Chapter 3
A Foundational Service Framework

Despite the proliferation of Web services, the field of service research is still in its infancy, where there has been little foundational work to date. In this chapter, we present the foundational service framework that helps address some fundamental research issues on Web services [44, 74]. The presented framework offers a systematic approach that enables users to achieve an optimal Service Execution Plan (SEP) by submitting declarative service queries. Figure 3.1 shows the overall architecture of the service query optimization framework. At the bottom, we have the service space for a given application domain that consists of the actual service providers, a.k.a., service instances. The integration layer helps resolve the discrepancy of the service schemas and provides a uniformed view of the service space. It can achieve this by applying the existing data integration techniques [40], which are not the focus of this work. The QoWS manager deals with the collection of quality related information from the service instances. The quality data will be used by the query optimizer to make selection from the competing service providers. The service model captures a set of key features of Web services that lay out a foundation for service query specification, processing, and optimization. The service calculus enables users to use declarative service queries to locate Web services, which is more precise and reliable than the keyword based search. The query processor generates SEPs which can be used by the users to invoke services.

The remainder of this chapter is organized as follows. In Section 3.1, we describe a scenario that will be used as a running example through this chapter. In Section 3.2, we present the formal service model. In Section 3.3 and 3.4 we describe the service calculus and the service algebra, respectively. In Section 3.5, we present the implementation of the algebraic operators. In Section 3.6, we propose a QoWS model, which serves as the cost estimation criteria in the QoWS optimization. Based on the model, we propose two optimization algorithms. We present an analytical model in Section 3.7 and conduct experimental studies in Section 3.8.
3.1 Case Study: Car Brokerage

As a way to illustrate this work, we use an application from the car brokerage domain (see Figure 3.2). A typical scenario would be of a customer, say Mary, planning to buy a used car having a specific model, make, and mileage. She naturally wants to get the best deal. Assume that Mary has access to a Web service infrastructure where the different entities that play a role in the car purchase are represented by Web services. Examples of Web services that need to be accessed include Car Purchase (CP), Car Insurance (CI), and Financing (FI). A single Web service may provide multiple operations. Different operations may also have dependency relationships. For example, the paymentHistory and financingQuote operations are both offered by the financing service. The latter operation depends on the former operation, i.e., the payment history decides the financing quote. We also anticipate that there will be multiple competitors to provide each of the services mentioned above. It is important that the users’ quality requirements be reflected in the service query as criteria for service selection. To purchase an entire car package, Mary would first like to know the price quote of the selected car and the vehicle history report. She then needs to get the insurance quote. Finally, since Mary needs the financing assistance, she also wants to know the financing quote. In addition, Mary may have special requirements on the quality of the service operations. For example, she wants to spend less than 20 dollars to get the vehicle history report.

The proposed service query optimization framework is designed to help Mary with her car package purchase. Mary can specify her car package through a declarative service query. The service query is formed based on the service model. The declarative service query is a service calculus expression. It specifies the functionalities that the user wants to retrieve in terms of service operations. There is no need for the user to have the knowledge of the dependency constraints between service operations. In addition, the