Towards integration of the occupational diseases differential diagnosis decision support system

A model-driven approach to the development process

A.G. Ivanov
Ecological and Hygienic Researches Laboratory
East-Siberian Institute of medical and ecological researches
Angarsk, Russia
iag2009@yandex.ru

M.P. Diakovich
Business and Management Faculty
Angarsk State Technical University
Angarsk, Russia
marik914@rambler.ru

S.V. Bachvalov
Department of Automated Systems
Irkutsk National Research Technical University
Irkutsk, Russia
bsv@istu.edu

***Abstract* — this study develops the vision on computer diagnostic aid facilities deployment and usage issues due to heterogeneity of unregulated information environment. Missing a consistent integration among necessary informational and computational entities, which are scattered, it's rather impossible to gather case data and to provide decision support service for end users. Even more, settling up such integration from scratch, leads to high maintenance cost, growing with the system complexity. As the problem resolution, we propose a novel requirements-centric computer-aided engineering method for service-oriented systems integration, using a set of model transformations at a ramified MDA (model-driven architecture) cycle. BPMN (Business Process Model and Notation), SoaML (service-oriented architecture modeling language) and a set of Eclipse (available at eclipse.org) Modeling Framework (EMF) approaches are involved. The method is discussed applying to the occupational diseases differential diagnosis support system integration. Further study directions are also denoted.**

Keywords — integration; architecture; modeling; service; transformation; decision support system

I. INTRODUCTION

Information systems usage widens daily, inducing further need for interoperability. After decades of development, distributed object communication technologies reached their present state of the art at the service-oriented paradigm. Applicable universally, it gives new capabilities, being introduced to the field of medical computer applications. Constant growth of service-oriented architecture (SOA) deployments number could be seen in medical domain, as corresponding researches continue [1,2]. Among the set of functions, disease differential diagnosis decision support task is an especially sophisticated one. It demands diverse data to produce learning samples and to classify cases. And some SOA solutions are being elaborated for that task support [3,4]. When it additionally comes to assess the occupational factor impact on the cause of a person's disease, even more outside data entities are required by the decision support system to access. Although there are mature and feature-rich enterprise service buses (ESB) and integration suites available [5], it's still a challenge to assemble and maintain a distributed software complex. Changes come often; requirements may contradict; plenties of particular SOA vendor approaches are in competition, while analytical and implementation capabilities are always limited. This leads to the lifecycle costs and project's risks increase. We believe a consistent automated engineering method could improve the situation; likewise it was in case of CASE (computer-aided software engineering) technique empowering separate information systems development process. Suchlike methods and frameworks are being elaborated but seem to mostly remain in academic research milestone [6-9]. The approaches commonality is phased requirement implementation through ramified inter-model transformation cycles, obtaining database structure and application components descriptions as the final artifacts. Considering known scientific results and ready to use technologies, we propose a review of our approach to requirement-centered model-driven SOAP (Simple Object Access Protocol) web service-based SOA automated development process, applied for the occupational diseases differential diagnosis decision support system integration.

II. METHODS AND TOOLS

In order to confidently design and implement a SOA system, as well as to update it according to new requirements, a stable and well defined development process should be applied. Nowadays, most of SOA suite vendors offer such a process [10-12]. Since engineered around a particular proprietary software platform, a vendor-specific process couldn't be reproduced elsewhere. A process is mature but usually closed and provided as-is. The developer hardly can interfere with mechanisms – standard-based or irregular - used to shape out the system. The task is to outline the open and adjustable process based on publicly available standards and technologies. In this flavor, the sought-for process instance could be derived configuring the matter at each stage of the following sequence.

A. Determining process roadmap

At the initial point, it's necessary to identify the process chain graph, denoting the artifact transformation routes. The functional requirements classification scheme seems to be a suitable pattern for that. Discovering metamodels to be involved at each process stage will guide to gather pivotal specifications stack. A significant criterion for an alternative preference here is whether sufficient implementing software tools are available for a particular specification. Thus, as it will be shown below, in our process application, we have laid the Java programming technology in the foundation of the stack root.

B. Adapting lifecycle model

Understanding the process contents gives a viewpoint at the lifecycle configuration needed to run the SOA system development and use. For the commonality of present SOA engineering methodologies, the Rational Unified Process (RUP) model inheritance is noticeable. In our approach, a similar top-down methodology is introduced, dealing all with requirements at "inception", modeling the system's aspects at "elaboration", performing inter-model transformations at "construction", building and deploying final artifacts at "transition" phase. Iterations could differ at artifacts being processed volume and composition. Configurability and automation are the priorities for the lifecycle development.

C. Functional requirements preprocessing

The mandatory discipline and a special one for the functional requirements preprocessing is the classification among technological types of the SOA system final artifacts. The next steps of the sequence being described, from D to H inclusively, are dedicated to those particular types. Some extra documentation and refinery analysis could be applied to the requirements in form of building UML (Unified Modeling Language) system representations, if needed. But we believe that it also suitable to compose system artifact models directly just after the requirements were thoroughly classified.

D. Generating and modifying databases

The essential requirements give knowledge concerning the data structure behind the SOA system components. Being reflected in form of class UML diagrams, it could be straightforwardly transformed into the source code, and then to the databases objects through an object-relational mapping (ORM) procedure. In our approach, we rely on Ecore metamodel provided within EMF as a kind of a "Meta-Object Facility" (MOF) compatible mean for defining object models. Perceptible closely like UML class diagrams through "Ecore tools" project (available at http://www.eclipse.org/ecoretools), those models could be in further handled with either Teneo (available at https://wiki.eclipse.org/Teneo) or Texo (available at http://wiki.eclipse.org/Texo) persistence frameworks, to produce ORM-ready Java classes. In case of Teneo, they are backed up with high-level CRUD/IRUN (operations profile as "Create, Retrieve, Update, Delete" for entity instances, "Insert, Retrieve, Update, Nullify" for entity attributes) API (Application Programming Interface), simple for use but introducing some extra dependencies to all codes where the production entities have to be included. In turn of Texo, the classic POJO (Plain Old Java Object) fashioned units are prepared, but are for substantive injecting into a Java-based persistence application, by any mean. Having the persistence environment set up, any database structure alteration required is performed instantly through the Ecore model modification. We use Hibernate (available at http://hibernate.org/) as an ORM engine, while Eclipselink (available at http://www.eclipse.org/eclipselink/) is available for the purpose, too.

E. Building web applications

Eventually, in a SOA-specific development process we do not anticipate to have a strong need for special preliminary modeling of web applications, since the main business logic is to be reflected at the business process and web service system's perspectives. Moreover, we already have essential API-s for the web applications after applying stage D to each SOA system component. We look forward for adopting "Tapestry" (available at http://tapestry.apache.org) framework or an analogue for building web applications interfacing the database objects with CRUD/IRUN and select query operations upon the persistence API, while "Vaadin" (available at https://vaadin.com) framework seems to be a superior choice for web applications in front of web services. Although, to enhance loose-coupling feature of the SOA system, the last mentioned type of web applications is to be preferred, mediating all the CRUD/IRUN scenarios via sided XML (eXtensible Markup Language) or via digital spreadsheet files.

F. Designing business processes

The sense of business processes in a SOA system delivers its permissible behaviors. It is straight to describe a business process via BPMN 2.0 (Business Process Model and Notation) Object Management Group (OMG) specification (available at http://www.omg.org/spec/BPMN/2.0/) for now as it could be considered both as a distinct documentation and as a compliant Business Process Management (BPM) engine executable scenario. We make use of the Activiti BPM engine and designer tool in order to describe and execute business process models invoking and orchestrating the web services and being initialized by them. Besides that, BPMN model is a natural

source of information needed for corresponding web services description by means of SoaML specification (available at http://www.omg.org/spec/SoaML/1.0).

G. Outlining web services

Among approaches to web service design, the SoaML is to get in focus as the only subject OMG open specification available at the moment. Even being reasonable criticized [13], it neatly bridges in between of business process perspective and the web services source code java (or any else) implementation. Having three specifications as the sufficient formal metamodels, it will be quite straightforward to sequence corresponding transformations, supporting the requirements automatically at a significant part. To do so, we advice to include "Obeo Designer" (available at http://www.obeodesigner.com) software tool into development configuration. Based upon Eclipse-specializing "Sirius" (available at https://eclipse.org/sirius/index.html) visual workbench, it designed to handle random specifications as XML-based Domain Specific Languages (DSL-s), furnish them with rich graphical representations. Incorporating Atlas Transformation Language (ATL) (project is available at https://eclipse.org/atl), the tool provides the M2M (Model-to-Model) transformation capabilities to be used for BPMN-to-SoaML procedure, following by the SoaML-to-Java code conversion with M2T (Model-to-Text) technique by means of "Acceleo" subproject (https://eclipse.org/acceleo). As the implementation technology for web services the Apache CXF (https://cxf.apache.org) framework is selected.

H. Preserving system communications

At the root of a SOA system communication there is a router, providing files and messages transfer among the system components, forming out a complete "mail" facility for them. What also important about SOA routing, it is the way to maintain inner system namespace of communicating components as endpoints – web services, databases, applications and more, avoiding direct dependencies from the physical infrastructure addressing details. In our approach, the Apache Camel (http://camel.apache.org) integration framework routing capabilities are used to establish the unified communication environment for our SOA system. Each route in the system is described as a transfer chain of Enterprise Integration Patterns (EIP) [14] involved between sending and receiving endpoints. For Camel, composition of such a chain could be supported with either Redhat "Fuse Integration Editor" (available at http://tools.jboss.org/features/apachecamel.html) or Talend "Open Studio" (available at https://www.talend.com/products/talend-open-studio), or even could be coded without any graphical accompaniment.

I. Production Deployment

Producing final artifacts obtained both manually and by M2T transformations, generally means deployment to the preconfigured environment. In special cases, a build scenario should be prescribed for Apache Maven (https://maven.apache.org) software tool to bring the SOA system components in right places and make them ready to function together.

J. Integration testing and quality assurance

Particular development iteration is finalized with a set of quality assurance procedures, including integration testing of the SOA system deployed. The primary idea behind the testing is to detect SOA inconsistencies by running tests, querying web services with appropriate SOAP messages, and even stressing the system with a flow of generated queries. The "grey box" testing strategy [15] is considered the most suitable for web service based SOA systems, since the WSDL (Web Service Description Language) interfaces are the only available resource for tests composition. The "WebInject" (http://www.webinject.org) software tool can be recommended as a SOA integration testing framework.

III. Results and discussion

Testing the proposed SOA system development process ideas, we do settle them at our occupational diseases differential diagnosis decision support system (DSS) [16] integration. Initially built as a desktop java application backed with relational database and electronic spreadsheets, it's open to adapt in form of a java runtime library, deployed "as is" or within a container, and addressed through web services as the heart of the SOA system. A container deployment is preferred for task parallelism, though.

A. Work accomplished

Functionally decomposing the DSS domain, we have distinguished among interaction participant entities, available for representing as the set of web services providing and consuming necessary values. The generalized entities external functions profile is given in the table 1.

TABLE I. PARTICIPANTS AND FUNCTIONS

Participant entity	External functions profile
DSS core	Case description acceptance
	Hypothesis provisioning
Data storage	Clinical cases data acceptance
	Data selection terms acceptance
	Data samples provisioning
	Necessary RRI[a] requesting
	Necessary metadata requesting
Metadata storage	Participant delegates identification data acceptance
	Inner and delegated identification data collation provisioning
RRI storage	Data acceptance and provisioning concerning cliniclal process and medical diagnostics
	Data acceptance and provisioning concerning industries, workplaces and labor conditions
	Data acceptance and provisioning concerning regulative documents

Participant entity	External functions profile
EMRS[b] delegate	Case diagnostic hypothesis requesting and acceptance
PMES[c] delegate	Occupational routes and clinical examinations history collation metadata provisioning
SALCS[d] delegate	Industry infrastructure data provisioning
	Occupational routes specifications data provisioning

[a.] Regulatory and reference information

[b.] Electronic medical record system

[c.] Periodic medical examinations system

[d.] Special Assessment of labor conditions system

In parallel, we have identified the development process methods and tools structure for the functions listed above supplement, overlaying the requirements tree with appropriate technologies and facilities as shown in fig. 1.

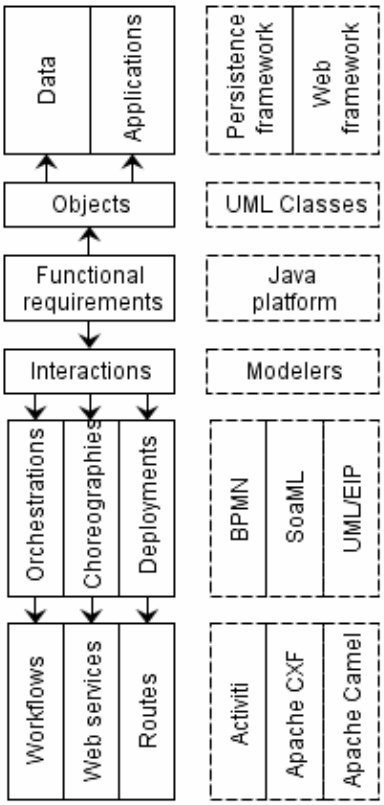

Fig. 1. Technology stack identified by the functional requirements classification

According to the "Objects" part in the scheme presented above, we have successfully settled the requirements processing branch for the databases maintenance behind the storage participant entities. Modeling the data in the form identical to UML class diagrams within "Ecore tools", the Ecore model is being built instantly and then easily transformed to java source code via Teneo framework. Any incremental changes are performed by modifying the Ecore diagram following the transformations repeated. From no to little amount of manual workaround is required afterwards. We also tried to replace Teneo framework with Texo one, meaning facilitate java class dependencies and keep the persistence technologies set at a classical minimum. This way, the POJO-s produced were injected into the system manually, meanwhile it's expedient to do this by a Maven deployment scenario. Both cases, the received API is designed to be used at the side of corresponding storage participant's edge web services or applications to access databases with CRUD/IRUN operations.

B. *Work in progress*

Committing the integration requirements, we start at describing the business processes in BPMN 2.0 via Activiti Designer tool. Designed, such models could be executed by means of Activiti engine, performing orchestrations for the web services. To get the basis for outlining them, we configure the technique of "BPMN-to-SoaML" transformation, focusing on rule principles given in [17]. The illustrative example below (fig. 2 and fig. 3) provides a contracted view at the way the participants' activities correlates with how web services are provided and requested.

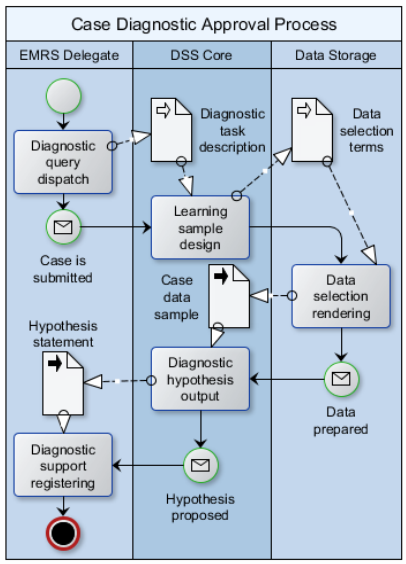

Fig. 2. Case diagnostic approval business-process fragment

The entire SOA system is primarily communicated in a message-driven manner. The messages being delivered through Apache Camel routes and handled by Activiti engine, instantiating business processes requested. The ActiveMQ facility of Apache Camel is used for delivering messages among signal queues designed one per a system function (see table 1.). Other Camel routes are used for transferring data files, and initiated from within business processes. Once a diagnostic query is submitted by EMRS delegate, a message is sent to Activiti, engine, invoking the decision support process instance. In its turn, the process does here two things. At first, it initiates the "Camel" route transferring the diagnostic task

description XML file to the FTP (File Transfer Protocol) endpoint to be accessed by DSS Core. At second, it calls the DSS Core web service by sending control message at the certain Camel CXF endpoint. This engages DSS Core to start processing the diagnostic task. Interpreting the task, DSS Core proceeds with designing the case learning sample for the Bayesian belief network it later builds and uses for the hypothesis assessments. As an intermediate result, the case selection terms are formulated and sent to the Data Storage. Since this activity predefined at the DSS Core web service inner logic, there is no need to orchestrate it from business process but the direct choreographies between DSS Core and Data Storage web services seem to be a better solution. While Data Storage is busy with data sample rendering, DSS Core can participate in any other activities asynchronously, until the message of data sample readiness comes from Data Storage, along with the case data sample for learning a Bayesian network, then using it to assess the hypotheses. Done with this, DSS Core sends the hypothesis statement and a control message to the EMRS delegate as the task response, using the mechanisms discussed above. The business process is finished after the proposed diagnostic hypothesis is registered by the EMRS delegate.

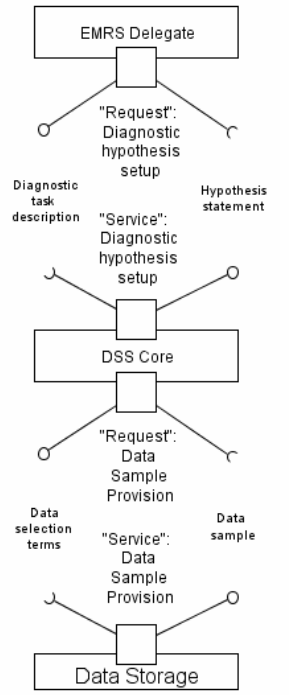

Fig. 3. SoaML service participant diagram fragment

A persistent design problem is how to combine the orchestrations, choreographies and messaging, providing the best configuration for SOA system entire activity. There isn't the univocal decision, although a certain one influences much on the transformation strategy between BPMN and SoaML SOA representations. We use "Obeo Designer" to reproduce them as DSL-s in accordance to official specifications and within the parts needed for further transformations.

C. *Future work*

Progressing with the process design and implementation, we still have much work to complete. As we have searched an accessible software tool for SoaML modeling to incorporate into development environment, producing pure standard-compliant XML model representation accompanied with a graphical workbench, the search gave no acceptable results. Having "Obeo Designer" as the SoaML metamodel constructor, we look forward to introduce a graphical appearance for the DSL by means of the same tool. The ATL and Acceleo transformation scenarios are to be improved and developed forth. The BPMN, SoaML and Java metamodels are for progressive extension, as the SOA systems produced complicate. Herein, it's strongly demanded to provide coherence with prevalent reference and legal systems for timely updating the corresponding RRI storage resources. Another question to be resolved - are there theoretical correlations between SoaML elements and EIP chains sufficient to establish a task of Camel routes base derivation from SoaML representations via an M2M transformation. At implementation level, it's important to assess the process migration prospects and conditions from relying on separate SOA infrastructure facilities to the deployment at an open source enterprise service platform like ServiceMix (https://servicemix.apache.org).

IV. CONCLUSION

In this paper we have presented a novel approach to a SOA system development process, based on the MDA technique, involving transformations in between of BPMN 2.0, SoaML 1.0. and Java specifications. Trying out the process described above applied in the field of occupational diseases differential diagnosis decision support, to develop corresponding integrated information technology, we observe the promised features at weaving the sought-for SOA system in a unified way with manual development amount noticeably reduced. The production system adapts the existing electronic medical record system for communicating to the decision support system via dedicated web services, user transparently. Running lifecycle from the requirements to the final artifacts, instantly keeps the model documentation actual for the continuous improvement. As the SOA systems linear volume is on to increase, the complexity growth is suppressed by rearrangement and organizing of any new communications in domain of routing, messaging and web services. By other hand, the process is knowledge-based and demands qualified support for maintenance and elaboration. The scheduled future work is necessary for the further process refinement.

REFERENCES

[1] Yang, Tzu-Hsiang, Yeali S. Sun, and Feipei Lai. "A scalable healthcare information system based on a service-oriented architecture." Journal of medical systems 35.3 (2011): pp. 391-407.

[2] Ko, L. F., Lin, J. C., Chen, C. H., Chang, J. S., Lai, F., Hsu, K. P., Hsieh, S. L.., "HL7 middleware framework for healthcare information system." IEEE Healthcare 1 (2006): pp. 50-165.

[3] Chang, Chung C., and Hsueh, Ming Lu. "A SOA-based medical diagnosis decision support system using the Bayesian theorem and web service technology." Journal of the Chinese institute of engineers 32.7 (2009): pp. 923-930.

[4] El-Bathy, N., Azar, G., El-Bathy, M., & Stein, G. (2011, May). Intelligent information retrieval lifecycle architecture based clustering genetic algorithm using SOA for modern medical industries. In Electro/Information Technology (EIT), 2011 IEEE International Conference. pp. 1-7

[5] Olsson, Joakim, and Johan Liljegren. "Examining current academic and industry Enterprise Service Bus knowledge and what an up-to-date testing framework could look like." (Bachelor Thesis in Software Engineering, School of Computing Blekinge Institute of Technology, 2012). Available from www.diva-portal.org/smash/get/diva2:831084/FULLTEXT01.pdf

[6] Cherkashin, E. A., Paramonov, V. V., Fedorov, R. K., Terehin, I. N., Pozdnyak, E. I., & Annenkov, D. V. Information Systems Framework Synthesis on the Base of a Logical Approach. E-society Journal, 1. Available from http://tfzr.rs/esociety/issues/eSocietyVol3No2.pdf#page=6

[7] Wirsing, M., Hölzl, M., Koch, N., Mayer, P., & Schroeder, A. (2008). Service engineering: The sensoria model driven approach. Proceedings of Software Engineering Research, Management and Applications (SERA 2008), Prague, Czech Republic. Available from d3s.mff.cuni.cz/sera/files/WirsingSERA2008-InvitedTalk.pdf

[8] Delgado, A., Ruiz, F., de Guzmán, I. G. R., & Piattini, M. (2011, June). Business process service oriented methodology (BPSOM) with service generation in SoaML. In Advanced Information Systems Engineering. pp. 672-680. Springer Berlin Heidelberg.

[9] Traore, B. B., Kamsu-Foguem, B., & Tangara, F. (2016). Integrating MDA and SOA for improving telemedicine services. Telematics and Informatics, 33(3), pp. 733-741.

[10] Zhang, L. J., Zhou, N., Chee, Y. M., Jalaldeen, A., Ponnalagu, K., Sindhgatta, R. R., & Bernardini, F. (2008). SOMA-ME: A platform for the model-driven design of SOA solutions. IBM Systems Journal, 47(3), pp. 397-413.

[11] Gaur, H., Zirn, M., & Subramaniam, S. (2010). Oracle Fusion Middleware Patterns. Packt Publishing Ltd.

[12] SUN Microsystems, “SOA RQ methodology - A pragmatic approach”, available from http://www.sun.com/products/soa/soa_methodology.pdf

[13] Michael Poulin, “SoaML is about everything but SOA”, available from http://www.ebizq.net/blogs/service_oriented/2009/06/soaml_is_about_everything_but_soa_part_1.php

[14] Hophe, Gregor, and Bobby Woolf. "Enterprise Integration Patterns." Designing, Building, and Deploying Messaging Solutions. Boston et. al.: Addison-Wesley (2003).

[15] Jehan, S., Pill, I., & Wotawa, F. (2013, March). SOA Grey Box Testing--A Constraint-Based Approach. In Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on (pp. 232-237).

[16] Ivanov, A. G., Dyakovich, M. P., & Bachvalov, S. V. (2014). “Decisioning support in occupational chronic neurointoxications differential diagnosis provided within a clinical information system”. Applied and Fundamental Studies, 34 (pp. 34-39).

[17] Elvesæter, B., Panfilenko, D., Jacobi, S., & Hahn, C. (2010, October). Aligning business and IT models in service-oriented architectures using BPMN and SoaML. In Proceedings of the First International Workshop on Model-Driven Interoperability (pp. 61-68). ACM.