Connecting Medical Informatics and Bio-Informatics
R. Engelbrecht et al. (Eds.)
ENMI, 2005

Designing Web Services in Health Information Systems: From Process to Application Level

Juha Mykkänen[a], Annamari Riekkinen[a], Pertti Laitinen[b], Harri Karhunen[c], Marko Sormunen[a]

[a]University of Kuopio, HIS R & D Unit, IT Service Centre, Kuopio, Finland
[b]University of Kuopio, Shiftec, Dept. of Health Policy and Management, Kuopio, Finland
[c]University of Kuopio, Dept. of Computer Science, Kuopio, Finland

Abstract

Service-oriented architectures (SOA) and web service technologies have been proposed to respond to some central interoperability challenges of heterogeneous health information systems (HIS). We propose a model, which we are using to define services and solutions for healthcare applications from the requirements in the healthcare processes. Focusing on the transition from the process level of the model to the application level, we also present some central design considerations, which can be used to guide the design of service-based interoperability and illustrate these aspects with examples from our current work in service-enabled HIS.

Keywords:
Health information systems, Services, Integration, Interoperability, Interfaces

1. Introduction: Service-oriented architectures for HIS?

Central challenges for Health Information Systems (HIS) include lack of reuse, redundant data and functionality and heterogeneous technologies. Adaptation to new requirements and multiple medical cultures and integration with existing systems is difficult in constantly changing health environment. [1,2]. There are also growing demands to coordinate or automate various processes, and to find common description and ways to execute them via electronic transactions to support seamless and quality care to the patients [3-6].

Service-oriented architectures (SOA) have been suggested as a design and technology strategy of complex enterprise application environments [7,8]. SOA includes practices and frameworks that enable application functionality and information to be provided and consumed as services [8-10]. From technical viewpoint SOA is essentially a collection of software services that communicate with each other over network to pass data or to coordinate some activity. Services can be implemented using different technologies, and can encapsulate functionality and information from existing applications, thus allowing the reuse of the existing IT investments. [10,11,7]. Thus, the main expected benefits of SOA seem to support well the requirements of heterogeneous IS domains such as healthcare.

We argue that a SOA-based approach must support different *interoperability needs*, *different degrees of integration*, different *messaging patterns*, and different phases of the

development process. In this paper we propose a model for design considerations that should be perceived in service-based application development and integration for HIS.

2. Methods: Web services and service design approach

Web services are software components or applications which interact with one another using XML-based Internet technologies [12,5,13]. They offer a platform-neutral interfacing and communication mechanism, have wide infrastructure support and have significantly increased the interest in SOAs [7]. However, two distinct approaches to web services can be identified [14]. The *procedural* approach focuses on bottom-up application integration. It is based on the architecture of the existing remote procedure call (RPC) middleware, and current SOAP, WSDL and UDDI specifications. The *document-oriented* approach focuses on top-down business exchanges, and tries to describe the elements of (commercial) exchange, including the technology solutions. It is based on electronic commerce, documents and loosely-coupled messaging, and includes e.g. ebXML specifications.

We are using a high-level service-oriented approach to analyse healthcare processes and requirements and to define services and solutions, which support the needs of health professionals and patients (see Figure 1). We have adapted the viewpoints from Gartner's four-platform model [9]. The *producer platform* includes tools and technologies for Web services. The *provider platform* hosts services in the enterprise. The *management platform* contains solutions for managing the infrastructure for Web services. The *consumer platform* consists of techniques by which the services are used. The four-platform framework, however, focuses mainly on technical issues and does not represent the way the organizations really use Web services [15]. In comparison to [9], the solutions require a broader approach in which the requirements of the processes and the applications drive technical solutions. We examine the design considerations on three levels – *process, application and platform*, from the four viewpoints – provider, consumer, production and management. On each level the viewpoints require different kinds of considerations. The process level is the most relevant to the healthcare domain. While moving from the process level to the application level and to the platform level, the design considerations become more independent from the healthcare domain and more technology-oriented.

The viewpoints on the *platform level* have been presented above. On the *application level*,

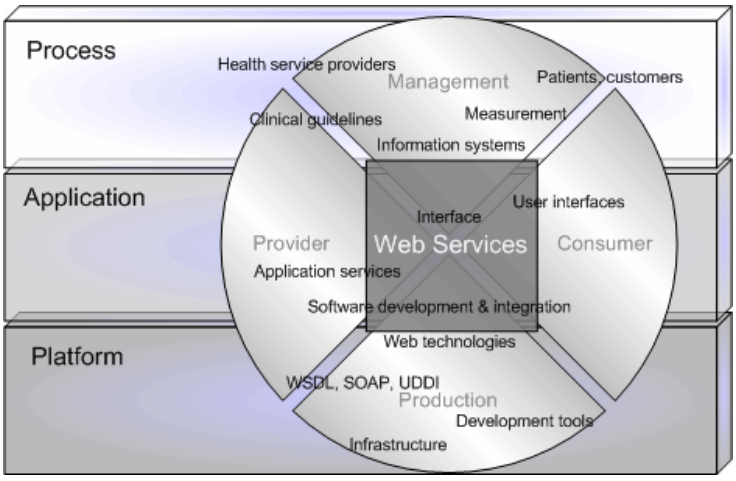

Figure 1. Service-oriented analysis and design approach on Process, Application and Platform levels and some example considerations on these levels.

software services are providers, and applications that use services are consumers. Management is the control and maintenance of the applications and the services, and

production refers to the tools, technologies and methods used in the development on the platform level.

On the *process level* the service providers are personnel who provide health services to their customers, including patients. The management of the processes, and services themselves are supported by different conventions and especially information systems, which links process level to the application level. The process level must drive all the application design decisions: its requirements must be addressed in the design of the solutions and services on the application level using the tools and technologies on the platform level. On each level, consumer, provider, management and production issues must be addressed to consider the needed aspects of a given solution.

3. Results: Design considerations for healthcare software services

In this paper, we consider the linkage of process level to the application level design. We present several questions, whose answers guide the design of different types of service-oriented solutions. To define basic solutions for service specifications, we apply the integration specification process in [16]. In particular, we focus on the phase where the main design decisions are made. This phase has the following steps:

1. Select the set of *requirements* to be included in the services, and the basic *integration model*.
2. Evaluate and select the content *standards* and other specifications to be utilised. These standards may have explicit or implicit consequences to the further design.
3. Link the *information and functions in the participating systems* to the requirements.
4. Identify and name the *participating components* (e.g. providers and consumers).
5. Map the selected requirements to the *responsibilities* of the identified components.
6. *Refine the interaction solution*, for example the deployment and interaction style.
7. Identify *integration points* in the architecture of the participating systems. Refine the responsibilities of the components, identify possible extension needs.
8. Specify *interaction sequences*, which may also contain user interaction.
9. Specify service *interfaces*. Information contents and semantics are specified as e.g. document definitions in document style, and as parameter definitions in procedural style, and functional needs as e.g. operation names or document/message types.
10. Define features as either *required, optional* or *extensible* from all implementations.
11. Refine the *requirements* for implementations or further technical specifications.

We propose the following questions to guide the solution specification, especially in steps 1 and 6. The design implications of different options are also discussed briefly. The list in non-exhaustive, but according to our experiences, it covers the main considerations to guide the design of services.

a. What is the main *integration model*? Integration models can be *information-oriented, service-oriented, user-driven or process-oriented* (adapted from [17]). If the main requirements are focused on information sharing or transfer, document approach with declarative messages or documents is often selected. If the main requirements are computationally oriented, e.g. shared functionality, a service (or API) interface with imperative operations is a natural fit. If the main requirements focus on usability or a consistent view to information or several applications, it should be considered whether to use services or other approach, although service-oriented context management interfaces or visually-oriented services such as WSRP (Web Services for Remote Portals) [18] can also be used. If the main requirements are focused on processes, a breakdown of the steps of the process into smaller (information or service) interactions, and the use of process-oriented specifications such as BPEL (Business Process Execution Language) should be considered.

b. What is the required level of *adaptability*? Changes can occur in the information contents, the context of the system, or the system itself. In the procedural style, exact parameters for operations are defined. This results in precise standard interface, and can be supported by automated tools. In the document style, there are more chances for adaptation. To support maximum flexibility, a two-level approach with e.g. (meta) reference model and constraining archetypes can be selected [2], but this type of solution may impose difficulties for existing systems due to the changes in the development approach. Requirement may also be to encapsulate changes, e.g. to support legacy migration, or to provide content-neutral operations, which may carry different information (e.g. templates) in different settings.

c. Is the goal to use a *shared/unified* model, or *federation/mediation* between the participants? In information-oriented integration, a shared document repository and message-based document transfer using messaging broker to copy information from one system to another are examples of unified and federated solutions, respectively. Common service interfaces such as those of OMG Healthcare DTF [4] are examples of shared service-oriented model.

d. What is the required level of *granularity*? For rapid and repeated interaction, e.g. related to the atomic user actions, small procedure calls are suitable as parts of a larger service. For batch-oriented or cross-organizational transactions, with large amounts of diverse information, document-oriented messaging can provide benefits.

e. How *tight integration* is aimed at? Typically in a shared service, only an identification or a reference to a given entity is given, whereas in loose integration (e.g. between organizations), all the relevant information is transferred between the systems. Tight integration often requires guaranteed availability of the invoked service, whereas in document style messaging, guaranteed delivery is aimed at. Also bidirectional invocations and strict data typing tighten the integration solutions.

f. What is the *number of consumers and providers*? In RPC/API style, point-to-point request/response model is usually used between the consumer and the provider, and registries are used for addressing. In document style, mediating brokers or buses can route and copy the messages to several consumers or providers. This requires additional addressing and routing specifications and infrastructure.

g. What is the *message exchange pattern*, e.g. is a response needed and should it be immediate? While RPC is inherently designed for user-driven call-reply with quick response, this can also be implemented with document-oriented services.

h. Does the provider need to maintain consumer-specific *state*, *context* or *session* between invocations? Common design recommendation for services is to avoid stateful services to ensure performance. However, state management may be an explicit requirement for a service (e.g. user-specific context repository, authorized session management or to maintain state in lengthy processes).

To efficiently solve design problems, answers should be specified for each of these questions. After this it is relatively straightforward to select technologies and tools capable of supporting the needed solutions, or to select another approach instead of services.

4. Discussion: Examples of different types of service specifications

In Table 1, we have compared three solutions for software services from the HIS projects we have been involved in, using the approach above. DRG classification services are used for assessment of resource utilisation in healthcare facilities for billing or care assessment. EPR archive interfaces support the storing of clinical documents in archives. Context

repository interfaces are used to provide single sign-on and application synchronization to improve the usability of various clinical and administrative applications.

As seen in Table 1, the answers to the specified questions point the designs for different cases towards different types of services: in DRG and Context interfaces, API/RPC approach is used, whereas documents and integration platforms are utilised for EPR archiving. Procedural services are useful in interactive requirements and e.g. within one organization, whereas document-oriented interfaces are often used in situations where the integration is more loose and flexible, and the infrastructure or applications are controlled by different organizations. For stabile functional and computational requirements where the participants are well-known, API-style invocations are used, where automated tools hide many technical aspects. For cross-organizational, federated, or one-to-many solutions, document style and messaging platforms are needed, typically along with additional work. Such solutions include Enterprise Service Bus (ESB) approach [3], which includes also SOA and web services. In any selected integration model, standards ease the implementation and improve the tool support.

Table 1. Comparison of three service scenarios.

Scenario	**DRG classifier interfaces**	**EPR archive interfaces**	**Context repository interfaces**
Requirement	Produce information based on diagnosis and other information for the assessment and comparison of resource consumption	Provide a common interface for archiving different types of clinical documents related to a patient (e.g. referrals, prescriptions)	Maintain user-specific context information for several applications (user, patient, encounter etc.)
Integration model	service	information	user
Adaptability	static, parameters well-known	dynamic, different types of documents, support for local variation	static interface, extensible subject definitions
Unified or federated	unified model (common service)	unified model (common archive) and federated model (may need local content translations)	unified model (common service)
Granularity	fine-grained operations and parameters	large-grained documents	fine-grained operations and parameters
Tight/loose	tight, common service must be available	loose, archive may be queued, notification	tight, but applications work also without the service
Consumers / providers	many consumers use one provider	many consumers use one provider, variations exist	many consumers use one provider
Message exch. pattern	immediate response needed	no immediate response needed, guaranteed delivery needed	immediate response
State	stateless	stateless	stateful
Additional conside-rations	can be used in interactive and/or batch-oriented way	clinical documents, simple transfer/ archiving interface, additional digital signatures	low implementation threshold for participating applications, no bi-directional invocation

5. Conclusions and future work

We presented an approach and a set of considerations for analysing and designing software services in healthcare applications. We illustrated the approach with examples of different types of solutions. Our model considers essential differences in service designs to support

different requirements in the healthcare domain. This model alone does not cover the requirements elicitation on the process level, or the most technical platform considerations. We are using the presented approach to define several healthcare-specific solutions, and also participating in the standardization of service interfaces and specification approaches.

6. Acknowledgements

This work is part of the SerAPI and SOSE projects, funded by the National Technology Agency of Finland TEKES grants no. 40437/04 and 70070/04 together with a consortium of software companies and hospitals.

7. References

[1] Van de Velde R. Framework for a clinical information system. Int J Med Inf 57 (2000) 57-72.

[2] Beale T. Archetypes: Constraint-based Domain Models for Future-proof Information Systems. OOPSLA 2002 workshop of behavioural semantics, 2002.

[3] Chappell D. Enterprise Service Bus. O'Reilly, 2004.

[4] OMG Healthcare Domain Task Force. CORBAmed Roadmap, Version 2.0 (draft). OMG Document CORBAmed/2000-05-01; 2000.

[5] Wangler B, Åhlfeldt R-M, Perjons E. Process Oriented Information Systems Architectures in Healthcare. Health Informatics Journal 2003: 4: pp. 253-265(13).

[6] Ryynänen O-P, Kinnunen J, Myllykangas M, Lammintakanen J, Kuusi O. Suomen terveydenhuollon tulevaisuudet. Skenaariot ja strategiat palvelujärjestelmän turvaamiseksi. Esiselvitys. Eduskunnan kanslian julkaisu 8/2004. In Finnish. (The Future of Finnish Health Care - Strategies and scenarios to secure health care services in Finland in the future).

[7] Natis YV. Service-Oriented Architecture Scenario. Gartner group, 2003. (28 Dec 2004.) Available at http://www4.gartner.com/DisplayDocument?ref=g_search&id=391595.

[8] Champion K. SOA: the new architecture that leverages the old. Developers.net, 2004 (28 Dec 2004.) Available at http://www.developers.net/node/view/106.

[9] Smith D. Web Services Architecture: A Four-Platform Framework. TopView, 16 May 2002. Gartner group, 2002.

[10] Plummer D. A Closer Look at Service-Oriented Architecture. Bea dev2dev, Bea Systems, 2003. (3 Jan 2005) Available at http://dev2dev.bea.com/trainingevents/webinars/Gartner_02_14.jsp.

[11] Sprott D. Moving to SOA. Cbdi forum, 2003. (3 Jan 2005) Available at http://roadmap.cbdiforum.com/reports/soa/ index.php.

[12] Aissi S, Malu P, Srinivasan. E-Business Process Modeling: The Next Big Step. IEEE Computer 2002:35(5):55-62.

[13] Turner M, Zhu F, Kotsiopoulos I, Russel M, Budgen D, Bennet K, Brereton P, Keane J, Layzell P, Rigby M. Using Web Service Technologies to create an Information Broker: An Experience Report. Proc. of the 26th Int. Conf. on Software Engineering, IEEE, 2004, pp. 552-561.

[14] Alonso G, Casati F, Kuno H, Machiraju V. Web Services Concepts, Architectures and Applications. Springer, 2004.

[15] Schmelzer R.& Bloomberg J. Retiring the Four-Platform Framework for Web Services. Zapthink, 2003. (28 Dec 2004.) Available at http://www.zapthink.com/report.html?id=ZAPFLASH-12032003.

[16] Mykkänen J., Porrasmaa J., Rannanheimo J, Korpela M. A process for specifying integration for multi-tier applications in healthcare. Int J Med Inf 2003:70(2-3):173-182.

[17] Linthicum D. Leveraging the heritage – Approaches to Integrating Established Information Systems. Intelligent EAI, April 15, 2003. CMP Media LLC, 2003.

[18] Reshef E. Building Interactive Web Services with WSIA & WSRP.Web Services Journal 2002:12(2):2-6.

7. Address for correspondence

Juha Mykkänen, University of Kuopio, HIS R & D Unit, P.O.B. 1627, Fin-70211 Kuopio, Finland, Juha.Mykkanen@uku.fi