The big picture idea for SOAP involves decontextualized requests that maintain transactional integrity in an asynchronous environment. In the business realities of real systems, however, the context has to be put back into the request. First, we must associate identity with the request, and then credentials, and then sign the message and encrypt sensitive information, and on and on. The "simple" burden of issuing SOAP requests gets encumbered by the interaction style and our business needs. If someone in an organization wants to retrieve some information, why can't they just ask for it? And, once these questions have been answered once, why do 10 (or 100 or 1,000) people asking the same question have to put the same burden on the backend systems every time they issue the same query?

These questions highlight some of the abstraction problems that exist with the conventional Web Services technology stack and offer at least a partial explanation for the WS-Dissatisfaction that pervades the halls of IT departments around the world. These technologies are implementation techniques for decomposing our invocation of behavior into orchestrated workflows of services, but we cannot express the full vocabulary of an organization's needs with only the notion of services. We lose the ability to identify and structure information out of the context in which it is used in a particular invocation. We lose the ability to simply ask for information without having to understand the technologies that are used to retrieve it. When we tie ourselves to contract-bound requests running on a particular port of a particular machine, we lose loose-coupling and asynchronous interaction patterns as well as the ability to embrace changing views of our data. Without the ability to uniquely identify the data that passes through our services, we lose the ability to apply access control at an information level. This complicates our already untenable problem of protecting access to sensitive, valuable, and private information in an increasingly networked world.

SOAP and WSDL are not the problems here, but neither are they complete solutions. We will very likely use SOAP in the doc/lit style in the resource-oriented architectures I am about to describe; we just do not have to accept them as the only solution. Nor will we always need to advertise that we are using them behind the scenes if there is no need to do so. In order to take this next step, we need to look at the Web and why it has been so successful as a scalable, flexible, evolvable information-sharing platform. Implementation details are often not relevant to our information consumers.

The Web

The prevailing mental model for the Web is document-centric. In particular, when we think about the Web, we think about consuming documents in web browsers because that is how we experience it. The real magic, however, is the explicit linkage between publicly available information, what that linkage represents, and the ease with which we can create windows into this underlying content. There is no starting point, and there is no end in sight. As long as we know what to ask for, we can usually get to it. Several technologies have emerged to