concurrently takes part. This guarantees the Web service is properly specified and ready for execution per composition scenario. The execution perspective is about looking for the computing resources on which a Web service operates, and monitoring the capabilities of these computing resources so the Web service's requirements are constantly satisfied. It has been argued that the availability of the computing resources affect the capabilities of a Web service [6]. Finally, the preference perspective is about ensuring that user preferences (for example, execution time at 2 P.M.) are integrated into the specification of a composite service.

In Figure 1, participation, execution, and preference perspectives are all interconnected. First, deployment connects participation and execution perspectives, and highlights the Web service that is executed once it accepts participating in a composition. Second, tracking connects execution and preference perspectives, and highlights the significance of monitoring the execution of a Web service so user preferences are properly handled. Finally, customization connects preference and participation perspectives and highlights the possibility of adjusting a Web service so it can accommodate various user preferences.

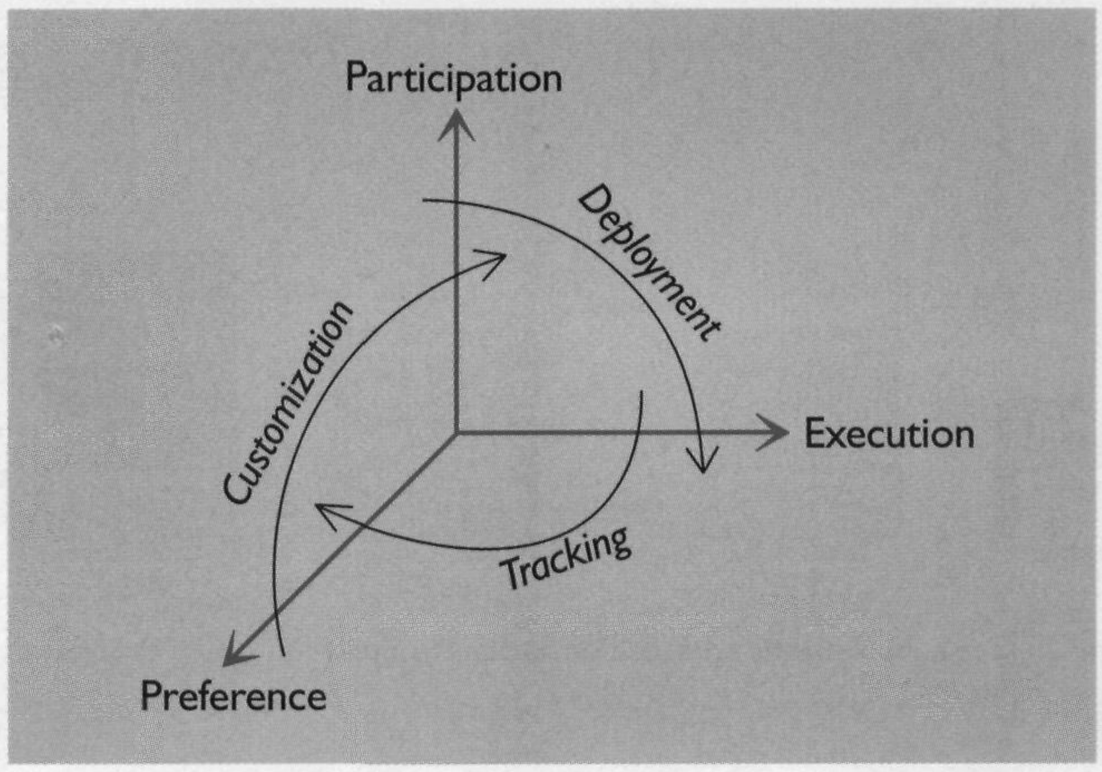

Figure 1. Context organization of a Web service.

The integration of context into Web services composition ensures that the requirements of and constraints on these Web services are taken into account. While current composition approaches rely on different selection criteria (for example, execution cost and reliability), context supports Web services in their decision-making process when it comes to whether accepting or rejecting participation in a composition. Moreover, context permits tracing the execution of Web services during exception handling. It would be possible to know what happened and what is happening with a Web service at any time. Predicting what will happen to a Web service would also be feasible in case the past contexts (that is, what happened to a Web service) are stored. Web services can take advantage of the information that past contexts cater, so they adapt their behavior for better actions and interactions with peers and users.

On contextualizing Web services composition. It is known that the composition of Web services using current standards is only conducted at the level of message interactions, which is hardly sufficient since composition must also be conducted at the level of message semantics. Because independent bodies develop Web services, achieving semantic composition is concerned with various conflicts that arise when the same concept has different data structures, the same concept has different meanings in different use scenarios, and different concepts have the same meaning.

A contextual semantic composition of Web services is subject to the satisfaction of two conditions. The first is that Web services must agree on the meaning of the exchanged data; the second is that semantic-data conflicts must be resolved automatically using the information that context caters. Figure 2 illustrates the stages that enable reaching a contextual semantic composition of Web services. In a conventional composition (Figure 2a), WSDL is used to describe Web services' interfaces. During Web services interaction, any data conflict of type value is resolved at the level of the receiver Web service. For example, upon reception of a value V in USD from WS_1, WS_2 explicitly converts V into V' in local currency.

In a semantic composition that excludes the context of the exchanged data (Figure 2b), Web services' interfaces are semantically described using high-level languages like OWL-S and WSDL-S. While these languages can handle data conflicts of type structure and semantics between Web services' interfaces, they cannot handle data conflicts of type value. A value V that WS_1 outputs must be explicitly converted into V' in the body of WS_2. The conversion function is tightly attached to WS_2; this prevents reusing WS_2 in other composition scenarios.

To achieve semantic composition that includes the context of the exchanged data (Figure 2c), context information must be provided. This context information is any metadata that explicitly describes the meaning of the data to be exchanged between Web services. When Web services' inputs and outputs are explicitly described using metadata, they can be automatically transformed from one context to another during Web services execution. By automatically, we mean the conversion function is not embedded into the body of any Web service. This function is loosely attached to Web services and becomes, thus, an active