Developing Web Applications

Sabah Al-Fedaghi
Computer Engineering Department - Kuwait University
sabah@alfedaghi.com

Abstract

One approach to developing service-oriented Web applications is to transform high-level business models to a composition language that implements business processes with Web services. Object-oriented analysis and design and UML-based diagrams are typically used in the software development process. In this paper, we propose using flow as a fundamental notion underlying understanding of activities in Web Services. We discuss the development of business processes through introduction of a conceptual model as a framework for design. We scrutinize current modeling used in transformation methodologies, and then introduce a flow-based conceptualization of services through case studies with a high-level business description.

Keywords: Conceptual model, service-oriented Web applications, software development process, object-oriented analysis.

1. Introduction

Service-Oriented Computing uses “services as fundamental elements” [14, 15].

Service-Oriented Computing … utilizes services as the constructs to support the development of rapid, low-cost and easy composition of distributed applications. Services are autonomous, platform-independent computational entities that can be used in a platform independent way… Any piece of code and any application component deployed on a system can be reused and transformed into a network-available service… Services are most often built in a way that is independent of the context in which they are used. This means that the service provider and the consumers are loosely coupled [8].

Web Services technology is based on the concept of service-oriented computing. Web services are standards that integrate Web-based applications through connecting and sharing of business processes across the network where applications of different vendors, languages, and platforms communicate with each other and with clients.

Web applications refer to applications accessed via Web browser over a network and developed using browser-supported languages (e.g., HTML, JavaScript). For execution, Web applications depend on Web browsers and include many familiar applications such as online retail sales, online auctions, and webmail.

Web applications are needed in the area of business-to-business interaction over networks, e.g., for overseas companies that outsource projects to each other. The adoption of a Web applications infrastructure can provide vital processes such as transfer of funds and updates of pricing information.

Because of the complexity of service systems, analysis of each component and subsystem becomes more challenging. In the field of Web engineering, the need exists for methodologies for the development of Web services. Web Services provide tools
for developing and implementing business processes, e.g., Web Services Description Language (WSDL) and Business Process Execution Language (BPEL).

Services hold the promise of moving beyond the simple exchange of information … to the concept of accessing, programming and integrating application services … The end result is that it is then easier to create new composite applications that use pieces of application logic and/or data that reside in the existing systems… This represents a fundamental change to the socio-economic fabric of the software developer community that improves the effectiveness and productivity in software development … ([15], reference to [13]) [Italics added].

Software development methodologies play an important role in this technology. According to [15]:

Object-oriented and component based development… cannot be blindly applied to [service-oriented architecture] and Web services as they do not address the key elements of [service-oriented architecture]: services, flows of information and components realizing services … One of the main challenges in the development of Services Oriented systems is the provision of methodologies that support the specification and design of compositions of services. [Italics added]

In this paper, we discuss a model that enhances other methodologies used for developing Web services. The focus of our model is on the analysis of the flow streams of all “things that flow” in different subsystems.

This proposed flow model is a promising approach to software development in service-oriented applications. Flow-based conceptual models can reflect high-level design components of Web service and e-business solutions produced early in the application development lifecycle. The models can be utilized by business managers and analysts to trace transformations of models used by software developers. They also provide a means of communication to promote collaboration and standardization.

2. Motivating Example

Web applications require a comprehensive approach that embraces many aspects, including technical, organizational, and legal/philosophical dimensions. Hence, information processing methods, techniques, and tools have been extended to support development of applications of this kind, e.g., Object Oriented Web Solutions. Conceptual modeling methods have been used to abstractly describe requirements for software development processes for the Web; for example, use cases and scenarios are applied to model functional requirements.

De Castro et al. [7] defined a method for development of service-oriented Web applications that starts “from a high level business model … that simplifies the mapping to a specific web service technology.” The method uses “extended use cases” to arrive at a Service Process Model. EclipseCon [8] illustrates the methodology in the following example.

Example: A conference management system is required that includes three different services: submit an article, display submitted articles, and edit the data of an author, in addition to log-in or registration services. Figure 1 shows a partial “use case overview” for this conference management system given by EclipseCon [8]. It contains <<include>>, which indicates that the behavior of the included use case is inserted into the one including it, and <<extend>>, which specifies how and when the behavior defined in the extended use case can
be inserted into that behavior. The Service Process Model is then developed. “Each complex service identified in the previous model is mapped to an activity, while the basic services that it uses are represented as service activities” [16].

![Diagram](image)

**Figure 1. Use Case Overview, Partial View from (From [8]).**

According to Vara Mesa [16], a business service is defined “as a complex functionality offered by the system, which satisfies a specific need of the consumer.” The conference management system shown in Figure 1 includes the services “Submit article,” “View submitted articles,” “Edit author data,” “Log-in,” and “Register.”

3. **Scrutinizing UML-Based Methodology**

We observe that Figure 1 contains conceptually arbitrary services. The author (a person) is directly connected to services “Submit article,” “View submitted articles,” and “Edit author data.” The connection seems to mean that these three services are available to authors. Nevertheless, “availability” is not a notion that should guide the classification of services at this level. The modeling should be guided by a conceptual hierarchy of relationships of services. “Submit article” and “View submitted articles” are article-related services, while “Edit author data” is related to a different artifact (to be explained later).

The design in Figure 1 lacks criteria for constructing a conceptual picture of services showing the hierarchy of these services. Our methodology, introduced in the following sections, involves a flow of artifacts such as articles and author data. In this case, the first-level relationship ought to be between authors and *conference service* and contain *article service* and *author data service*.

The UML activity diagram of the example also reflects conceptual vagueness. It mixes services (e.g., “Edit author data”) and other procedural activities such as downloading of files. Conceptually, this is analogous to mixing shop services such as “selling vegetables” and “selling medicine” with such activities as “weighing items.”

Furthermore, the relationships between the three services and “Log-in” appear odd depicted at the same level in the conceptual picture. “Log-in” is a separate service that facilitates “entering” the services area, but in Figure 1 it is modeled as being the same as any other service.

We next introduce a proposed alternative conceptual description.
4. Flowthing Model

The Flowthing Model (FM) has been used in several applications, including software requirements and privacy in RDF [1, 2, 3, 4, and 5]. This section provides a review of the basic model as it has been described in other publications.

A flowthing model is a uniform method to represent things that “flow,” i.e., things that are received, processed, created, released, and transferred. “Things that flow” include information, materials (e.g., manufacturing), money, etc. In economics, “goods” can be viewed as flowthings that are received, processed, manufactured (created), released, and transported (transferred). Information is another type of flowthing that is received, processed, created, released, and transferred. The flow diagram of flowthings is shown in Figure 2. Sample flow systems are shown in Figure 3.

The flowthings in Figure 3(a) are physical persons flowing through an air travel system. In this situation, there is no “creation” stage. If it were an information system of “travellers’ records” then the flow of “records of persons” would be as shown in Figure 3(c). Figure 3(b) shows the flow of transported materials.

**Figure 2. Flowthing Flow Diagram.**

**Figure 3. Sample Flow Systems.**

The flowthings in Figure 3(a) are physical persons flowing through an air travel system. In this situation, there is no “creation” stage. If it were an information system of “travelers’ records,” then the flow of “records of persons” would be as shown in Figure 3(c). Figure 3(b) shows the flow of transported materials.
Different types of flow can trigger each other. To illustrate, consider the description of interaction between customers and supplier shown in Figure 4 [6]. In the context of the flow model, the “orderGoods” arrow can be interpreted as flow of orders. The “makePayments” arrow represents two types of flows: invoices and money.

![Figure 4. Typical Representation.](image)

Accordingly, Figure 5 shows the flow model description of the interaction.

![Figure 5. Flow Model Description of a Sample Customer/Supplier Transaction.](image)

The customer creates an order that flows to the supplier, triggering flow of an invoice. The flow of an invoice to the customer triggers a flow of money. Upon receiving an invoice, the customer creates money (e.g., a money order) that flows to and is received by the supplier.

In this scenario, flows are coordinated, but do not “mix” with each other. It is a commonsense approach, comparable to designing a house, where electricity, water, and gas flows appear on the blueprint as separate systems. Electricity may trigger the flow of gas; however, each flow is specified separately. Even in a diagram that includes electrical lines and gas pipes, they are represented differently (e.g., different types of arrows).

5. Using the Flow Model to Develop Service-Oriented Web Services

The flow model is unique in the sense that it can be used to capture core “information flows” and their relationships, in contrast to other conceptual models based on information entities/relationships such as the entity-relationship model and UML-based conceptual model.

The fundamental concept of entities/relationships models is centered on entities and relationships: entity types, properties, entity instances participating in relationships, entity sets, instances of relationship types, relationship sets, etc. The flow model uses flow of flowthings as a fundamental notion. “Flows” are first identified, even before
specifying the structure of flowthings and their relationships. This seems to be a commonsense approach, just as in solving a problem, an agent examines the “streams” that lead to the solution. This is most clear in certain applications:

- Before deciding who is traveling, a travel agent searches for a route from city of departure to city of arrival.
- Before identifying product or customer details, a Web sales agent determines ways of delivery, e.g., only within North America.
- Before targeting an enemy post, a pilot examines the flight route.

Entities/relationships-based models obstruct early phases of conceptualizing this stream. The mere concept of an entity is usually manifested in different forms in preparation for mapping, representation, and binding: objects amenable to implementing, transformation into flows (e.g., XML streams), conversion into in-memory structures (e.g., lists), etc. Object-oriented methodologies have introduced an additional abstraction layer for entities. Still, entities precede, conceptually, the flow of flowthings. In this methodology, solving the problem of traveling from Chicago to Lhasa, in Tibet, begins with determining the details of travelers and their relationships before the path between the two cities is checked.

The flow model utilizes “flow” as a fundamental concept for solving problems. Many flow mechanisms exist: flowcharts, workflow, data flow, cash circular flow, flow diagrams, process, and flow diagrams. Ordinary programming flowcharts depict fixed patterns of flow of control (sequential, irritation, etc.). Flow is a basic concept in modeling systems, including Web services applications; however, it has never been explicitly singled out as a foundation of modeling.

In the next section, we discuss the advantages of utilizing flow-based methodology in a case involving a high-level business model that simplifies mapping to a specific Web services technology. Identifying flows provides an abstraction layer of architecture for logical (e.g., messaging) and physical enterprise (e.g., services) businesses.

For example, “orders” are logical flowthings that are created, received, processed, released, and transferred. The flow model description supplies transformation and routing of these orders. Business processes involve a flow of orders that includes purchasing, and management and inventory are modeled in one coherent picture of flows that trigger each other. The purchasing process may involve several flowthings, including information. The inventory process may involve such flowthings as information and actual materials.

In such a scenario, orders are realized “physically” (in the sense of computer jargon) as messages (exchange of information between processes) that can be built upon the same description. Figure 6 shows this correspondence between the logical (description of processes in reality) and physical descriptions (implementation in information system) of this example. To save space, the figure is limited to the inventory process, where it is assumed that some type of assembly is performed before the product is delivered (e.g., toys). The left-hand side of Figure 6 includes two flows: orders and products. The right-hand side represents the corresponding conceptual model used early in the application development lifecycle. On this side, instead of products, the flowthing is information that flows in the information system (IS) and is actualized as a software shadow of the (real) flow of materials. An order triggers the following:

**Assembling**: actualized in IS by signals/information to the assembler to start assembling products.

**Releasing**: actualized in IS by receiving a “finished” signal from assembler.

**Transporting**: actualized in IS by printing address, informing management, etc.
Thus, the right-hand side of Figure 6 is expanded to implement control, monitoring, and tracking in the logical process of inventory upon the arrival of new orders.

Figure 6. Correspondence between Logical Description and its Shadow Description used in Application Development.

6. Reconsidering the Conference Management System

Consider again the development of a conference management system discussed earlier. Viewing it from the perspective of flowthings, we can identify four types of services:

- **Manuscript service**: This service provides uploading, viewing, and editing of manuscripts.
- **Manuscript record service**: This service allows creation of a manuscript record (information about a manuscript such as title, number of words, etc.) and viewing and editing of manuscript records.
- **Author record service**: This service allows creation of an author record (information about author, such as name, affiliation, etc.) and viewing and editing of authors’ records.
- **Service request service**: This service allows selection of an available service.

Services are identified by their flowthings: manuscripts, manuscript records, author records, and service requests. This contrasts with the approach described earlier, where flowthings are fragmented and services are disorganized hierarchically and mixed with other activities.

In the flow model, the “object” (flowthing) flow forms a nucleus of classification and hierarchy of that classification. This is analogous to commonsense conceptualization, such as, for example, a pharmacy providing services related to medicine flow, and a factory providing services related to its products flow. Similarly, a “manuscript service” in our scheme is a category that appears before descriptions of different types of processes included in this service.

Identifying flowthings leads to identifying services, and it also leads to specifications of flows. Each type of service is a realization of flows starting from the flow of requests for that service. A “new manuscript service” request can be mapped as shown in Figure 7, leading to the following actions:

- **Receiving** and **processing** a request,
- Triggering a manuscript service that includes
  - **Receiving** a manuscript,
  - Triggering a manuscript record service that includes
    - **Creating** a manuscript record.
A service is a flow that receives, processes, creates, releases, and transfers flowthings. Flowthings change their states according to the stages of the flow model. This is in line with Hill's [10] definition of services as “a change in condition or state of an economic entity (or thing) caused by another.”

Figure 7. A Service Request is a Realization of Flows.

A conceptual description of this example is given in Figure 8. The figure includes four flows. It can be used to specify procedures such as the one described above for “new manuscript service.” The numbers in circles identify triggering points.

Figure 8. Flow-Based Conceptual Description of a Conference Management System.

7. Composition System: MARKETPLACE

A more comprehensive example that demonstrates the conceptual orderliness of the flow-based approach is the marketplace design detailed by Foster et al. [9]:

The marketplace consists of a series of requests and replies, formed by the offering and requesting of products, request and offer of price for the products and confirmation of an iterative negotiation phase which determines if an agreement of price for product is made… The marketplace provides three stages to a negotiation. Firstly, a product may be either offered or requested. The message is passed from the seller or buyer role respectively, and
is received by the marketplace service. Once a request is received the marketplace instantiates a new transaction and awaits for either a seller or buyer to offer or request a similar product...

Foster et al. [9] then present “a context diagram of the marketplace composition” as shown in Figure 9. Examining the diagram, we observe the conceptual blurriness reflected by the arrows. The labels in the rectangles can be thought to indicate different types of flows (arrows). As shown in Figure 10, there are five types of flow:

- Flow of offered prices
- Flow of offered products
- Flow of agreed prices
- Flow of requested prices
- Flow of requested products

![Figure 9. Marketplace Context Diagram (modified from Foster et al. [9]).](image)

Figure 9. Marketplace Context Diagram (modified from Foster et al. [9]).

![Figure 10. Flow-based Marketplace Context Diagram.](image)

Figure 10. Flow-based Marketplace Context Diagram.

The flows are disconnected where it is assumed that Marketplace connects them. There is no connection between product flows and price flows. Requested prices are not connected to offered prices. “Structural semantics” do not correspond to intended meanings. For example, “offer” indicates making/creation of prices; nevertheless, it is not clear who is the creator of these prices because of the corresponding bidirectional arrow. Does the Marketplace offer prices? Does it request prices?

Conceptually, the figure is a fragmented representation that does not provide a suitable starting point for design of a system. To illustrate this point, Figure 11 shows an analogy to a communication system where fragments of flow are unnecessarily introduced.

![Figure 11. Fragmented Conceptualization of Communication System.](image)

Figure 11. Fragmented Conceptualization of Communication System.

It may be argued that the figure is meant as a rough sketch of the interactions between roles; nevertheless, it is possible to “sketch” a more appropriate initial conceptualization. Do engineers produce such an initial casual “sketch” when designing, say, a transportation system?
This type of picturing of systems of different flowthings is common in early stages of information systems analysis and is also used in further stages of development of these systems. It lacks a fundamental notion that underlies understanding of activities in information systems. We propose that flow of flowthings is a concept that provides a basis for this understanding.

Flow in this context is used to identify connections and roles of different subsystems. It plays a similar role in engineering systems, where it provides the encompassing notion to build a system of pipes, ducts, utility tunnels, poles, wires, and any other appurtenances needed to design and maintain a composite of gas mains, electrical and telecommunication manholes and conduits, steam and water mains, utility poles, fuel storage tanks, and aerial and telecommunications cables. The flow between components is used to calculate results such as energy, torque, power, etc.

Alternatively, flow-based conceptualization provides a systematic blueprint of the composition of services, as shown in Figure 12. There are two types of flowthings: products and prices, each with its own explicit flow stream. These streams can be described as follows.

Flows originating from Seller: Products (descriptions) are created (circle 1) and
(a) Flow to Marketplace (circle 3)
(b) Simultaneously triggering (circle 5) the creation of prices (circle 6) and flow of prices to Marketplace (circle 7). We assume that products and prices are connected through some indexing method, e.g., (product 1, price 1), (product 2, price 2), …, (product n, price n).

Flow through Marketplace to Buyer: Products (circle 3) and their prices (circle 8) flow through Marketplace to Buyers.

Price flows in Buyer and counter price proposal: Prices received by Buyers are processed, and Buyer may:
(a) Agree, e.g., upon receiving (product i, price i), the Buyer agrees on the deal by returning (circle 10, then 12) an agreement for (product i, price i).
(b) Create (circle 9) a counterproposal and send back (circle 9, 11, then 12). This counter price proposal flows to the Marketplace (circle 12), then to the Seller (circle 13). Upon receiving (product i, price i), the Seller agrees to the deal by returning (circle 14) agreement of (product i, price i); or Seller makes (creates) a further counter price proposal (circle 15).

**Price flows in Seller and counter price proposal:** It is possible to allow the exhibit of product without price; hence, Buyers can bid for the product by triggering the creation of price (circle 4).

Notice that “offer,” “agree,” and “request” are types of processes.

**Buyers’ requests:** Buyers may create a (description of) product (circle 15) that is transferred to the Marketplace (circle 16), where it is processed (e.g., included in a search), and—if found—triggers (circle 17) a price sent to the requesting Buyer. It also may flow to Sellers (circle 18) that process it and trigger the creation of a price (circle 19) for the requested product.

This blueprint of marketplace is a conceptual picture of different flow streams that enable the system designer to analyze any flow. An animation can be developed to show the flow between components of the system.

After introducing the Marketplace Context Diagram shown in Figure 9, Foster et al. [9] develop “a high level specification diagram” for the marketplace service shown in Figure 13. We will not describe in detail the semantics of this graph as we did previously with their Marketplace Context Diagram. While Foster et al. [9] introduce a very valuable contribution, our interest is not in this contribution; rather, we want to expose the weaknesses in the conceptualization style common in the software development process.

![Figure 13. Specification of Marketplace Composition (in part from Foster et al. [9])](image-url)

Figure 13 seems to start from scratch in the software development process, explicitly disconnected from the Marketplace Context Diagram of Figure 9. The semantics in the specification diagram of Figure 13 seem to express relationships between processes such as Negotiation, Seller Offer, etc. Some arrows resemble “control flow” as in Negotiation after Seller Offer. Some arrows may indicate sub-processes (communication), as in the connection between Negotiation and Seller Agree. The box “Initial scenario” seems to be a system module, unsuitable for the conceptual level.

In comparison, the flowthing model presents a “one-shot” conceptualization that can be further anatomized in an unbroken process to reach the design level.

### 8. Conclusion

In this paper, we have proposed using flow as a fundamental notion underlying understanding of activities in Web Services. We introduce a flow-based conceptualization of services through case studies with a high-level business description. It can be concluded that flow-based conceptualization promises to provide
a firmer base for Web applications development at the initial stages of design. It is possible to integrate this flow-based approach with current methods of software development processes.

References


Author

Sabah Al-Fedaghi holds an MS and a PhD in computer science from Northwestern University, Evanston, Illinois, and a BS in computer science from Arizona State University, Tempe. He has published papers in journals and contributed to conferences on topics in database systems, natural language processing, information systems, information privacy, information security, and information ethics. He is an associate professor in the Computer Engineering Department, Kuwait University. He previously worked as a programmer at the Kuwait Oil Company and headed the Electrical and Computer Engineering Department (1991–1994) and the Computer Engineering Department (2000–2007).