Management of Traffic Accidents Using Web Services

Fadi F. Fouz, Mohammad O. Afandi
King Abdulaziz University, Saudi Arabia

Abstract
In this paper, a traffic accident management system based on web services is proposed and developed. Web services are used to integrate a number of subsystems and databases. The subsystems include incidents management system, patrol vehicles tracking system, Google maps API, police data center and accidents reconstruction system. The proposed model of integration is a composition of object, aspect and service orientation paradigms. Although the subsystems have heterogeneity in database, data structures, functions, and resources, the integration based on the proposed model succeeded to pass most of the heterogeneity problems.

Keywords
Web services, Data integration, Systems Integration, XML,

1. Introduction:
Data integration has been a very hot topic in research and industry for many years. It is a pervasive challenge which faces application or system that needs to query across multiple autonomous and heterogeneous data sources. In previous approaches of data integration, the direct link to the database was a poor solution since the database owners prefer to maintain the database architecture and structure hidden. Also in system integration, the owners of service or functions believe that they put a lot of efforts to build their functions, so they do not want to provide them as open source. Furthermore, some functions have confidential methods and information which are required to be secret. In addition, the interfaces and usage is different from language to others. Hence, we could recognize the difficulties of data and system integration.

In general, data integration aims at combining selected data sources to give users the illusion of interacting with one single data source. Since data sources might be heterogeneous, data integration becomes more difficult. The heterogeneity could be in data source schema, location, naming difference, value difference, and type difference. For instance, data source could be XML data source, Relational Database Management System (RDBMS), flat data, or a file system. In XML and RDBMS, the definition, vocabulary, interface and connectivity are completely different. One of the major bottlenecks in setting up a data integration is the effort required to create the source description, and more specifically, writing the semantic mapping between the sources and the mediated schema. Writing such mapping and maintaining them requires database expertise and business knowledge. Hence, a significant number of researchers focused on semi-automatically generating schema mappings. In general, automatic schema mapping is an AI-Complete problem, aims at creating tools that automate, speed up the creation of the mapping, and reduce the amount of human effort involved.

Data integration could be accomplished through three steps. First, searching and finding data sources distributed over network, which are related to specific domain, and finding the description of these sources. Second, generating unified schema mapping for discovered data sources. Third, providing a way to deal and interoperate with schema mapping to pass the data in efficient way (Fig-1).
Fig-1: Integration basic steps

1.1 Data representation, and semantic integration

Traditionally, scientific applications deal with inputs and outputs (I/O) mostly via custom flat file formats or databases. Furthermore, different communities often use proprietary formats to describe similar data. So in order to couple such applications together, application-specific scripts are needed to perform conversions between data formats. Such scripts are error-prone, hard to maintain and to reuse. The Web services approach is very promising for such application due to the use of standard protocols and data formats. Web services are networked endpoints that exchange data in XML format, and are accessed via standard internet protocols (HTTP, SMTP). In the past few years, XML has become de facto format for representing data in Grid and businesses communities. XML Schemas are typically used to define the structure of an XML document. There are several robust, freely available, software tools that can parse and validate XML documents in almost every language in use today. Web services provide a clean separation between interfaces and implementation. Applications can define their publicly accessible interfaces by using the Web Services Definition Language (WSDL). WSDL operations handle strongly typed I/O parameters that are defined using XML Schemas. Therefore, Web services are suitable for loosely coupled workflow composition, provided that XML schemas for I/O parameters are properly understood by the applications involved. The data types involved in the I/O parameters for different applications may be similar structurally or semantically. Data integration between applications is easier to perform if the data types are structurally similar, i.e. if they use a standard XML schema that is commonly agreed upon within a particular discipline or community. Integration between data that is only semantically similar, but structurally different, is more difficult to accomplish. Data represented as XML can be easily deposited in XML databases, because utilities exist to perform conversions for mapping XML data into relational databases. This makes querying XML data reasonably straightforward. On the other hand, querying traditional flat files is extremely difficult because of the lack of strong data-typing and structure.

2.2 Web Services

Web Services are essential to the evolution of the Web. They allow placing active objects on web sites and providing distributed services to potential clients. One of their main current uses is for the management of distributed information. Distributed database systems always suffered from platform and software incompatibilities. Web services are not inventing anything new, but they are bringing an important breakthrough to distributed data management simply by proposing web solutions that can be easily deployed independently of the nature of the machine, the operating system and the application languages. The first step in using web services is to find the service/information. This could be achieved for instance with a standard, Universal Discovery Description and Integration (UDDI), which is a specification for distributed Web-based information registries of Web Services, and also a publicly accessible set of implementations. The second step is to understand how to obtain it. This could be achieved in particular by using the Web Service Definition Language(WSDL), a language for describing web service interfaces, something like Corba"s IDL for the web. WSDL is an XML
format for describing network services based on operations and messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define a functionality of a service. The third step is to get and use it, to obtain directly the information, or use it in our (mediator or warehouse) system using the Simple Object Access Protocol (SOAP), which is an XML based lightweight protocol for exchange of information in a distributed environment. SOAP allows specifying the (XML) types of arguments and service results, and could be used in combination with HTTP.

2. Integration problems
In general, information systems are not designed for integration. Thus, whenever integrated access to different sources is desired, the sources that do not fit together have to be coalesced by additional adaptation and reconciliation functionality. While the goal is always to provide a homogeneous, unified view on data from different sources, the particular integration task may depend on:
- The architecture view of an information system.
- The content and functionality of the component system.
- The kind of information that is managed by component systems (alphanumeric data, multimedia data, structured, semi-structured, unstructured data).
- Requirements concerning autonomy of component system.
- Intended use of the integrated information system (read-only or write access).
- Performance requirements.
- The available resources (time, money, human resources, know-how, etc.)
In addition, several kinds of heterogeneity have to be considered. These include differences in hardware, operating system, data management, data models, schema, data semantic and naming differences.

3. Data Heterogeneity
The integration between distributed data sources would be relatively easy if the data is homogeneous. The heterogeneity in data raise the difficulty in integration process as the integrator or broker should achieve unified forms or format to be understandable from the system. The heterogeneity can appear in many ways such as: naming, relational structure, value semantic, data model, timing, syntax, transaction and security. Different instances of heterogeneity can be classified into one or a combination of the following:
- **Naming heterogeneity:** This occurs when the same values are stored in different databases but the names given to the attributes are different in different systems. These can be handled by a simple (syntactic) attribute transformation of the query.
- **Relational structure heterogeneity:** When the composition of elementary attributes into composite structures varies but once again values stored are identical. This can be handled by a (syntactic) relational transformation of the query.
- **Value heterogeneity:** In this case the way in which values are represented is different in different databases. This may involve type and value transformations.
- **Semantic heterogeneity:** This is the most difficult form to deal with as in this case the data stored in different databases embody different assumptions, e.g. in what they represent or in how they have been captured.
- **Data model heterogeneity:** Here the data model itself is the issue and transformations between data models and differences between them are relevant.
- **Timing heterogeneity:** This concerns the changes over time in the structure of a database, the representation of attributes and the values themselves. Basically, almost any difference from each of the preceding categories, which can occur between databases,
may also arise within a single database if it changes with time.

4. The traffic accidents management system (TAMS)
This section states and describes the traffic accidents management system, its data sources and subsystems that will be integrated together.

4.1. Incidents Management System
In this system, the initial information about the incident (accident) will be gathered, so traffic police can access the location easily and start accident reconstruction process Fig.2. The system stores information related to incident such as driver involved information, vehicles information, location information, date and time, call time, policeman (or investigators) arrival time and damages information. Incident data could be helpful later on to determine the hot point area for accidents and measuring KPI (Key Performance Indicator) for traffic police. The easy steps to create a new incident in the system are: first, the operator receives a call from drivers and get some information and enter it into the system. Second, the system automatically stores time of incident creation. Third, the operator determines the nearest patrol, call them and provide them with the accident location information. Finally, the investigator or policeman proceeds to the location to begin accident reconstruction process.

4.2. Patrol Vehicles Tracking System
This system is based on Automatic Vehicle Locating (AVL) technique. It determines the geographic location of patrol vehicles and its transportsations history. The core aim of this system is to ease the process of specifying the nearest patrol to the accident location. Patrol locations appear colored on a map depending on their current state. These states are divided into: free patrol, patrol assigned with accident, patrol arrive to accident, patrol reconstructing an accident, and patrol just has done reconstruction process. Vehicles information are stored into a separate database which is integrated with Google maps API. This system will be integrated later with Incident Management System to automate finding nearest patrol process.

4.3. Google maps API
Google maps API could be considered as data source. It provides a wide array of APIs that let developers embed the robust functionality and usefulness of Google Maps into their own website and applications and overlay their own data on top of them. Developers have the option either simple use or extensive customization. There are now several API offerings: Maps JavaScript API, Maps API for Flash, Google Earth API, Static Maps API, Web Services and Maps Data API.

4.4. Traffic Police Data Center
Time factor is very important in accident reconstruction process. So, to decrease time needed to complete reconstruction process, accident reconstruction system retrieves data about drivers, vehicles and insurance involved in accident from Traffic Police Data Center. Traffic Police has full information about all drivers, vehicles and insurance, so by passing driver ID (for example), full information about driver will be retrieved. This also applied on vehicles (by passing plate no.) and insurance (by passing insurance-no.)Fig-3.
4.5. Accident Reconstruction System
This system includes all scientific processes of investigating, analyzing, and drawing conclusions about the causes and events during a vehicle collision. Investigators are employed to conduct in-depth collision analysis and reconstruction to identify the collision causes and the contributing factors in different types of collisions, including the role of the drivers, vehicles, roadway and the environment. After reconstruction process, all data is transferred to a central server from accident site. Drivers get a reference no. to allow them to report the accident to insurance company. Once accident reconstruction process has done, drivers and insurance company can access accident information online, print report or print repair authorization.

5.4. The proposed integration model
The proposed model of integration is a composition of object, aspect and service orientation paradigms. Object orientation is a manner of programming based on the concept that used systems should be built from a collection of reusable components called objects. These objects may contain properties (data field) and methods (functions) together with their interactions. Fig.-4. Following this technique in programming improves usability and maintainability, catch errors at compile time rather than at runtime, reduces large problems to smaller. Aspect orientation is a programming paradigm that isolates secondary or supporting function from the main program's business logic. Service-orientation is a programming paradigm that deals with usable services distributed over network. Web services make functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network-enabled.

To apply this integration model, subsystems have to be redesigned by dividing them to components and aspects. The components have to match subsystem entity, and the entity field becomes a property in these components, and any function related to this entity becomes a method in the same component. The non-main functions (sometimes called support function) have to be isolated in aspects. For example, to validate user identity, component will communicate with authorization aspect, and pass username and password. The aspect returns the results as a component containing the validity of this user and his permissions.

Finally, any function in the subsystem which needs to be integrated with external data source or external function must be passed through web service. These web services could be considered as interface for communicating and dealing with other subsystems or data sources. The results of these web services are encapsulated as component (object) interpretable by other subsystem either by WSDL or generated proxy. WSDL (Web Service Description Language) is an XML-based language that provides a mode for describing web service. Generated proxy is a predefined XML file that is generated from WSDL via specific tools to catch programming errors while using remote web service during development-time.
Redesigning subsystems costs efforts, money and time. A better approach is to implement the integration model as a middleware. The middleware will have all the integration functions needed as web services, and will pose web services results as components. In this way each subsystem can manage the results as needed. Fig-5.

6. Implementation:
To implement the proposed integration model the following methodologies of integration were followed:

6.1 Using web service to link TAMS with Traffic Police data:
In this method, the system uses web service to query remote databases located in different machines. Fig-6. This method follows simple steps: first, the application uses the proxy interface to call particular function by passing some parameter. Second, the proxy serializes the request to XML and transfers it through network. Finally, the returned result is received by web service caller via web service proxy. The application stored retains data in its database and makes it available for future use.

6.2 Using web service to integrate TAMS off-line with central database:
In this method, the traffic accident management system uses web services to synchronize offline data with central database Fig-7. While the system is running offline, user could add data and the system generates unique ID’s via special scientific methods. When the connection becomes established and stable, the synchronization between central database and local database will be achieved using web service. Thus, the data on both databases will be semi-real and the application will work smoothly.
6.3 Using web service to integrate client tool
To allow application to communicate with database, Microsoft Silverlight is used to support such kind of connectivity. Fig-8. Microsoft Silverlight is a programmable web browser plugin that enables features such as animation, vector graphics and audio-video playback that characterize rich Internet applications. It is compatible with multiple web browser products used on Microsoft Windows and Mac OS X operating systems. Mobile devices, starting with Windows Mobile 6 and Symbian (Series 60) phones, will also be supported. Silverlight has very nice interface but unfortunately, it could not interact with any database directly. So a web service is used to deal with database.

6.4 Experiments results
The five subsystems comprising TAMS have been run for 4 months before integration and six months after integration. The number of incidents and accidents recorded were around 22,550. In order to observe the integration impact on both system, performance and process performance, key performance indicators for incident, accident, users and system transaction were recorded and stored into the system. i.e. the time needed for performing accident reconstruction process, time needed to reach accident location, time needed to send claim to insurance company, time needed to send complete data to traffic accident police etc. Table-1 shows the average time required to accomplish some processes before and after integration.

<table>
<thead>
<tr>
<th>Task</th>
<th>Before integration</th>
<th>After integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiving incident, get location</td>
<td>10 minutes</td>
<td>3 minutes</td>
</tr>
<tr>
<td>Assigning incident to expert</td>
<td>Immediate</td>
<td>Immediate</td>
</tr>
<tr>
<td>Sending a patrol to accident site</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>Arriving accident site</td>
<td>32 minutes</td>
<td>11 minutes</td>
</tr>
<tr>
<td>Accident reconstruction</td>
<td>20 minutes</td>
<td>Immediate</td>
</tr>
<tr>
<td>Sending data to traffic police</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>Sending data to insurance company</td>
<td>Manual</td>
<td>Manual</td>
</tr>
</tbody>
</table>

Table-1 clearly shows how integration improved business performance. Furthermore, some tasks which usually were done manually before integration have been automated after integration.

7. Scope and limitation of the work
The employment of a federated schema structure was constructed on constituent information access services that exist in various sites. This was possible because the quantities of information access services are very small, and not changing frequently. But if this approach is to be scaled up with rapidly changing component, data access services might not be viable.

The trial product made use of RPC procedures for the interaction of the clients and the servers. The implementation of this was very simple, and offered excellent performance for easy calls. However, it imposed very inflexible typing of data.

The proposed Web service description has been used exclusively in describing the web services. Generally, it gave adequate description of the prototype, but authors feel it will not be sufficient enough to fully describe functional and non-functional attributes necessary for realistic service oriented approach. Since it does not provide adequate
description of versioning, security, quality of service and cost. Performance of the entire system is affected by the performance of subsystems or data sources. So to enhance performance, the architecture of each subsystem should be optimized which is very difficult and maybe impossible.

8. Conclusion and future work
Although the subsystems have heterogeneity in databases, data structures, functions, and required resources, the integration based on the proposed model succeeded to pass most of the heterogeneity problems, by serializing data to standard data format (XML) and then deserializing it on the target site. One of the most important advantage of the proposed model is its scalability i.e. the possibility to add new subsystems or other data sources without the need to modify the comprehensive system, or the legacy data source, or the subsystems. Successful data integration helps discovering relationships that enable making better and faster decisions, thus considerably saving time and money. Future research will concentrate on security issues, since all data transfer is in clear XML standard and through public network, which make the system piracy-prone. Also, the proposed model may be modified to include heterogeneous data space as well as data source.

8. REFERENCES