’s physical environment. Computing device characteristics such as screen size, communications capabilities, keyboard configuration, accelerometers, GPS sensors, network identity and many others are being added to more accurately characterize the physical context within which applications are being used.

Figure 2. Mobile user context



In more general terms, *context* can refer to any aspect of a situation where an entity (person, place, or computational object) may invoke computational functions. With today's mobile systems, functions would most likely be expressed as calls for service from either local or remote service providers. In this type of scenario, the mobile user and his device becomes the *service consumer*.

Physical and logical worlds in context

Any given context may include information about the physical world (location, movement, temperature, pressure, device characteristics, and so on) and about the logical world surrounding the service consumer. This logical world consists of information about identity, privileges, preferences, and relationships in different domains, such as home, work, family, legal and others. (See Figure 2.) Even historical information about any of these aspects might be included. For example, home address might be included in the service consumer’s logical context, but for certain services prior (historical) home addresses may also be required.

Context information about the physical world can be gathered in real time from sensors embedded in a mobile device, such as a smart phone or PDA. But where does the information about the

logical world come from? The short answer is: it must be gathered directly from the user or deduced from interactions the user has made with service providers over time. Whatever the nature of this information, context may come from disparate sources and has a relatively transient lifetime.

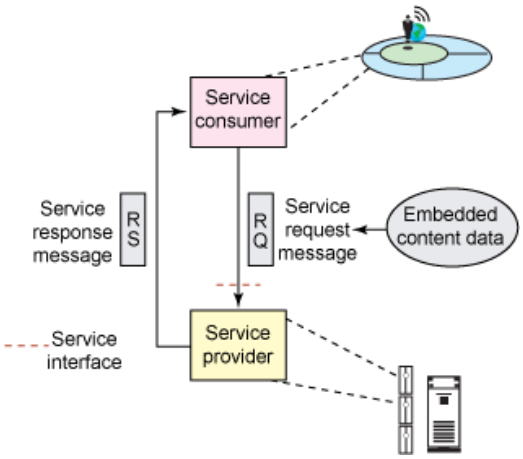
Architectural approaches

When context-aware applications first emerged, they appeared to be meeting largely personal needs for mobile users. Now, business requirements are also emerging. Given the exciting prospect of commercial service providers acknowledging and reacting to contextual information, you can consider several architectural approaches for implementing context-aware scenarios.

Embedded context

The scenario in Figure 3 below is perhaps the simplest architectural approach. But, it is the least flexible and has several drawbacks. The service consumer gathers context information from sources in its local execution environment in unspecified, and probably internal, ways. The context information gathered is then packaged together with details of service requests and sent within service messages to known service providers.

Figure 3. Service architecture with embedded context



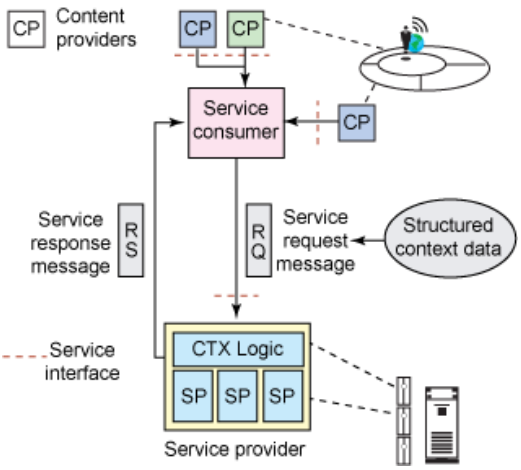
Service providers process the context information, along with request details, by selecting local logic that best meets the user's needs in that context. Response messages, containing functional information and selected context information as defined in the service description (portType), are sent back to the service consumers.

In this scenario, there is really no distinction between context information and functional details at a conceptual level. At logical and physical levels there may be some separation using appropriate (proprietary) XML schema namespaces and types in service descriptions. You might conclude that context information has "polluted" the service interface. Or, you might consider the context information to be bona fide functional details. However, in either case, the context information is embedded and context processing is hidden.

Explicit context

The context scenario in Figure 4 introduces explicit sources for contextual information. Logically, these can be considered context provider (CP) services. The service interface for each context provider encapsulates the nature of the source and the mechanism by which the contextual information is acquired. Not shown in Figure 4 is the possibility that, in larger environments, context providers may be included in a local registry and made dynamically discoverable to local service consumer logic.

Figure 4. Service architecture with explicit context



Once contextual information is gathered by service consumer logic it's packaged in structured schema, with details of the requested function, and sent to known service providers. As in the previous scenario, service providers process the context information, along with the request details, by selecting local business logic that best meets the user's needs in that context. However, Figure 4 shows that service provider logic may be partitioned according to different contextual situations and functional needs behind the service interface.

Some amount of reasoning is applied to context information received by the context consumer logic (CTX logic) before service provider business logic is selected and invoked. This partitioning between context processing and business function is a useful step toward more sophisticated architectural solutions that might be possible in the future.

As before, response messages containing requested functional information, and selected context information defined in the service description (portType), are sent back to the service consumer.

In this scenario, the service consumer logic must be programmed to:

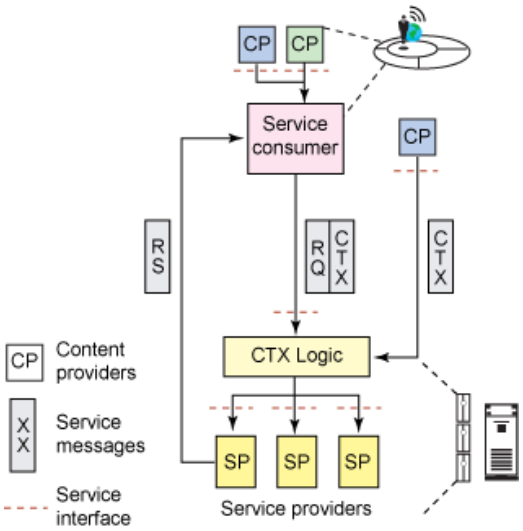
- Send specific required elements of context with a particular service request (as specified in the service description).
- Interrogate specific local context providers to provide those elements.

Service consumer and context provider logic will often be implemented as components in a local mobile device JVM.

Dynamic context-awareness

A more general scenario for context-aware service architecture is shown in Figure 5. The main difference here is the isolation of context processing logic from other components, and the architectural separation of supporting service providers.

Figure 5. Context aware service architecture



This approach to context-aware service architecture introduces the possibility that context processing logic may dynamically enrich a given context by invoking additional context providers, which may be external to the physical world of the service consumer.

For example, from a conference meeting room a company executive requests key performance metrics for an important business process. The context processor receiving the request invokes a context provider to get calendar information for the executive to enrich the logical context for her request. Awareness of the nature of the meeting being held enables the most appropriate service provider logic to be selected, and the most appropriate metrics to be served to the executive at the meeting.

It is important to separate the context provider, context consumer logic, and service provider components when implementing this scenario. Separation allows for:

- Future enrichment of the physical and logical context information over time.
- More sophisticated reasoning about the context.
- Extension of the number and nature of supporting service provider endpoints.

At first, the sophistication of the contextual reasoning introduced might be extremely simple. Over time, reasoning might be introduced to recognize new situations for up-selling or cross-selling of products, or recognizing the need for preemptive intervention in customer satisfaction or emergency scenarios.

Experimental research is currently exploring how the several domains of a context can be described using Web ontology language (OWL), and how they can be processed using inference

engines. These technologies allow for extensibility and new levels of sophistication in processing requests in context-aware service architecture.

Implementing context-aware services

The architectural approach in Figure 3 can be implemented today using mobile device and server technology. For example, many hand-held mobile devices are programmable and already support Java communications with Web servers and application servers accessible through the Internet. Some of these devices have integrated sensors capable of providing useful context such as GPS, fingerprint, and video data.

By using service consumer application code specifically designed for particular mobile devices you can gather context information, and embed it in service request messages that are handled by service provider logic executing in standard application servers. For example, IBM WebSphere Application Servers can be deployed to support this embedded context approach. Access to context data in the mobile user's environment is very similar to the standard approach used today for implementing service architecture.

Next steps

As memory and processing capabilities within mobile devices improve, the capacity to collect context information and package it for consumption by service providers will also improve. Identification of business services, which vary according to the context in which they're invoked, will drive the need to partition context processing logic from basic business function in service providers. Such partitioning is already implementable using the IBM SOA Foundation set of software products.

Once context processing logic is partitioned, and supporting service providers are deployed using an ESB with supporting Service Registry and Repository, it's possible to plan for expanded context-awareness. Incoming context instance-data from multiple related context domains can be explored to identify new business opportunities. Once the new opportunities have been identified in real time, the appropriate supporting business service providers can be dynamically discovered and invoked using standard SOA technologies.