# USING WEB SERVICES TO DYNAMICALLY EMBED MEDICAL CONTENT IN A CLINICAL INFORMATION SYSTEM

Martin Luethi
*Picis, Inc., 9550 W. Higgins Rd, Rosemont, IL, 60018, USA*

Keywords: Web services, medical calculators, drug dosage calculators, dynamic user interface.

Abstract: This paper describes a prototype software application that makes use of Web services to retrieve medical content from an external Application Service Provider (ASP) service. The retrieved data is used by the system to dynamically generate user interface components within a user-customizable department information system and by clinicians to obtain results that will assist with medical decision-making. Medical forms are frequently built and rearranged by medical system administrators. From the perspective of clinical end users the content is seamlessly integrated and allows querying of dosages, interactions, and formulations using collected parameters such as age, weight, and physiological data. Whereas the use of standalone personal digital assistant (PDA) devices provides measurable benefits to clinicians, the feasibility of a seamless integration with user-customizable information systems has not been researched well. This paper describes one approach that could be taken to integrate such medical calculators with a web-based clinical information system. The solution described is not intended to represent functionality available in any actual or planned product.

## 1 INTRODUCTION

Medical content and knowledge such as drug interactions, medical calculators, drug dosing calculators, drug formulations, and illustrations provide a high benefit to clinicians and may decrease patient risk significantly (Knollmann et al., 2005). Medical knowledge has been made available in vast compendiums and in more recent days on the internet or as standalone applications on mobile devices such as PDAs. Whereas these applications provide a measurable benefit, they lack integration and documentation capabilities with clinical information systems that are concurrently used in the department. Consequently retrieved results have to be duplicated using the departmental procedures and systems in place.

Often departmental systems provide this medical content as well. It is either provided by the software vendor directly or external databases are purchased from vendors and then tightly integrated with the application. However, the conventional approach has two major disadvantages: First, changing content requires a new release or system maintenance cycle. Second, the integration is often static and does not allow the user to modify the clinical content.

The presented approach provides a solution to these problems. Using an ASP model, the content is directly queried from the content provider via Web services. Updates of data and newly added functionality are immediately available at sites where the system is in use. Further, the user interface (UI) is instantiated in an ad hoc fashion and the content is integrated within the dynamic structure. Updated content is immediately available for use and required UI elements are rendered. Patient data already available in the departmental system such as age, weight, and height is automatically pulled into the calculators. Once all necessary parameters are collected an additional Web service call is made to retrieve the calculated result based on the parameters provided by clinicians.

This paper also describes security challenges specific to the use of external Web services with medical data. Availability and reliability issues are discussed as well as general patient safety risks and testing strategies. Scenarios are discussed from the perspective of the service requester.

## 2 CLINICAL CONTENT

Medical knowledge is increasing rapidly. It is difficult for clinicians to keep up-to-date with new evidence-based studies, medical literature, and FDA-approved medications and their interactions, even though they should be considered during medical decision making. Prescription errors and adverse drug events are a leading cause for malpractice litigation (De Sousa, 1996). Improving access to this knowledge could lead to better decision-making, improve patient outcome, reduce costs, and increase bed utilization (Pronovost et al., 2002).

Portable devices, namely PDAs, have been used to make this knowledge accessible for clinicians (Lapinsky et al., 2004). The largest criticism of these systems besides data-related issues is their lack of integration with departmental information systems, their lacking ease of use, and the need to manually update the devices' databases. Content is preferably accessed in one single application and used on desktop computers when away from the bedside. Subsequently, examples are provided for often accessed content types.

### 2.1 Drug Dosage Calculators

Drug dosage calculators are used to calculate medication doses taking into account individual patient data such as weight and age and other relevant information. The applied formulas may vary by treated disease and type of medication.

The following table presents an example of a simple calculation formula (1) for IV Dobutamine. The parameters required for the calculation a-d are listed.

$a$ = drug amount (mg)
$b$ = per fluid (ml)
$c$ = weight (kg)
$d$ = dose (mg/kg/min)
$x$ = drip (ml/min)

$$x = a / b * c * d \tag{1}$$

### 2.2 Medical Calculators

Medical calculators are used to generate a numeric or textual result that is used for further medical decision making. Typical examples are the calculation of probabilities of diseases and outcomes, index numbers, and scales. They can have one or multiple data entry parameters, which can be numeric or chosen from a selected textual option.

The following example is Basal E Expend for females (2), which requires weight, height, and age as input parameters and results the calorie use per day.

$a$ = weight (kg)
$b$ = height (cm)
$c$ = years
$x$ = kcal/day

$$x = 9.6 * a + 655 + 1.8 * b - 4.7 * c \tag{2}$$

### 2.3 Drug Interactions

The intake of more than one drug can result in severe interactions. Before prescribing or applying drugs, possible interactions have to be checked. With the current prototype several dozens can be checked simultaneously and the seriousness of interactions is rated on a scale of 1-5. Each interaction is accompanied with an explanation of the effects of the interaction.

As an example the interaction between Aspirin and Atenolol is provided. The result contains the severity of the interaction and its description. To obtain the desired result the drug identifiers have to be provided.

$a$ = Aspirin (drug 1)
$n$ = Atenolol (drug n)
$x$ = drug interaction (level, message)

$$x = a, \ldots, n \tag{3}$$

### 2.4 Other Content

A simple case of medical content is the provision of textual and visual data regarding a contextual topic. The difficulty is to provide an appropriate search term to retrieve the data dynamically depending on the context and to link it with the application.

#### 2.4.1 Text Content

Medical text content requires one or more search terms and a product code. Access of an encyclopaedic database is currently not supported. The service returns a result of drugs and products in textual representation.

#### 2.4.2 Illustrations

Images are used for educational purposes. They provide explanations and textual values (Figure 1). Images are static and do not need any input parameter except the expected graphic format and the illustration identifier. The binary output is embedded in the XML body as base-64 encoded string.

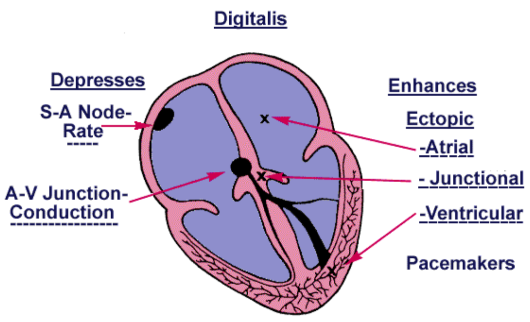


Figure 1: Example of a medical illustration.

## 3 WEB SERVICES

A Web service is a software system that is designed to support interoperable machine-to-machine interaction over a network. Its interface is described in a machine-processable format so that other systems can interact with the service exchanging messages (W3C 2004). For a successful message exchange mechanics and semantics of a message have to be defined and mutually agreed upon.

The mechanics describe the message format, data types, transport protocols, and transport serialization protocols. Commonly these definitions can be found in a Web services Description Language (WSDL) document. The semantics give the message meaning and its definitions are less standardized and not necessarily written or negotiated.

The use of Web services in a Service-Oriented Architecture (SOA) allows for the flexible and loose coupling of applications in platform and language independent manner. It accelerates application integration and facilitates rearranging the sequence of required components as needed. However, communications of distributed systems are naturally less reliable and slower than direct code invocation and shared memory. Further, there are additional security and privacy consideration in particular for services transmitting confidential patient information and cases in which the Web service data is directly used for patient treatment.

Application of Web services in clinical areas and related-fields as opposed to payer and business areas are not very common. A few examples of their use have been described (Cheng, Yang, Chen, Chen, & Lai, 2004; Eaton, 2006; Jiang, 2004; Mykkänen, Riekkinen, Sormunen, Karhunen, & Laitinen, 2007).

### 3.1 Standards

The Web services programming stack (Figure 2) is a collection of standardized protocols that are used to implement Web services (Gottschalk, Graham, Kreger, & Snell, 2002). The foundation of distributed Web services is the network layer. It is based on HTTP allowing accessing the service from within a hospital network through its firewalls. On top of the network layer, an Extensible Markup Language (XML)-based messaging protocol ensures the mechanics of the Web services. The Simple Object Access Protocol (SOAP) is used to invoke method calls for the content needed. Descriptions in WSDL give details of the mechanics and the available services of the interface. Available services and their location (URI) are known to the service requesters by informal means. However, publication and discovery of Web services could be implemented using the Universal Description Discover and Integration (UDDI) standard. The service flow layer allows the composition of services to an application. It has been implemented proprietarily.

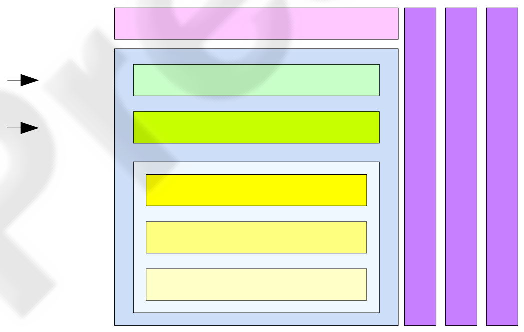


Figure 2: Web services programming stack.

Security, manageability, and quality of service apply to all the discussed layers.

### 3.2 Security and Reliability

Web services are suspect to common threats of networked data exchanges such as message alteration, confidentiality, man-in-the-middle, spoofing, denial of service, and replay attacks (W3C 2004). Such attacks can be prevented by point-to-point security technologies that secure the transmission from one Web services to another or, if routed through multiple services, by end-to-end security measures.

For the use of medical content as described, message alteration and denial of service are possible security risks. However, point-to-point technologies are sufficient to secure the transmission since messages are not routed among service providers. To achieve this security, password authentication of the requester and encryption of the communications link using the Secure Socket Layer (SSL) protocol are considered sufficient. Confidentiality attacks and privacy concerns are not relevant for the discussed system since there is not enough data transmitted to identify a patient. Age and gender in conjunction with physiological data alone is in most cases not sufficient to deduce the identity of a patient.

Additionally, since the information system is used in an information technology (IT) environment that is not directly controlled by either party, it is important to consider that Web service calls need to be able to flow through organizations' firewalls. It can be tunnelled through existing ports and protocols, which makes this less of an issue.

Requesting services over a public network involving a requesting and providing agent, is intrinsically less reliable than local services. It can be distinguished between message reliability and service reliability. If no answer can be retrieved an error message is thrown. It would be possible to offer a fail-over service provider that could be hosted within a hospital network. Before values are submitted to the service, the values are checked on the client. They are checked for their validity and reviewed by the clinician when documenting a visit. Failures are not safety critical.

## 4 APPLICATION

The Web services consuming software application is a web-based emergency department information system (EDIS) that provides functionality required by clinicians such as patient tracking, registration, clinical workflow and task management, physician and nursing documentation, orders, and post-disposition management. The server-based application is interfaced with several other information systems in the hospital such as the main registration system, laboratory systems, pharmacy systems, billing and coding systems, etc.

The workflow in emergency departments depends on forms and documentation requirements vary significantly from site to site. Therefore, the clinical documentation component allows medical administrators to build and customize their own forms. The form building process may be performed via a drag-and-drop interface where UI elements are selected from a library of elements. Further, metadata such as charge codes, clinical identifiers, risk elements, and wording data can be attached. The content elements could be stored in a library.

Once customizations have been finished the content is published and is used for documentation by clinicians. Documentation is complaint-driven and can consist of many different forms, whose sections are selected dependent on patient attributes such as age and gender.

### 4.1 Architecture

Clinical documentation forms are created in the content builder component. The elements that make up a form are stored in a relational database containing the metadata, but no presentation layer data. The data is separated from the presentation layer, which allows using the form definition data in a different context such as for PDAs and paper forms for digital pen devices.

Figure 3 illustrates the process of using the Web service. In the content builder component the clinical form is assembled. At the location where a medical or dosage calculation is required, a generic calculator widget is positioned. A list of available calculators is presented to the user. This list could be retrieved through the Web service also, but is kept locally for performance and manageability reasons. The current prototype can contain hundreds of different calculators.

When the calculator is selected a Web service call is made to the ASP service with the needed calculator identifier (1). The Web service responds with an XML document containing the field data for the requested calculator. In the response, each required data field is specified with label, variable name, and data type. If the data type is an enumeration all possible options including their text and data representation are included. Further the formula used to calculate the result is provided in Reverse Polish Notation (RPN), where applicable.

The response is parsed and the UI elements are dynamically generated (2). Since the presentation layer is built using an object-oriented hierarchy of classes the received data has to be mapped so that the necessary objects can be instantiated. The entire form containing also non-Web service retrieved elements is then rendered to be used in the application. During form creation, response time and performance are not a primary concern since the form is compiled into an intermediary format and cached locally.

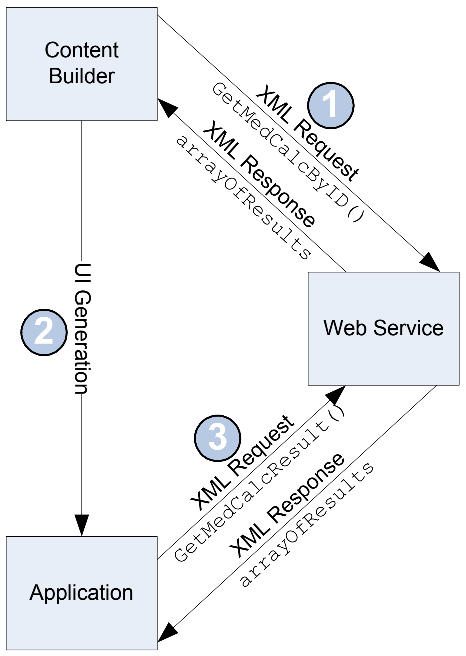


Figure 3: Architecture.

The response is parsed and the UI elements are dynamically generated (2). Since the presentation layer is built using an object-oriented hierarchy of classes the received data has to be mapped so that the necessary objects can be instantiated. The entire form containing also non-Web service retrieved elements is then rendered to be used in the application. During form creation, response time and performance are not a primary concern since the form is compiled into an intermediary format and cached locally.

During system use by clinicians the required parameters are filled in and an Asynchronous JavaScript and XML (Ajax) request is made to the Web service to calculate the results (3). The score/result field is populated with the result as soon as available. The value can be a string or numeric type depending on the calculator. Due to the asynchronous nature of the call, the response does not have to be instantaneous and some delays are tolerable. If the service is unavailable or does not provide the expected result an error message is thrown. Alternatively, the request could be routed to a different service, handled locally, or queued for later retrieval.

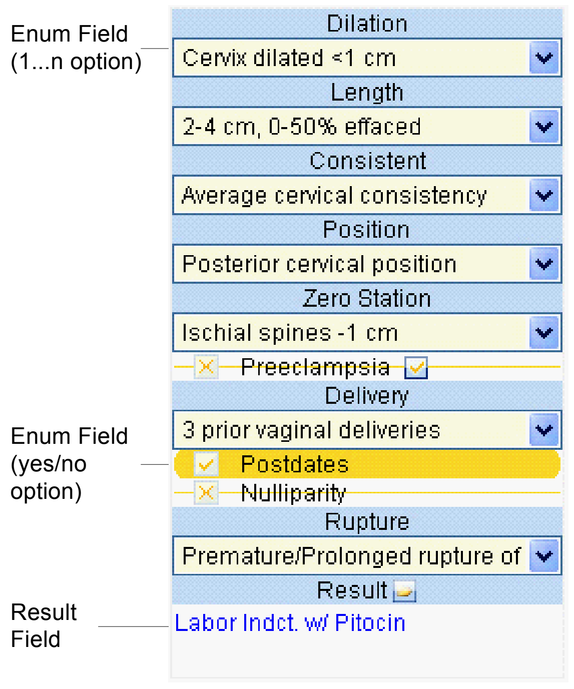
## 4.2 Visualization

The XML response of the Web service containing the calculator data looks as presented below. Additionally there is a WSDL service description available that describes the expected response.

```
<MedCalcInfo …>
  <Error />
  <MedCalcID>bish</MedCalcID>
  <Title>Bishop</Title>
  <Inputs>
    <Input>
      <Name>Dilation</Name>
      <Var>a</Var>
      <InputType>Enum</InputType>
      …
      <Options>
        <Option>
          <Text>Cervix … <1 cm</Text>
          <Value>0</Value>
          <Selected>true</Selected>
        </Option>
        …
      </Options>
    </Input>
    …
  </Inputs>
</MedCalcInfo>
```

The response is parsed and the object hierarchy is instantiated. The response also specifies the default values for enumerations as well as data types that are required. The rendered interface enforces data type checking using client-side validation. The UI also consists of data types unknown to the Web service. For example, in medicine the concept of pertinent negatives is constantly encountered. This means that such an element can assume three different states: not answered, positive, or negative. In the UI this concept is implemented as a checkbox that can be negated by cycling through its different states by clicking the mouse repeatedly. Positive values are represented in a circled manner; negative values are displayed slashed. This application specific UI component could be considered a type of “SuperCheckbox”.

Figure 4 shows how the different data types are displayed for the end users. This concept has to be mapped from the Web service response. Enumerations that are specified as yes/no option value are visualized as SuperCheckbox instead of a dropdown.



Dilation
Cervix dilated <1 cm
Length
2-4 cm, 0-50% effaced
Consistent
Average cervical consistency
Position
Posterior cervical position
Zero Station
Ischial spines -1 cm
✗ Preeclampsia ✓
Delivery
3 prior vaginal deliveries
✓ Postdates
✗ Nulliparity
Rupture
Premature/Prolonged rupture of
Result
Labor Indct. w/ Pitocin

Figure 4: Dynamically generated user interface.

The UI element types are selected according to the criteria presented in Figure 5.

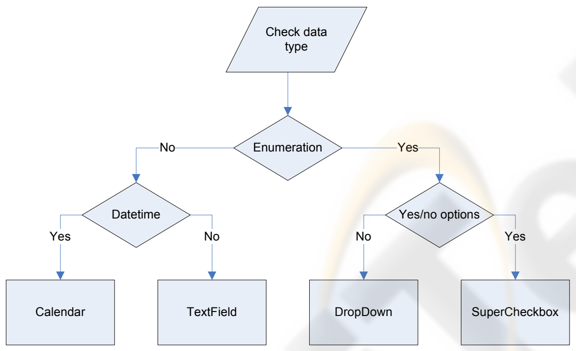


Figure 5: Data type mapping.

Additional to the field definitions received from the Web service, a result field is rendered to present the calculated scores or results. The results and values do not only have to be submitted to the Web service for calculation but also documented in the EDIS according to site-specific guidelines. The request and response are validated by the service whether they lie in a sensible range.

## 4.3 Integration

Many values that are used in calculators are already available in the system either through manual data entry using different forms or through data interfaces with external systems. Such values include vital signs, weight, height, age, gender, etc. These values can be automatically imported and pre-selected in the appropriate UI elements. To provide real semantic integration, a system would require the use of a common nomenclature. String parsing, mapping, and data type checking could be applied, which in conjunction would provide the required reliability.

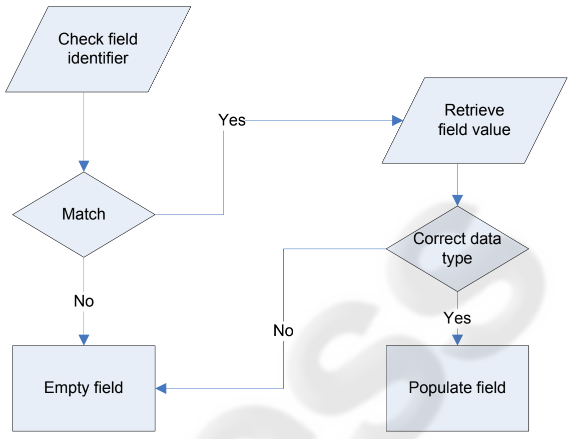


Figure 6: Populate fields with existing data.

The quality of the described system can be assured by automated unit tests that query the Web services for several test cases. They also record the access time and latency of the service response time. The UI is also tested via a test automation framework.

## 4.4 Measurements

As indicated above, response times and reliability of the discussed Web services are of some concern. This is in particular the case for the calculation of medical results. Whereas some delays are acceptable due to asynchronous calls, response times should generally not be beyond a 4 second threshold.

The measurements were taken querying the Web services over the internet over a distance of 14 hops. The average roundtrip time was 43 milliseconds. The services were tested with 1 to 20 concurrent clients. For each client 100 samples were taken. The infrastructure is scalable but further prototype development is required to identify practical limits.

Table 1 shows the results for retrieving the calculator data during form administration. During the test, all requests could be completed successfully. Generally not more than 5 concurrent users are expected to use the administration tool at any given time. Rendering times of the data are negligible.

Table 1: Response times in seconds of calculator service.

| Clients | Average | Minimum | Maximum | Deviation |
|---|---|---|---|---|
| 1 | 0.4930 | 0.3750 | 0.7500 | 0.0638 |
| 5 | 1.8393 | 0.4220 | 2.5310 | 1.1898 |
| 10 | 3.7522 | 0.4840 | 12.3140 | 1.8493 |
| 20 | 8.8832 | 0.3590 | 62.3220 | 5.7396 |

Table 2 lists the response times of the Web service that calculates the results. It performs slightly better using the same number of concurrent clients. An acceptable wait time is arbitrarily set at 4 seconds. To process more concurrent clients beneath that threshold the infrastructure would need to be load-balanced.

Table 2: Response times in seconds of result service.

| Clients | Average | Minimum | Maximum | Deviation |
|---|---|---|---|---|
| 1 | 0.6528 | 0.3120 | 3.0160 | 0.3870 |
| 5 | 3.1357 | 0.3440 | 14.1410 | 3.8315 |
| 10 | 3.4759 | 0.4530 | 12.9840 | 1.3377 |
| 20 | 7.8717 | 0.6250 | 27.4390 | 3.3215 |

Table 3: Conclusions.

| **Advantages** |
|---|
| Immediate updates of frequently changing medication information and no maintenance of additional servers. |
| Vast amount of available medical content for immediate use with minimal integration effort. |
| Seamless integration with clinical documentation. |
| Ability of user customization of clinical content. |
| **Disadvantages** |
| Dependence on external service. |
| Potential service outages and response time problems. |
| Regulatory and liability implications need to be determined. |

## 5 CONCLUSIONS

This paper showed how medical content can be dynamically integrated into a clinical information system. The primary challenge was the seamless integration of external content into a customizable UI. The components are rendered on-demand but still can be transparently arranged by medical system administrators.

Whereas the retrieval of the form definition data is only performed during system customization, the retrieval of results occurs during the documentation by clinicians and requires a higher level of availability. Reliability was found to be high for the described service and security requirements can be fulfilled by point-to-point security technologies.

It was found that the advantages generally outweigh the disadvantages. The potential of having a vast amount of external content available without a significant integration effort was considered to be the biggest advantage. Further, having this content available using an ASP concept as opposed to in-house hosting reduces maintenance to a minimum for the service requester.

Future research is needed to address the acceptance of this content by clinicians. Also, the response times and reliability over a high number of customers needs to be considered. In a next step, patient history from other systems could be integrated using Web services across network boundaries. Also, legal and regulatory concerns need to be addressed.

## REFERENCES

Cheng, P., Yang, C., Chen, H., Chen, S., & Lai, J. (2004). *Application of HL7 in a collaborative healthcare information system.* Paper presented at the Engineering in Medicine and Biology Society, 26th Annual International Conference of the IEEE.

De Sousa, B. (1996). Prescription problems: clinical pharmacology and lawsuits. *Texas Medicine, 92*(5), 57.

Eaton, A. D. (2006). HubMed: a web-based biomedical literature search interface. *Nucleic Acids Research, 34, W745–W747*.

Gottschalk, K., Graham, S., Kreger, H., & Snell, J. (2002). Introduction to web services architecture. *IBM Systems Journal 41(2), 170-177*.

Jiang, K. (2004). *Integrating clinical trial data for decision making via web services.* Paper presented at the Engineering in Medicine and Biology Society, 26th Annual International Conference of the IEEE.

Knollmann, B., Smyth, B., Garnett, C., Salesiotis, A., Gvozdjan, D., Berry, N., Lee, H., & Min, D. (2005). Personal digital assistant-based drug reference software as tools to improve rational prescribing: Benchmark criteria and performance. *Clinical Pharmacology & Therapeutics, 78*(1), 7-18.

Lapinsky, S. E., Wax, R., Showalter, R., Martinez-Motta, J. C., Hallett, D., Mehta, S., et al. (2004). Prospective evaluation of an internet-linked handheld computer critical care knowledge access system. *Critical Care, 8*(6), 414-421.

Mykkänen, J., Riekkinen, A., Sormunen, M., Karhunen, H., & Laitinen, P. (2007). Designing web services in health information systems: from process to application level. *International Journal of Medical Informatics, 76*(2-3), 89-95.

Pronovost, P. J., Angus, D. C., Dorman, T., Robinson, K. A., Dremsizov, T. T., & Young, T. L. (2002). Physician staffing patterns and clinical outcomes in critically ill patients. *The Journal of the American Medical Association, 288*(17), 2151-2162.

W3C (2004). *Web services architecture*. Retrieved 2007-06-11, 2007, from http://www.w3.org/TR/ws-arch/wsa.pdf