**Service composition model**

The use of Web services as a standard technology facilitates seamless business-to-business (B2B) interaction and the process of integrating various systems and applications. It enables mass knowledge distribution by providing a standard way of exposing data sources and applications/systems as Web services. These services can be created/generated, updated, and composed at runtime because they are loosely coupled. Web service composition provides the ability to compose services by manually or automatically generating a service composition plan in order to achieve the business goal, resolve a scientific issue/problem or provide new service functionality. They are reusable services that can be used to implement a business process. Web service composition can be achieved via two models: dynamic and static. This paper introduces taxonomy of Web service composition models and approaches, and provides a survey of Web service composition models and their approaches. Additionally, we present comparisons of different models and approaches for each model.

we looked at how to maximize the potential of an SOA solution by identifying services that are business-relevant. We designed the service topology based on the business requirements and connected the services back to the service collaborations that represented requirements contracts that the service solution must fulfill.

we modeled the details of the service specification. A service specification defines everything potential consumers of the service needs to know to decided if they are interested in using the service and exactly how to use it. It also specifies everything a service provider must know to successfully implement the service.

we modeled the realization of the service specifications resulting in service providers: Invoicer, Productions, and Shipper. Each of these components provides services and capabilities according to the service specifications. Each provided service operation has a method that describes how the service is actually implemented. A method can be any UML behavior, including an Activity, Interaction, StateMachine, or OpaqueBehavior. The choice is up to the modeler.

"Modeling SOA: Service implementation," the final article in this series, uses the IBM® Rational® Software Architect UML-to-SOA transformation feature to create a Web services implementation that can be used directly in IBM® WebSphere® Integration Developer to implement, test, and deploy the completed solution.

Web Service Composition creates new composite Web Services from the collection of existing ones to be composed further and embodies the added values and potential usages of Web Services. Web Service Composition includes two aspects: Web Service orchestration denoting a workflow-like composition pattern and Web Service choreography which represents an aggregate composition pattern. There were only a few works which give orchestration and choreography a relationship. In this paper, we introduce an architecture of Web Service Composition runtime which establishes a natural relationship between orchestration and choreography through a deep analysis of the two ones. Then we use an actor-based approach to design a language called AB-WSCL to support such an architecture. To give AB-WSCL a firmly theoretic foundation, we establish the formal semantics of AB-WSCL based on concurrent rewriting theory for actors. Conclusions that well defined relationships exist among the components of AB-WSCL using a notation of Compositionality is drawn based on semantics analysis. Our works can be bases of a modeling language, simulation tools, verification tools of Web Service Composition at design time, and also a Web Service Composition runtime with correctness analysis support itself.