Components, Services, and Contracts

An SOA is a system of distributed software components, and SOA policies are declarative statements of a component's relevant behavioral aspects. A service-oriented model is founded on the idea that designers can represent an organization's computing environment as a set of distributed software components. A component is a software building block that exposes one or more services to potential requesters. These definitions are consistent with the service component model underlying the Web Services Business Process Execution Language (WS-BPEL, or BPEL; http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html) and the Service Component Architecture (SCA) specifications (www.osoa. org/display/Main/Service+Component+Architecture+Specifications), the two Web services specifications that define the mechanisms for assembling SOA components into new services.

A component's full specification requires internal and external definitions. The internal specification determines the component's implementation and execution environment. A component might be a BPEL script executing on a compliant BPEL engine, for example. From an SOA point of view, the external definition is the component's most critical part because it determines its interaction capabilities with other components and is thus the basis for the component's composability. This external specification is the component's contract.

In general, SOA policy refers to the component contract's nonfunctional aspects, but is typically restricted to the technical aspects that guide component interaction. Depending on the interaction context, a component's contract can include different sets of functional or nonfunctional aspects. In some cases, it is possible to infer from the environment required information that might be missing from the contract.

Contracts support two major goals of service-oriented systems. The first is the principle of third-party use, which requires that an SOA component be accessible from different management and organizational domains. Explicit contracts enable third-party access because they help component users understand the technical and business requirements of the service they access. The second goal is automatic integration between service components, a goal not yet attained but core to the SOA vision. The challenge of automatic integration motivates the intense focus on service discovery and automatic composition technologies.

Because of the focus on automating the integration process, SOA contracts have been defined in machine-readable formats, even at the expense of human-readable documentation. For this reason, the main specifications defining component contracts are encoded as XML documents, geared toward their use in code-generation tools, repositories that can be queried, and the like. The Web Services Description Language and Web Services 1.5 Policy (www.w3.org/TR/2007/REC-ws-policy-20070904) are the two main XML formats used today to encode service descriptions for functional and nonfunctional service contracts, respectively.

The Web services stack builds on the Simple Object Access Protocol (SOAP; www.w3.org/TR/soap), an extensible message-oriented interaction protocol that can be supported on top of multiple transport protocols such as HTTP, raw TCP/IP, or proprietary messaging. QoS protocols such as message-level security or reliable delivery are extensions of SOAP. The SOAP message can include QoS headers, such as a transaction context header, to support specific QoS properties. SOAP's extensibility is a central feature of the Web services stack and key to supporting runtime QoS.

Services describe their support for or the requirement of QoS protocols by attaching a Web services policy to their WSDL descriptions, thus extending their functional contract with nonfunctional attributes. Service clients that intend to access

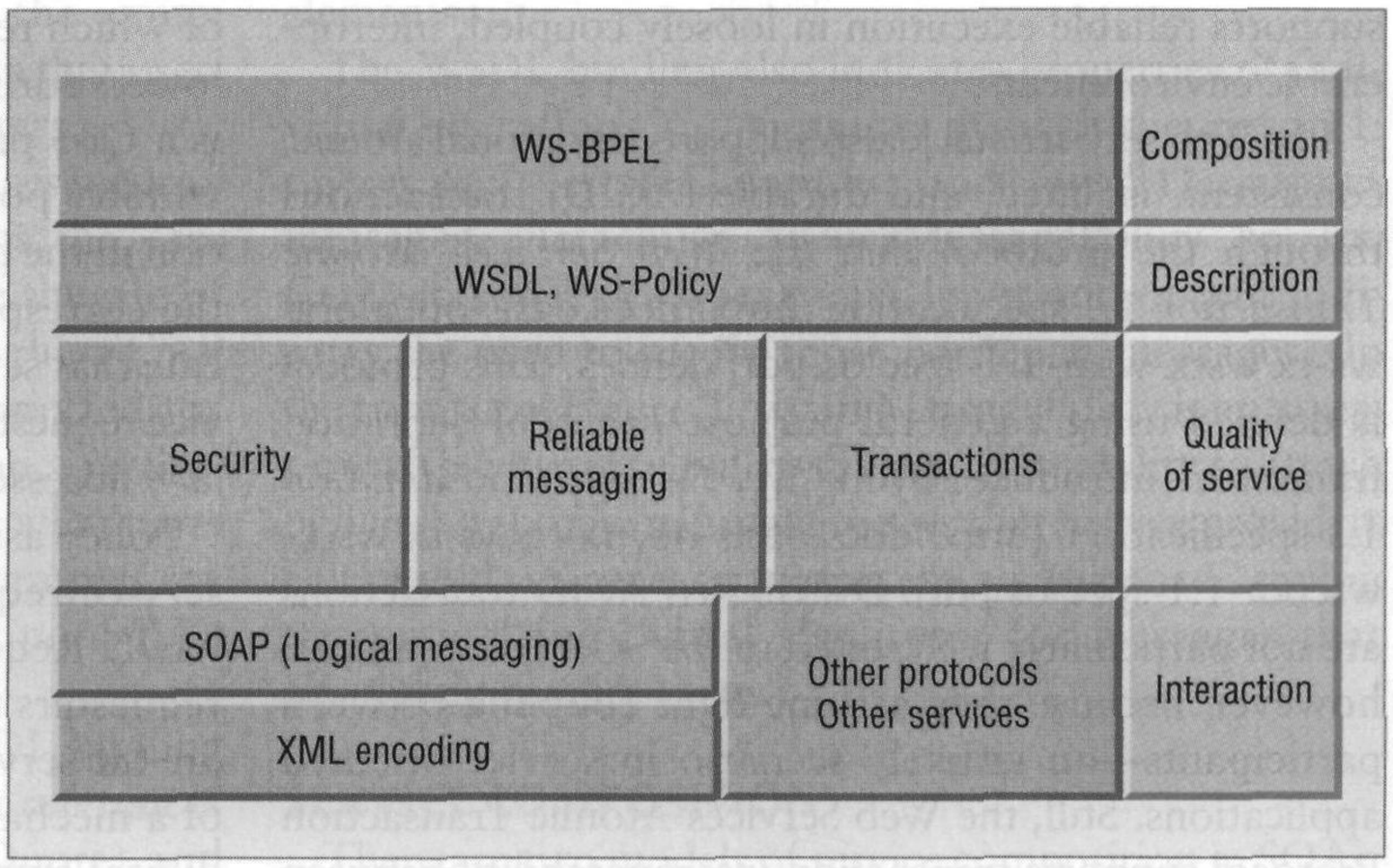

Figure 1. A simplified view of a Web services stack. Including service composition specifications like the Business Process Execution Language takes the core stack a step beyond strict interoperability and into SOA-centric implementation.