



From the Editor in Chief...

Do You Know How to Get to Carnegie Hall?

Robert E. Filman • *Research Institute for Advanced Computer Science* • rfilman@computer.org

The older I get, the more I come to understand that everything I know is wrong. (If you ever find yourself doubting that, just acquire a teenager and you will receive continuous reinforcement of the underlying rule.) One of the first principles I learned in software design was service abstraction: you define a service with well-defined interfaces and make calls on that service, which might respond with some answer. (A procedure call is a service abstraction, for example.) Service abstraction argues that the called service doesn't need to know anything other than the provided service parameters to successfully perform its task, the caller knows everything necessary to properly invoke the service provider, and there is exactly one appropriate service-provider behavior for any particular combination of parameters.

Service abstraction is a wonderful concept. It provides a well-defined computational context and allows the service implementer the freedom to concentrate on the problem at hand. Being able to implement something specific while ignoring the real problem's context is most programmers' ultimate goal. Unfortunately, service abstraction provides only a painfully limiting interaction quality. As evidence of this, consider that people don't do it.

Context Awareness

If someone approaches you in Times Square and asks if you know how to get to Carnegie Hall, for example, you don't answer, "Yes." Rather, you take account of the question's context. You might consider the weather or seek information (or infer from age and dress) on whether the questioner prefers the subway, a taxi, or walking. People are context-aware in their service responses – and more con-

cerned with intent than literal interpretation.

The Internet and the emerging plethora of communicating devices are pushing us toward context-aware applications. We want information applications that can reply in a variety of natural languages, with information customized to the communication channel's bandwidth, the user's particular privileges and preferences, and the client's characteristics, including location, screen size (or existence), linguistic preferences, and ability to handle alternative communication mechanisms (sound generation, speech understanding, or haptic feedback, for example). And we don't want to send all that information with every request. The Web page that wants to answer "yes" doesn't need to be as careful about phrasing its response as the one that can send an interactive color map of midtown Manhattan. Future systems will need to be able to have conversations.

What Does Context Awareness Enable?

Mobile, ubiquitous applications are the ones most in need of context awareness because they have the most uncertain and variable contexts. The direction-providing system will need to know where you are (either from your internal Global Positioning System or your physical environment's location pings) and what facilities it can use to describe how to get where you want to go. The automated art museum tour guide needs to know not only what you're looking at and which language you speak, but also what you already know and how much attention you're paying to what it's telling you. Emergency medical response will depend not only on your physical location, the

weather, your medications, and medical history, but also on which insurance card is in your pocket. Augmented reality games will want to know where you are, what you're looking at, and your pulse rate, and they will want to communicate some distillation of that information about your teammates and opponents.

Currently, context awareness is implemented in an ad-hoc manner for each application that needs it. Software architecture has progressed to the point of content-free communication mechanisms (such as TCP/IP, Corba, HTTP, Web services) and content-free data-structuring mechanisms (such as XML). Certain kinds of communications semantics are built into certain communication mechanisms (security in Corba and preference parameters in HTTP, for instance). Domain-specific data semantics are sprouting wildly in the XML space. The challenge for system developers is to generate generic context-aware application architectures. This includes defining a mechanism for representing contexts, specifying a uniform mechanism for exchanging and interpreting contextual information, and doing it in a way that appropriately minimizes (but does not inappropriately eliminate) the need for human intervention.

Practicing Context-Awareness

Someday, we might include in our software curriculum the principle that the foundation of intermodule communications is context-based conversation. That's likely to include the idea that requesting a service implicitly carries some handler for inquiring for more information about the request and requestor; that callback might present an interface for handling the "why do you want to know?" responses. (This interface itself would use contextual data such as the caller's location, preferences, and what they are working on.) Around this structure we will need to develop data standards for the various types of contextual information that could be exchanged. After all, if you're going to have a conversation, it helps to have a common language.

And do you know how to get to Carnegie Hall? Practice, my friend, practice. ☐

How to Reach IC

Articles

We welcome submissions about Internet application technologies. For detailed instructions and information on peer review, *IEEE Internet Computing's* author guidelines are available online at www.computer.org/internet/author.htm.

Letters to the Editor

Please send letters, including reference to articles in question, via e-mail to internet@computer.org.

Reuse Permission

For permission to reprint an article published in *IC*, contact William J. Hagen, IEEE Copyrights and Trademarks Manager, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08855-1331; w.hagen@ieee.org. Complete information is available at www.computer.org/permission.htm. To purchase reprints, see www.computer.org/author/reprint.htm.

EDITOR IN CHIEF

Robert E. Filman • rfilman@arc.nasa.gov

ASSOCIATE EDITOR IN CHIEF

Li Gong • li.gong@sun.com

EDITORIAL BOARD

Jean Bacon • jean.bacon@cl.cam.ac.uk
Miroslav Benda • miro@amazon.com
Elisa Bertino • bertino@dsi.unimi.it
Scott Bradner • sob@harvard.edu
Siobhán Clarke • siobhan.clarke@cs.tcd.ie
Fred Douglass • f.douglass@computer.org
Stuart I. Feldman • sif@us.ibm.com
Ian Foster • foster@cs.uchicago.edu
Michael N. Huhns • huhns@sc.edu
Leonard Kleinrock • lk@cs.ucla.edu
G.S. Kuo • gskuo@ieee.nccu.edu.tw
(IEEE Communications Society Liaison)
Doug Lea • dl@altair.cs.oswego.edu
Frank Maurer • maurer@cpsc.ucalgary.ca
Daniel A. Menascé • menasce@cs.gmu.edu
Chris Metz • chmetz@cisco.com
John Mylopoulos • jm@cs.toronto.edu
Peter Norvig • peter@norvig.com
Charles J. Petrie • petrie@nrc.stanford.edu
(EIC emeritus)
Krithi Ramamritham • krithi@cse.iitb.ac.in
Ravi Sandhu • sandhu@gmu.edu
Munindar P. Singh • singh@ncsu.edu
(EIC emeritus)

STAFF

Lead Editor: Steve Woods
swoods@computer.org
Group Managing Editor: Gene Smarte
Staff Editors: Scott L. Andresen,
Jenny Ferrero, and Kathy Clark-Fisher
Production Assistant: Monette Velasco
Magazine Assistant: Hazel Kosky
internet@computer.org
Graphic Artists: Larry Bauer, Alex Torres
Contributing Editors: David Clark,
Greg Goth, Keri Schreiner, Joan Taylor,
Publisher: Angela Burgess
Membership/Circulation Marketing Manager:
Georgann Carter
Business Development Manager: Sandy Brown
Advertising Supervisor: Marian Anderson

CS Magazine Operations Committee

Jean Bacon (chair), Thomas J. Bergin, Pradip Bose,
Doris L. Carver, George Cybenko, John Dill, Frank E. Ferrante,
Robert E. Filman, Forouzan Golshani, Rajesh Gupta,
Warren Harrison, Mahadev Satyanarayanan,
Nigel Shadbolt, and Francis Sullivan

CS Publications Board

Rangachar Kasturi (chair), Jean Bacon,
Mark Christensen, George Cybenko,
Gabiella Sanniti di Baja,
Lee Giles, Thomas Keefe,
Dick Kemmerer, and Anand Tripathi

