Schreier G, Hayn D, Ammenwerth E, editors. Tagungsband der eHealth2011. 26.-27.Mai 2011; Wien. OCG; 2011.

SERVICE-ORIENTED ARDEN-SYNTAX-BASED CLINICAL DECISION SUPPORT

Fehre K[1,2], Adlassnig K-P[1,2]

Abstract

We describe a software architecture for clinical decision support systems. The core of the architecture is the Health Level Seven (HL7) Arden Syntax. Functionality and scalability of this architecture were proven in large routine applications for early detection and automated monitoring of hospital-acquired infections at the Vienna General Hospital. These systems automatically receive data from 15 intensive care units (ICUs) and generate indications as to whether various forms of ICU-acquired infections are clearly present in a patient, present to a certain degree, or absent.

Keywords – Clinical decision support, Arden Syntax, Web-services, Hospital-acquired infections, Moni/Surveillance-ICU and -NICU

1. Introduction

Clinical decision support systems (CDSSs) are software programs developed to assist physicians and allied health professionals in patient care. Extensive efforts have been made in medical institutions and software companies to establish clinically useful, well accepted, and viable CDSSs. However, given the growing complexity of patient care and the stringency of working hours for medical staff, extensive effort must be taken in deploying CDSSs to ensure that they become part of everyday clinical work processes. To this end, some CDSSs have met with varying degrees of success while others suffer from persistent problems which impede or prevent their successful clinical use.

The aim of this work is to develop an architecture that enables the clinician to access Arden-Syntax-based CDSSs in a transparent way, independent of platforms and programming languages. Such architecture will improve the integration of rule-based CDSS into existing and new systems as well as enable the clinician to use rule-based CDSSs as a service for the provision of knowledge.

1. 1. Arden Syntax for Medical Logic Systems

As defined by Wright and Sittig [12], the general evolution of software architectures for CDSSs may be divided into four phases: stand-alone systems, integrated systems, standards-based systems, and systems based on service models. Our approach falls into the category of standards-based systems because it is founded on Arden Syntax for Medical Logic Systems, which is approved as

1 Medexter Healthcare Gmbh, Vienna
2 Section for Medical Expert and Knowledge-Based Systems, Center for Medical Statistics, Informatics, and Intelligent Systems, Medical University of Vienna, Vienna

standard by the American National Standards Institute (ANSI) and is being further developed and maintained by the Health Level Seven (HL7) organization [3].

A rule-based CDSS consists of three parts: a knowledge base, an inference engine, and a communication mechanism. An Arden Syntax knowledge base (KB) consists of a set of units known as medical logic modules (MLMs), each of which contains sufficient logic for at least a single medical decision. An Arden Syntax MLM is a hybrid between a production rule and a procedural formalism and is designed to evaluate a single patient's data at a certain point of time. A more complex KB usually contains several MLMs which may mutually interact by calling each other or by importing declarations from each other. Read-and-write statements containing the so-called curly braces expressions are provided to permit interaction with the external environment; one such function would be loading patient data from the information system database of a host. These Arden Syntax constructs permit interaction with the environment. MLMs may be triggered by a direct call or after a specific event. The evaluation of an MLM after such an event may start immediately, may be delayed, or may start periodically. A more detailed description of Arden Syntax is provided by Hripcsak [4], which includes a tutorial for the novice Arden Syntax user.

When Arden Syntax was introduced, the purpose was to transfer KBs between medical institutions and share clinical knowledge. For this purpose, it is usually necessary to adapt the curly braces expressions used at the present institution to the technical environment and specific conditions of the second institution. This renders any MLM transfer inflexible and cumbersome. Approaches to alleviate or resolve this problem have been described by Jenders et al. [5, 6]. One suggestion is to establish a standard data model, such as HL7's Reference Information Model (RIM), in which members of a certain institution can map their individual and local vocabulary and database scheme. By this process, an MLM can phrase its queries in accordance with such a standard data model, which then allows those MLMs to be transferred to other institutions at which the same data model has been implemented.

1. 2. Service-Oriented Architecture

A service-oriented architecture is essentially a collection of services capable of mutually communicating with each other. Such communication covers simple data transport as well as coordination of two or more services to perform a more complex task. A service is a well-defined and self-containing function which does not depend on the internal state of other services or the current context. As long as the service is available within a certain environment, it does not matter where the service-running host is located. A web service is a service with a Uniform Resource Identifier (URI), whose public interfaces and data bindings are described by the Extensible Mark-up Language (XML) [9]. System frameworks based on functionalities providing web services apply standard communication protocols such as XML, Service-Oriented Architecture Protocol (SOAP), and Web-Service Description Language (WSDL). They use the Hypertext Transfer Protocol (HTTP) as their transport mechanism.

2. Technical Methods

2. 1. Arden Syntax Compiler and Engine

Our present Java-based web service system that fully implements HL7's Arden Syntax specification [3] is based on two essential components: an Arden Syntax compiler and an Arden Syntax en-

gine. The Arden Syntax compiler has implemented the complete version of Arden Syntax and compiles given MLMs into Java code, which is executable on the corresponding Arden Syntax engine. Furthermore, the Arden Syntax compiler generates a signature containing information for each compiled MLM about the following aspects: name, institution, and events. The Arden Syntax engine is able to run compiled MLMs (by MLM calls) and can handle events by executing all MLMs pertaining to the specific event. Moreover, the Arden Syntax engine approves the transfer of external objects resulting from the evaluation of curly braces expressions in the respective custom-specific mode. An Arden Syntax engine sub-component known as the MLM manager maintains the compiled MLMs and conceals the specific implementation of the MLM storage from the engine. In other words, the engine itself only takes over incoming requests from the framework (calls for MLMs or events), asks the MLM manager for the corresponding MLM objects, and invokes them in correct order. Thus, the Arden Syntax engine and the MLM manager jointly handle the requests from the framework which, in our case, is usually a medical host information system.

The Arden Syntax engine, the Arden Syntax compiler, and the MLM manager constitute a stand-alone system which can be run on a single host to verify, compile, and run MLMs.

2. 2. Arden Syntax Server

As Arden Syntax is meant to work closely with the host information system, the two must be connected. We developed a Java-based Arden Syntax server which encapsulates the Arden Syntax engine and provides interfaces to external systems.

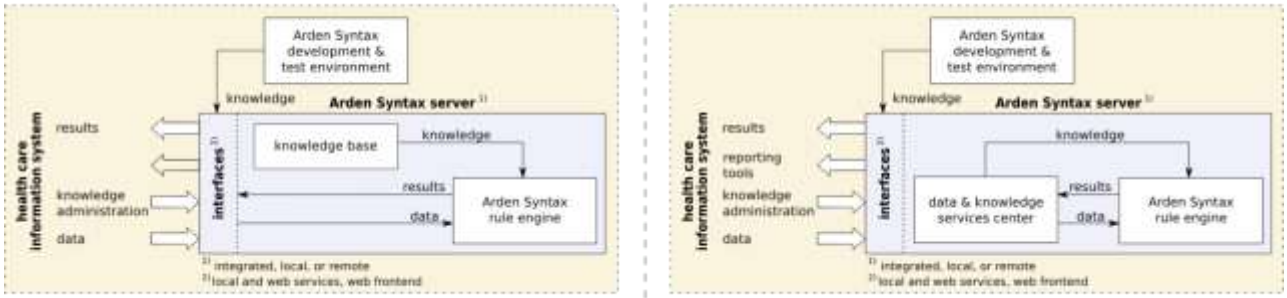

Figure 1: Two sample configurations of the Arden Syntax server

This Arden Syntax server performs the following tasks:

- *Concurrency*: The Arden Syntax server can handle concurrent calls from outside by executing the respective MLMs in parallel or by scheduling them in correct order.
- *Multiple executions*: The Arden Syntax server permits multiple executions of MLMs with only one call.
- *Asynchronic calls*: A requesting instance may use the Arden Syntax server to call an MLM and retrieve the results of the execution later.
- *Independence from Java*: By providing an external interface that communicates with the environment via XML messages, external resources do not have to use Java for their implementation.

Figure 1 shows two sample configurations of the internal structure of our Arden Syntax server and its external interfaces. The left configuration is the basic configuration, which contains only the engine and a simple store for MLMs. The right configuration additionally comprises a store for preprocessed and imported data, as well as intermediate and final results. The Arden Syntax server

provides a Graphical User Interface (GUI) that permits the end user to add and remove single MLMs or complete KBs (packages of MLMs). For specific applications, additional GUIs are added to present the calculated results to the user, permit reporting, or provide other tools.

To handle read-and-write requests from the Arden Syntax MLMs to the host information system, the Arden Syntax server was developed to process curly braces expressions and return the results of the requests to the Arden Syntax engine. If an MLM called by the above-mentioned process contains a read phrase within a curly braces expression, the Arden Syntax server has to interpret this expression and provide the required data to the MLM. This may be done by consulting the internal data services center, which contains preprocessed data. Alternatively, the Arden Syntax server may delegate the read call to a database-connected host. The latter usually implements a simple Arden Syntax host interface—a step that can be executed by host database specialists. This permits separation of data storage from the logic held by the KB-containing Arden Syntax server.

Communication with the Arden Syntax server follows the Arden Syntax server protocol, which is based on sending and receiving documents according to the rules of XML. The general approach of exchanging documents via the given protocol is to return the original document complemented by the results. We defined an XML scheme that specifies such communication between the Arden Syntax server and clients. The scheme proposes a document structure and syntax for each possible message type and the corresponding answer. *Figure 2* illustrates a sample document for calling an MLM.

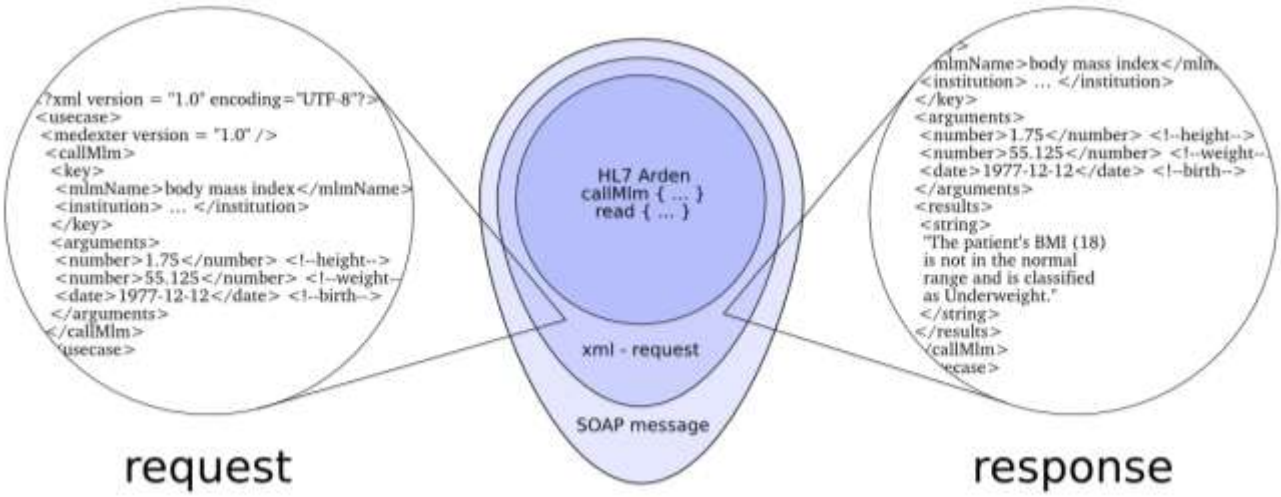

Figure 2: Wrapped Arden Syntax communication with request and response sample

It contains an entry labeled “key” used to identify the targeted MLM by its name and the creating institution. The “argument” entry contains the payload passed to the MLM as parameter. The answer to such a request is the original document extended by the result of the called MLM. In the above case, the resulting document is shown in *Figure 2*.

To enable communication with the Arden Syntax server over a network, each XML document is wrapped into a SOAP message which is transported via HTTP. As shown in *Figure 2*, messages between the participating hosts and the Arden Syntax server are three-layered messages, with Arden Syntax language constructs as their core. To be complete, the Arden Syntax server provides a description of its functionality given by a WSDL-formatted document. With this description, all subscribers in the network are able to discover the capabilities of the host.

3. Results with Moni/Surveillance-ICU and -NICU

Two extended implementations based on the architecture described above are in routine operation at the Vienna General Hospital in Austria: the Moni/Surveillance-ICU [2] and -NICU systems were developed and put into operation for early identification and continued monitoring of hospital-acquired infections at the intensive care units (ICUs) of the hospital. They contain two large KBs with about 75 and 150 MLMs, respectively. The first KB is for the detection of hospital-acquired infections in adult patients (12 ICUs with 96 beds), while the second serves to detect hospital-acquired infections in neonatal patients (3 ICUs with 36 beds). The system processes 6,000 to 8,000 data items daily, derived from the routine intensive care information system, and provides the hospital's infection control unit with an overview of the presence, partial presence, or absence of hospital-acquired infections at these ICUs. The gradedness of the results is determined by using fuzzy set theory and fuzzy logic to formalize and propagate the linguistic uncertainty of medical terms and the biological and practical uncertainty of medical conclusions [1]. Both KBs contain complex and elaborate MLMs. This was necessary because the applied MLM sets are based on complex definitions (derived from the definitions of healthcare-associated infections such as those by the Centers for Disease Control and Prevention in Atlanta, USA, the European HELICS criteria, or the German KISS definitions).

As shown by Koller et al. [8], automated surveillance by Moni/Surveillance-ICU is much faster and less dependent on human factors than conventional (manually operated) surveillance. In the above study, the overall accuracy of Moni/Surveillance-ICU was shown to be 97% (sensitivity 90.3%, specificity 100 %); the undetected cases were due to missing data in the patient database. The authors analyzed 1007 patient days and compared the results generated by Moni/Surveillance-ICU with those generated by trained surveillance staff. Furthermore, they compared the time taken for conventional surveillance and the time taken for Moni/Surveillance-ICU analysis.

The present Moni Arden Syntax server includes a database component to temporarily store both medical input data (partially preprocessed) of the patients and yielded intermediate and final results for them. It also contains a reasoning component which provides backward explanation of the gained results to the infection control team, and an analysis component to log reasoning steps for analysis and continued maintenance of knowledge.

Thanks to the advantages of the service-oriented architecture, both KBs – for adult as well as neonatal ICU patients – coexist harmoniously. The KBs were installed consecutively. This could be done without changing any of the essential components of the architecture. "Only" a new data input source had to be added.

4. Discussion and Conclusion

We have established an architecture which permits integration of CDSSs into existing hospital environments or that of other institutions by removing some of the disadvantages mentioned by Wright and Sittig [12]. The system was developed as a web service application and fully implements the Arden Syntax specification [3]. An Arden Syntax compiler and an Arden Syntax engine written in Java form the core around which an extended system was built. The latter is highly flexible and can be easily integrated for communication with external host systems. In accordance with the principles of a service-oriented architecture, the individual components can communicate with each other on the basis of the widely-used above-mentioned standards.

As shown by Osheroff et al. [10] and Kawamoto and Lobach [7], service-oriented architectures are considered to be one of the future approaches for CDSSs. In contrast to the SANDS system described by Wright and Sittig [11], our system is based on Arden Syntax. The provided architecture offers a number of convincing advantages, such as the share of existing KBs with other hospitals (by sharing a KB-containing Arden Syntax server by granting remote access to other hospitals, for instance) and the extension of the given solution (e.g., by adding new KBs, data sources, or GUIs) without changing the existing system.

5. Literatur

[1] ADLASSNIG, K.-P., BLACKY, A., KOLLER, W., Fuzzy-Based Nosocomial Infection Control, in: M. Nikravesh, J. Kacprzyk, L.A. Zadeh (eds.), Forging New Frontiers: Fuzzy Pioneers II – Studies in Fuzziness and Soft Computing. Vol. 218, Springer, Berlin 2008; pp. 343–50.

[2] ADLASSNIG, K.-P., BLACKY, A., KOLLER, W., Artificial-Intelligence-Based Hospital-Acquired Infection Control. Stud Health Technol Inform. 149, IOS Press, Amsterdam 2009; pp. 103–10.

[3] HEALTH LEVEL 7, The Arden Syntax for Medical Logic Systems, Version 2.7., Health Level Seven, Inc., Ann Arbor, MI 2008.

[4] HRIPCSAK, G., Writing Arden Syntax Medical Logic Modules. Comput Biol Med. 1994;24:331–63.

[5] JENDERS, R.A., CORMAN, R., DASGUPTA, B., Making the Standard More Standard: A Data and Query Model for Knowledge Representation in the Arden Syntax. AMIA Annu Symp Proc. 2003:323–30.

[6] JENDERS, R.A., SUJANSKY, W., BROVERMAN, C., CHADWICK, M., Towards Improved Knowledge Sharing: Assessment of the HL7 Reference Information Model to Support Medical Logic Module Queries. Proc AMIA Annu Fall Symp. 1997:308–12.

[7] KAWAMOTO, K., LOBACH, D.F., Proposal for Fulfilling Strategic Objectives of the U.S. Roadmap for National Action on Decision Support through a Service-Oriented Architecture Leveraging HL7 Services. J Am Med Inform Assoc. 2007;14:146–55.

[8] KOLLER, W., BLACKY, A., BAUER, C., MANDL, H., ADLASSNIG, K.-P., Electronic Surveillance of Healthcare-Associated Infections with MONI-ICU—A Clinical Breakthrough Compared to Conventional Surveillance Systems, in: C. Safran, S. Reti, H. Marin (eds.), Proceedings of the 13th World Congress on Medical Informatics (MEDINFO 2010), Stud Health Technol Inform. 160, IOS Press, Amsterdam 2010; pp. 432–6.

[9] MCGOVERN, J., TYAGI, S., STEVENS, M., MATHEW, S., Java Web Services Architecture, Morgan Kaufmann, Elsevier, San Francisco 2003.

[10] OSHEROFF, J.A., TEICH, J.M., MIDDLETON, B., STEEN, E.B., WRIGHT, A., DETMER, D.E., A Roadmap for National Action on Clinical Decision Support. J Am Med Inform Assoc. 2007;14:141–5.

[11] WRIGHT, A., SITTIG, D.F., SANDS: A Service-Oriented Architecture for Clinical Decision Support in a National Health Information Network. J Biomed Inform. 2008;41:962–81.

[12] WRIGHT, A., SITTIG, D.F., A Framework and Model for Evaluating Clinical Decision Support Architectures. J Biomed Inform. 2008;41:982–90.

Corresponding Author

Karsten Fehre
Medexter Healthcare GmbH
Borschkegasse 7/5, A-1090 Vienna
Email: kf@medexter.com