A Web Services-Oriented Approach to Unlock Information

Youcef Baghdadi
Sultan Qaboos University, Al-Khod, Oman

ybaghdadi@squ.edu.om

Abstract

This work proposes to use Web services to turn information into actions by leveraging and unlocking the informational assets of an organization. Indeed, Web services allow cost-effective composition and re-engineering of business processes because of their ability to connect applications, systems, and organization partners through the Internet-based standards (XML, SOAP, UDDI). The work consists of developing a process to generate interfaces to the knowledge in terms of information an organization possesses. These interfaces, implemented as Web services, are callable through the Internet. The proposed process is based on a new concept called factual dependency. Factual dependencies allow aggregations of attributes describing business objects and coordination artifacts that are affected by the same business events. Each resulting aggregation leads to a lowest level of granularity Web services. These Web services are then registered in a private or public UDDI to be discovered and (re)used at request to compose or re-engineer any internal or external business process. Unlike the approaches and tools that generate, in a spontaneous way or on a case-by-case basis, Web services from the complex and redundant elements of the information system, the proposed process generates Web services for the business objects and coordination artifacts as identified at the highest abstraction level of a business model. Indeed, the elements of the highest abstraction level that is the universe of discourse are unique and not redundant. The uniqueness and non-redundancy allows a generation, in a top-down-incremental approach with fewer analysts' intuition, of a comprehensive set of Web services reflecting the actual and the potential activities of the organization.

Keywords. Leverage and Unlocking Informational Assets, Factual Dependency, Web Services Generation, Integration, Business Process Composition and Re-engineering, Dynamic e-Business

Introduction

Web services, due to their ability in cost-effectively connecting applications, systems and partners, are leading to major change to business processes (e.g., improvement, re-engineering, composition, e-business and B2B). They are expected to effectively enable dynamic e-business (Maruyama, 2002). This requires IT organizations to evaluate their systems architectures and determine how they will deliver Web services (Chen, 2003). However, the deployment of Web services is still hindered by some technical and methodological issues. Technical issues are related to security, availability and performance.

While methodological issues concern with approaches, processes and methods to deploy Web services. This work deals with approaches of identifying, designing, implementing, and deploying Web services. Indeed, we still lack a coherent framework that allows guidance towards a method (including process, models and tools) to deploy such a revolutionary technology (Box, 2003). There are a number of potential approaches (e.g., top down, bottom up, incremental) that can be used to deploy Web services. However, major vendors use generally a bottom up approach (and to a less extend an incremental approach) that consists of mapping/wrapping existing applications, components or classes into Web services. The generation approach is generally performed through two steps. The first step consists of generating a callable interface of a component. The callable interface is defined in a Web services definition language (WSDL) file (similar to CORBA IDL file). For instance tools such as wsdlgen (http://longhorn.msdn.microsoft.com/lhsdk/indigo/conwsdlgentool.aspx) and GLUE (http://www.webmethods.com/docs/glue/guide/index.html) are used to generate Web services interface from existing Java classes. The second step consists of generating, for each WSDL file, the portion of the code that is used by the client applications to invoke the Web services. Tools such as Axis (http://ws.apache.org/axis), or WSIF (Web Service Invocation Framework: http://ws.apache.org/wsif/) are used to generate a stub (proxy) code from a WSDL file. That is, Web services are generally deployed:

- In ad-hoc, and spontaneous way, which makes them brittle (Altman, 2003).
- For a limited category of services (e.g., xMethods: http://www.xmethods.net).
- For tests, tutorials, etc.

Therefore, this kind of a case-by-case basis approach yields Web services that are not fully potential and useful. Moreover, the generated Web services are IT-perspective oriented. There is a lack of framework, or guidance to deploy a comprehensive and multipurpose set of fully useful and potential Web services. That is, Web services that are used for: interfacing legacy systems and components, integrating applications, B2B integration, and dynamic e-business.

This paper proposes a process to generate a comprehensive set of Web service from the knowledge an organization has on its elements at the highest-level abstraction level of a business model that is the universe of discourse. This knowledge is the information related to two types of elements: (i) the unchangeable and perpetual business objects, which are mainly products/services, raw material, parts, customers, accounts, and partners/suppliers; and (ii) the coordination artifacts representing the state of the business processes. Indeed, elements identified at the universe of discourse are sound and complete. That is: (a) any formal knowledge (information or processes) is derived from these elements; (b) these elements trigger the relevant business events or are affected by business events to which the organization is sensitive. For instances: the business event ‘shipment’ is triggered by the a state ‘accepted order’ of the coordination artifact ‘portfolio’ used to coordinate the ‘order entry’ business process. While the business event ‘customer order’ affects the balance of the business object ‘customer’. Each business event involves many operations. For instance, the business event ‘customer order’ involves operations such as ‘keying order’, ‘keying credit card number’, ‘transferring credit card’, ‘requesting payment’, and so on. The chain of operations forms a business process (e.g., payment processing, fulfillment).

The process is based on a concept we introduce and call factual dependency, which we define as: “two attributes X and Y describing business objects or coordination artifacts are factually dependent if they are concerned by the same business event”. In fact, the concept of factual dependency allows aggregation of attributes concerned by any create, retrieve, update or delete (CRUD) operation fired by the business event. These types of operations constitute a

framework to identify the lowest level of granularity Web services we can master with least ambiguity. Indeed, each distinct operation is readily specified in terms of input/output parameters, which eases the generation of Web services and their respective WSDL. That is, the Web services messages encoded in XML and conveyed by SOAP.

The process allows us to interface the relevant element of information as Web services. These Web services are then registered in a public or private UDDI to be dynamically discovered, invoked and used in the composition or re-engineering of internal as well as external business processes. Accordingly, we can dynamically build business processes at request by composing the existing Web services. This way, any unlocked information is turned into actions.

The next section presents the state of art and related work to interfacing and integration technology including approaches to deploy Web services. Section 3 introduces the concepts used in the generation process namely: factual dependency, business model, universe of discourse, information system, business object, business process, business event and coordination artifact. Section 4 develops the steps of the proposed process to generate and compose Web services. Finally, a conclusion section presents the results and further developments and issues.

Interfacing and Integration Approaches

The problem of interfacing and integration deals with technology that makes heterogeneous databases and applications "speaking" different languages interoperable (e.g., COBRA, DCOM, RMI), e-services and SOA model, to Web services and their composition in order to internally or externally integrate applications involved in business processes. Each technology is concerned with a specific aspect of the problem.

Traditional Integration Approaches

Heterogeneous databases (Batini, Lenzirini, & Navathe, 1986) deal with approaches to sharing data among databases designed with different data models, and implemented with different DBMS (Network, relational or object-oriented DBMS). It deals with heterogeneity and interoperability of structured data.

Object-oriented approach makes applications interoperable provided that we install a bus as mechanism of discovery and invocation. Despite the fact that they allow tightly coupling because they are built on their own technology (CORBA objects communicate using IIOP, DCOM heavily depends on Windows platform, RMI depends on Java Platforms), they yield great returns and quality productivity (Chen, 2003). However, this object-oriented model was designed mostly from an IT perspective in order to help IT operate more effectively (Wong, 2001). In the last years, the model has been extended with the introduction of the Service-Oriented-Architecture (SOA) model, which helps to separate business intent from IT implementation. This allows sharing business services across the enterprise and to support B2B initiatives. For instance, the e-services model is composed of a set of business services, a set of business components, a set of IT elements, and a set of business rules (Wong, 2001).

Web services come in the scope of SOA architecture. Unlike RPC, CORBA, DCOM and RMI, which require tight communication, Web services are built on loosely connected Web model.

Web services Integration Approaches

There exist a number of definitions of Web services. Business-oriented definitions focus on their role in timely connecting partners with reduced cost (Box, 2003; Maruyama, 2002). While, technology-based definitions focus on the standards on which they rely such as XML,

SOAP, WSDL, UDDI (Aoyama, Weerawarana, Maruyama, Szyperski, Sullivan, & Lea 2002; Ethan, 2002; Kreger, 2001), or BPEL4WS (Jablonski, 2003; Leyman, 2002). W3C/WS Architecture Group defines Web services as "software system identified by URI, whose public interface and bindings are defined and described using XML. Other software can discover its definition. These software may then interact in a manner described by its definition using XML-based messages conveyed by Internet protocols" (Austin, Barbir, Ferris, & Garg, 2002).

There are a number of potential approaches (e.g., top down, bottom up, incremental, in-out-incremental) that may be adopted to develop Web services. However, tools developed by major vendors and adopters of Web services are implicitly a bottom up, and to a less extend an incremental approach.

1. A bottom up and incremental approach consist of:

   - Generating Web services from existing classes, components or stored procedures or wrapping existing applications. For instance, having an existing Java class, tools such as GLUE and wsdlgen generate a Web services interface for this class. The interface is described using WSDL standard and stored in a WSDL file (similar to CORBA IDL file) accessible as Web resource.
   - Once the interface is described, tools such as Axis, Glue, and WSIF generate a stub (proxy) code that is a portion of code to be used by any client application to invoke the Web services.

The bottom up approach as actually used to deploy Web services looks like a kind of a case-by-case basis, ad-hoc, or a spontaneous way. The resulting Web services are depending on specific technology, or limited to a certain category of services such as those found in xMethods (http://www.xmethods.net). Moreover, this approach lets more intuition to the analysts instead of assisting them. Deploying Web services in this way yields Web services that are brittle (Altman, 2003), ambiguous and not fully useful.

2. A top down approach consists of:

   - Identifying the potential services, which are candidate to be Web services.
   - Defining their respective WSDL files.
   - Providing stub/skeleton code to access the services.
   - Composing applications and business processes using the specified Web services.

A top down approach yields a comprehensive set of Web services that are independent of any developing technology unless it is a standard.

3. In-out-incremental approach (Chen, 2003) consists of an internal implementation of Web services to be then expanded outward as standards and technologies mature, that is:

   - Deploy first Web services internally to improve operational efficiencies and gain a unified view of complex business processes. It is easy to ensure that standards are available.
   - After Web services are successfully utilized internally, organization can extend these services to customersThe approach described in this paper is a combination of a top down and an in-out incremental approach. It is in touch with the e-services and SOA model. However, it is mainly based on the concept of factual dependency that allows aggregation of attributes describing the business objects and coordination artifacts

as identified at the highest abstraction of a business model. This approach allows identification, a definition and a specification of Web services that are not intuitive to the analyst for less ambiguity. Moreover, the Web services are specified independently of any technology.

Factual Dependency

Our approach to turn data into business processes through Web services is based on the concept of factual dependency between the attributes that describe the elements of the universe of discourse. This section introduces firstly the concepts used to model a business, which are universe of discourse, information system, business object, coordination artifact, business event and business process; then the concept of factual dependency; and finally, the process to generate Web services.

Business Model

A business is an open system that seeks some goals or responds to events. There exist several models of business (Zackman, 1996). We consider a business as composed of the following components of:

1. Business Events, Input, and Output.
2. Production System.
3. Logistic System.
4. Partners.
5. Business Management/Control System.
6. Information System.

Elements of the Universe of Discourse

The business events, the input/output, the production system, the logistic system and the partners are parts of the universe of discourse (or the reality). The universe of discourse contains four types of elements:

1. *Business objects* are perpetual tangible as well as intangible elements. These elements are the managerial resources on which an organization focuses. They are real and unique. For example, raw material products/services, parts, are the main tangible elements, while the accounts (e.g., customer account, bank account) are intangible elements.
2. *Business Processes* transform business event/input into output. The processes are the decompositions of the value chain (Haag, 1999). Each process has a life cycle i.e. a sequence of activities fired by the business events.
3. *Coordination Artifacts.* They are managerial or organizational artifacts that represent the relevant states of the business processes. Indeed, decomposition of the value chain requires artifacts to coordinate and to interface the business processes.
4. *Business events* are identified in space, and time. There are two types of business events. External and internal business events. External business events effect business objects and coordination artifacts by changing their states (e.g., the business event 'customer order' adds new order and changes the balance of the customer), while internal business event are triggered by the state of business objects or coordination artifacts (e.g., the business event 'shipment').

Business objects, coordination artifacts, business processes and business events are differently represented in the information system. The representation is not unique and consistent.

Elements of the Information System

The information system is a technology-based representation of the elements of the universe of discourse that are business objects, business processes and coordination artifacts. This representation consists mainly of data, applications and integration middleware. Hence, it should contain only one representation of each element identified in the universe of discourse. However, the actual information system contains different representations of the same element. This is mainly due to three factors: (i) our different intuitions and perceptions of the reality, (ii) the different languages we use to communicate, and (iii) the variety of abundant implementing technologies. For instance, the business object 'customer' may be perceived as an account (for the accounting) and a partner (for the management). A track of this perception is kept in different database tables, flat files or XML documents. Similarly, a business process may be implemented differently (e.g. application, component, class, procedure or manual). This breaking in the representation requires integration mechanisms (e.g., middleware, ERP, EAI, B2B) for multiple reasons namely: (a) the continual need of reconstructing the entire representation, b) some processes involve more than one tier, and (c) the need to interact in real-time or (near real-time) with partners.

Therefore, we have two abstraction levels:

1. The universe of discourses where the elements are real and unique. That is, we have only the original of the elements there is no clone. If we damage or lose an element, we cannot reconstitute it.
2. The information system where the elements have various representations/images. That is, we may have various technology-based copies of the same element. If we damage or lose the copy, we can reconstitute it

It is obvious that the information system is more complex and ambiguous than the universe of discourse as the same element of the latter has various heterogeneous images in the information system.

Therefore, integrating elements in the information system is more complex. It is mostly influenced by IT rather than by some business perspectives. This is especially more evident in the case of e-business, where new business processes are more and more innovated, re-engineered, generated or completely built from scratch for some specific business events.

An approach to dynamically compose business processes at request by composing them from comprehensive ready-to-use services is required. We propose a process based on a new concept that is factual dependency, which we detail in the next section.

Factual Dependency Definition

A factual dependency is a dynamic-oriented constraint between two attributes X and Y describing an element of the universe of discourse (business object or coordination artifact). The constraint stipulates that two values *x* of X and *y* of Y are inserted/update/deleted/retrieved at the same time when a business event occurs.

The concept of factually dependency allows an aggregation of attributes describing tangible as well as intangible elements of the universe of discourse with respect to the business events they undergo or they trigger. The attributes having their values created, deleted, updated or

retrieved by the same business event are grouped together to be further interfaced by the same interface.

Formally, an attribute **Y** is factually dependent on an attribute **X** if the attributes **X** and **Y** are concerned with the same create, update, retrieve or delete operation.

A factual dependency between attributes X and Y is denoted X ➔ Y to keep the similarity with the concept functional dependency used in the relational database model (Codd, 1990).

Table 1 shows the following factual dependencies assuming that the business object 'customer', element of the universe of discourse, is described by the attributes: id, name, address, balance, and mode of payment.

- FD 1: {name, address, balance, mode of payment} is used for new customer.
- FD 2: {id, balance} is used to update the balance when the business event 'customer order' occurs (the balance is increased with the total amount of the order). It is also used when the business event 'customer pays' (the balance is decreased with the paid amount).
- FD 3 {id, balance} is used to inquiry the balance in order to trigger a business event.
- FD 4: {id, address} is used to update the customer address when the event 'customer changes address' occurs.

Table 1: Some factual dependencies that aggregate the attributes describing a customer

Factual Dependency	***Attribute*** ***CRUD Operation***	Id	Name	Address	Balance	mop
FD 1	Create	X	X	X	X	X
FD 2	Update	X			X	
FD 3	Retrieve	X			X	
FD 4	Update	X		X		

In a nutshell, each combination of attributes may be regarded as an operation. This is particularly true for the retrieve operation where each project operation (in the sense of the relational algebra) on a set of attributes is a retrieve operation. Moreover, a factual dependency as aggregation generates several interfaces (e.g., FD 2). Therefore, criteria to select relevant and formalized combinations are required. We propose to use first some rules and the relevant the business events as defined in the universe of discourse (e.g., customer order, payment).

Rules for Factual Dependencies

The concept of factual dependency is different from the well-known concept of functional dependency used in the relational schema design (Codd, 1990). Therefore, not all the inference rules used for the functional dependency are applicable for the factual dependency. However, factual dependencies respect certain rules, which are:

Rule1: Reflexive

X ➔ Y if Y is strictly included in X. Each attribute is aggregated with itself or a part of itself. This is trivial factual dependency.

Rule 2: Commutative

X ➔ Y then Y ➔ X.

If X ➔ Y, that is the attributes X and Y are concerned by a CRUD operation ‘o1’ then Y ➔ X, that is Y and X are concerned by the same operation ‘o1’.

Rule 3: Transitive

X ➔ Y and Y ➔ Z then X ➔ Z

When the sets {X, Y} and {Y, Z} are concerned by the same operation ‘o1’, then set {X, Z} is also concerned by that operation ‘o1’.

Rule 4: Augmentation

X ➔ Y and X ➔ Z then X ➔ {Y, Z}

If the of attributes {X, Y} and the set {X, Z} are affected by the same operation ‘o1’ then the set {X, Y, Z} is affected by the operation ‘o1’. For instance {Id, name} and {Id, address} are affected by the operation ‘add new customer’ then {Id, name, address} is affected by this same operation.

Rule 5: Multiplicity of interpretation of factual dependency

Two attributes X and Y may be involved in a set with different meanings. That is X ➔ Y may have different interpretations. For instance {Id, balance} may generate more than one interface. Indeed, we may generate an interface {Id, balance} to query the balance, and a second interface {Id, balance} to update the balance. That is, we may have the same aggregation of attributes, which may be concerned by many distinct CRUD operations. Each operation corresponds to a business event. Theoretically, each combination of attributes is a factual dependency that leads to an operation. However, not all the factual dependencies are relevant. We will consider only those corresponding to the actual business events as described in the universe of discourse.

Relationship of Factual Dependency to Business Event

A CRUD operation is applied only when a business event occurs in the universe of discourse. We update (create, modify or delete) the states of the elements of the universe of discourse when business events occur in this universe. Similarly, we attempt to retrieve information related to business object or state of the coordination artifacts in order to trigger events. However, there is a multitude of business events that occur in the universe of discourse. We consider only the events that affect the business objects or the state of the coordination artifacts. That is, there exist a relationship of business event to factual dependency (Figure 1). Business events allow us to determine the aggregation of attributes they affect. Indeed, a business event is captured as a value of attributes that it affects. For instance, a customer order is captured as a value <oid = 100, cid = 125, date = 12/12/12, item = 1250, quantity = 5>. Similarly, a factual dependency, as aggregation of attributes, allows us to specify the potential business events. Table 2 shows that the business event order is related to FD 1 {id, name, address, balance payment} and FD 2 {id, balance}, while the factual dependency FD 2 {id, balance} is concerned with three business events (customer orders, customer pays and call customer for payment}.

To validate the set of generated factual dependencies that will lead to Web services, we confront each factual dependency to an actual business event as shown in Table 2. For that purpose, we proceed in a reverse manner. That is, we first generate a list of business events from the set of factual in an automatic fashion. Indeed, factual dependencies are easier to specify as aggregations attributes describing perpetual elements of the universe of discourse. While the events are characterized by space and time making their perception and capture harder. Then

we confront the generated list of business events to the actual business events as identified in the universe of discourse.

Table 2: Factual dependencies and corresponding business events

FD	***Attribute*** / ***Operation***	Id	Name	Address	Balance	Payment	**Events**
FD 1	Create	X	X	X	X	X	Order
FD 2	Update	X			X		Order
FD 2	Update	X			X		Payment
FD 2	Retrieve	X			X		Call for Payment
FD 4	Update	X		X			Change Address

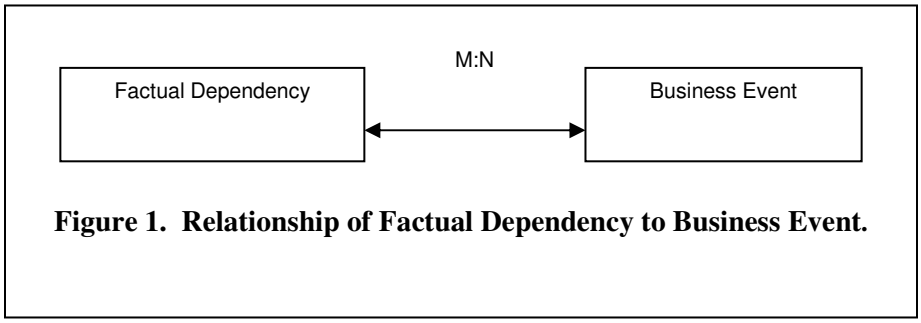

Figure 1. Relationship of Factual Dependency to Business Event.

Web Services Generation

Web services provide a standard way for any user, through an application, to access the elements of the universe of the discourse. Indeed, once we keep track of a representation of the element of the universe of discourse in a legacy database, a flat file or XML databases, we can interface it provided we get the business events that affect it or are triggered by it. In fact, the business event fires a set of operations. Each operation described in term of input /output parameters is identified as interfaces that can be described using XML, and implemented as Web services with respect to a specific technology (e.g., stored procedure in the case of the relational database, class, application). The Web service in turn is accessed by any application (e.g., Java application running on a Web server), which presents the information to the end user. Therefore, the specification of the Web service can be generated from the set of validated factual dependencies and business events.

To keep a very low level of granularity, each factual dependency leads to one-many CRUD operations depending on the number of business events related to this factual dependency. The operation will require input parameters/output parameters. In general an update operation (create, update and delete) requires only input parameters. Whereas, a retrieve operation requires input parameter (generally the identifier of the element of the universe of discourse) and output parameters.

Table 3 shows an example of the Web services that may be generated by the factual dependencies FD 1, FD 2 and FD 4.

Table 3: Factual dependencies and corresponding Web services

FD	*Operation*	*Web Service*	*Input Parameters*	*Output Parameters*
FD 1	Create	New Customer	name, address, balance = 0, mode of payment	
FD 2	Update	Customer Order	name, order amount	
FD 2	Update	Customer Payment	name, amount paid	
FD 2	Inquiry	Balance Inquiry	Id or name	balance
FD 4	Update	Mode of payment	Id, mode of payment	

Process of Turning Data into Business Processes

The proposed process is based on the concept of factual dependency. It consists of:

Part 1: Turning Data into Web services

This part consists of the following five steps:

1. Selection of the business objects and coordination artifacts. That is, pick up the business objects/coordination artifacts as described in the universe of discourse. Indeed, working on the elements of the universe of discourses, rather than the elements of the information system (more business perspective-oriented than IT perspective-oriented) minimizes the analyst's intuition and leads to less ambiguity in specifying the Web services.
2. Description of the business objects and coordination artifacts. This consists of defining, in terms of attributes, what we expect to know (in terms of information) about these business objects or coordination artifacts.

Steps 1 and 2 lead to a data dictionary where tangible as well as intangible elements of the universe of discourse are statically described and defined.

3. Exhaustive list of factual dependencies. That is:
   - Generation of all the factual dependencies by combining the attributes.
   - Deduction of the relevant factual dependencies by confronting them to the business events as they may occur in the universe of discourse.
   - Specification of each factual dependency as a CRUD operation with a focus on the input/output parameters.
   - Implementation, in the information system, of the operations related to the factual dependencies. This implementation may be a program, a stored procedure, a component, a class, Java Bean, etc.
4. Generation of Web services corresponding to these operations. This step is easy since the operations are specified in term of input/output parameters. We can use a CASE tool or any other tool at this stage to automatically generate the Web services.
5. Registration of the generated Web services. The resulting Web services are registered in a public or private registry and discovery artifact to be easily discovered.

Pat 2: From Web services to Business Processes

This part consists of the following five steps:

1. Identification of the business event. From the universe of discourse, identify the relevant business events such as 'customer order'.
2. Description of the flow of the operations corresponding to the business event (or business rules). Each business event fires a flow of operations. This flow begins with the capture of the business event and ends by a production of an output. We can use tools to model the flow (e.g. BPEL4WS).
3. Identification of the operations in terms of CRUD operations. The automated operation corresponds generally to a CRUD operation.
4. Matchmaking between the operations and the registered Web services.
5. Replace in the flow (BPEL4WS) the CRUD operations by their respective Web services.

The part 2 of the process is fired by any new business event, in a B2B perspective, or when re-engineering the existing business processes.

Conclusion

Different approaches and tools used for integration though Web services, excellent and de facto standards that facilitate the integration, have been studied in this work. The study concludes that the current approaches are more IT-oriented and therefore proposed from an IT perspective not from the business perspective. That is, Web services are deployed from the existing complex and IT-depending elements of the information systems, and in a kind of a case-by-case basis, which makes the integration a very hard task. We proposed a top down incremental approach to generate Web services, from the highest abstraction level (universe of discourse) where the elements namely the business objects and the coordination artifacts are easy to capture with less analyst intuition. The Web services are generated from validated factual dependencies that are aggregations of attributes describing the elements of the universe of discourse. The validation is performed trough a confrontation to actual business events.

The proposed approach allows a comprehensive set of useful and multipurpose Web services.

This is a significant issue nowadays where organizations are looking to sharing Web services across the enterprise for dynamic e-business, and to support business-to-business integration, which is more and more intensive and critical for a business survival.

We will develop after a global architecture and a supporting tool that allows organization to really turn information into action by interfacing the informational asset through Web services.

References

Aoyama, M., Weerawarana, S., Maruyama, H., Szyperski, C. A., Sullivan, K. J. & Lea, D. (2002). Web services engineering: Promises and challenges. *Proceedings of International Conference on Software Engineering*, 19-25.

Altman, R. (2003, July). The challenge of Web services. *Business Integration Journal*, 59-59.

Austin, D., Barbir, A., Ferris, C., & Garg, S. (2002). Web Services Architecture Requirements, W3C Working Group Draft 14. Retrieved from http://www.w3.org/TR/2002/WD-wsa-reqs-20021114

Batini, C., Lenzirini, M., & Navathe, S.B., (1986). A comprehensive analysis of methodologies for database schemas integration. *ACM Computing Surveys*, *18* (4), 322-364.

Box, B., (2003). The Web services revolution. Retrieved from http://www.eds.com/thought/thought_leadership_web_revolution.pdf

Codd, E., (1990). *Relational model for data management Version 2*. Addison-Wesly.

Chen, M., Chen, A. K. N, & Shao, B. B. M. (2003). Implications and impacts of Web services to electronic commerce research and practices. *Journal of e-Commerce*, *4* (4), 128-139.

Ethan, C., (2002). Web services essentials: Distributed applications with XML-RPC, SOAP, UDDI & WSDL. O'Reilly & Associates.

Haag, S., Cummings, M., & Dawkins, J., (1999). *Management information system for information age*. Irwin Mac Graw Hill.

http://longhorn.msdn.microsoft.com/lhsdk/indigo/conwsdlgentool.aspx

http://www.alphaworks.ibm.com/tech/wsif

http://ws.apache.org/wsif/

http://www.xmethods.net

http://ws.apache.org/axis

http://www.webmethods.com/docs/glue/guide/index.html

Jablonski, S., & Petrov, I., (2003). Web services, workflow and metadata management as the means in the electronic collaboration era. *Proceedings of ICEIS 2003*, Angers/France.

Kreger, H. (2001). Web services: Conceptual architecture (WSCA 1.0). IBM Software Group.

Leyman, F., & Roller D. (2002). A quick overview of BPEL4WS. IBM Developer Work.

Maruyama, H. (2002). New trends in e-Business: From B2B to Web services. *New Generation Computing, 20*, 125-139.

Zackman, J.A., (1996). Concepts of the framework for enterprise architecture. Los Angeles, CA: Zackam International.

Wong L. (2001). e-Services: A key component for success. *eAI Journal*, (March), 18-25

Biography

Dr. Youcef Baghdadi has a long and extensive experience in teaching undergraduate and graduates. His experience includes databases, information systems, cooperative information systems, Internet computing and e-commerce. His current research is oriented towards methods that bridge the gap between business and IT. It includes methods and processes to develop interacting information systems, Web applications, B2B e-commerce, and Web services.