# Keeping computing compatible

September 25 2008

(PhysOrg.com) -- As distributed computing becomes universal, the programs that make devices work really have to work together. European researchers have gone back to basics to create a development toolkit that guarantees this sort of compatibility.

Early in 2006, an EU-funded research group called SIMS, for Semantic Interfaces for Mobile Services, took on the challenge of how to envision, design and develop the next generation of software to power widely distributed and highly interactive devices.

The result – a suite of tools for speeding the design and validation of software and services that are guaranteed to interact smoothly – is now being applied and tested by a team of developers.

When SIMS-inspired services are widespread, says Richard Sanders, the SIMS project coordinator, devices such as smart phones, PDAs, and computers will interact with each other seamlessly, update themselves automatically, and offer users the ability to implement new services that are guaranteed to work from the start.

“If you have communicating software and the communication is important, you want to make sure it works when it interacts with other software,” says Sanders. “SIMS provides the tools to check those scenarios and actually guarantees compatibility.”

## Autonomous and collaborating components

The SIMS researchers based their approach on two key factors that they felt had previously been neglected.

Communication and computation are becoming increasingly collaborative and, at the same time, the programs and components that make the devices that we rely on to work are becoming increasingly autonomous.

To accomplish a goal as simple as delivering a package, multiple agents using a wide range of fixed and mobile devices must exchange a variety of messages. For the package to get to the right place at the right time, every exchange has to produce the desired result.

So, the software components making all those interfaces work have to be compatible.

Unlike a telephone call, where one device attempts to initiate a particular kind of connection with another, most real-world services now involve many loosely interconnected software components running on a variety of devices initiating complex sequences of contacts and utilising many different messaging modes.

Most developers, notes Sanders, still think in terms of a single client and server, where one component takes the initiative and the other responds. “We find this very limiting,” he says. “We’re used to lots of components whose combined behaviour produces a service, and where many of them can take the initiative.”

## Coded for success

To reach their goal, the SIMS researchers had to re-examine the process of service development from the ground up.

“The biggest challenge was to understand the basic concepts and find the right way to explain them to ourselves and others,” says Sanders. “Concepts like what is a service, what is a goal, what is a semantic interface, and how do these relate to software?”

One result of their back-to-basics approach is that the development of a new service starts with a model of what that service should accomplish rather than with computer code.

The model uses semantic interfaces to specify what goals need to be realised and how the components of the system need to behave and interact to bring that about. Semantic interfaces detail, in a highly structured way, what kinds of connections, exchanges and results are meaningful and useful within a particular domain.

Crucially, the ability of components to communicate with and understand each other can be checked within these models, rather than after reams of computer code have been written.

“We can validate that nothing goes bad; that you don’t send me a message that I won’t understand,” says Sanders.

Developers can create computer code to run devices directly from the validated models, code that is guaranteed to work with all the components of the system.

The researchers believe using their approach and tools could head off most of the interaction errors that trip up systems and frustrate users.

In addition, devices could detect when new or improved services become available, and update themselves automatically as they interact without the risk of introducing incompatible software.

Sanders is eager to see SIMS used wherever interactive services and the software that makes them work are being developed. The result he envisages is a dynamic, service-oriented market place that would work far more smoothly and efficiently than today.

“The greatest potential lies in the way it can support a market place with lots of people specifying services and lots of companies making components that implement these services,” says Sanders. “This market place would support the spreading of software in a much more efficient way than you currently see, and without quality and compatibility problems.”

The SIMS project received funding under the ICT theme of the EU’s Sixth Framework Programme for research.

Provided by ICT Results