PROCESS-ORIENTED INCIDENT REPORTING IN HEALTH CARE ORGANIZATIONS
Web Application Architecture and Modeling Experiences

Martin Schmollinger[1], Eric Stricker[2], Tibor Bazlen[1], Thomas Eßlinger[1], Christopher Röhl[1] and Brian Walter[1]
[1]*School of Informatics, Reutlingen University, Reutlingen, Germany*
[2]*Department of Anaesthesiology and Intensive Care Medicine, University Hospital Tübingen, Tübingen, Germany*

Keywords: Process-oriented system architecture, Medical incident reporting, BPMN 2.0, REST web services.

Abstract: In Germany more than 17,000 people die per year because of "medical errors", in the USA it is about 98,000. The number of avoidable malpractice cases or major complications is about tenfold higher. Nearly everyone will be a patient in a hospital sooner or later. A key concept to improve patient safety are modern, anonymous adverse event reporting systems in hospitals. These are known as incident reporting systems. The introduction of modern process-oriented technologies optimizes the efficiency of such systems and increases patient safety. IT saves lives, if it is possible to improve the analysis of medical incidents and to accelerate the detection of the underlying causes by optimizing the IRS processes. In this paper we present a generic process-oriented architecture and its application for incident reporting systems in health care organizations. Further, we formulate lessons learned concerning business process modeling and implementation for process-oriented architectures.

1 INCIDENT REPORTING IN HEALTH CARE

Since the report "to err is human" published by the Institute of Medicine (IOM) (Institute of Medicine, 1999), it is well known that up to 98,000 people die each year because of medical errors. In many countries medical errors are among the ten leading causes of death (Kohn, 2006). The number of avoidable major complications is about tenfold higher. Cases with injury to the patient are only the tip of the iceberg. The avoidance of medical errors increases patient safety. Unfortunately, there is little knowledge about the error mechanisms in health care. A main reason for errors in medicine is the failure to apply medical knowledge in imperfect, real world conditions of patient care and not the missing medical knowledge itself (Rall and Gaba, 2005).

Incident reporting systems (IRS) facilitate the continuous collection of data for a systematic analysis. Presently available systems do not collect the necessary amount and quality of data. The amount of unreported and therefore not systematically analyzed incidents is estimated to be between 50% and 96% (Institute of Medicine, 1999; Leape, 1994; Cullen et al., 1995).

Furthermore, ten years after the before-mentioned IOM report, medical IRS still have major problems to induct a fast realization of useful recommendations. Because the analysis of the incidents is not comprehensive enough, decisions are often made without sufficient knowledge of environmental conditions and various other factors. These quick fixes can have unintentional side effects and even worsen the situation (O'Reilley, 2009). Reported incidents are a window to the surrounding health care system (Vincent, 2006). Looking into this window gives the chance to detect dangerous constellations which may be the root causes of the reported incidents. Sometimes such factors do not actually correspond directly to the reported incident, but without this reporting no one would have ever questioned the detected nuisance. Eliminating the deeper problem behind the incident means to improve the situation for the given incident and moreover, for similar or dependant situations. In order to reduce the risk of adverse events all underlying factors and their relations have to be observed (Leape et al., 1991). The World Health Organization (WHO) published guidelines for adverse event reporting and learning systems (WHO, 2011). The most important aspects are organizational integration, data security, implementation of change requests and the establishment of some type of learning system. Some characteristics of successful reporting systems can also be derived from Leape (Leape, 2002) and Cohen (Cohen, 2000): e.g. non-punitive, confidential, professional analysis, timely, systems-oriented and responsive.

1.1 Patient Safety Information System

The Tübingen Center of Patient Safety and Simulation (TüPASS) at the university hospital Tübingen develops and administrates the interprofessional IRS PaSIS (Patient Safety Information System) (Dieckmann et al., 2006). PaSIS is widely accepted with more than 70 participating hospitals and about 30 rescue helicopter bases in Central Europe. More than 3000 incidents have been reported. All reports undergo a four-eye anonymization process by neutral domain experts trained in incident reporting and using checklist protocols to prevent any lapses. After that the reports are professionally analyzed in order to find the causes and factors for the medical error. Within this process, it is possible to pose questions to the anonymous reporter. The result is prepared for the different user groups and recommendations are derived from the analysis. The implementation of the recommendations is supported and evaluated throughout the system.

PaSIS is a classical web application based on Apache, PHP and MySQL. The complete incident reporting process for each hospital beginning with reporting step, anonymization, posing questions to reporters, analyzing and suggesting recommendations is hard wired in the program code. Each customization of the system is associated with time-consuming, error-prone manual programming. Moreover, it is not possible to execute processes of different versions at the same time. Changes to the process are very complex, because the data of all pending processes have to be adapted to the new one.

2 PROCESS-ORIENTATION IN INCIDENT REPORTING SYSTEMS

The success of an incident reporting system depends on the optimal cooperation of all its stakeholders. Stakeholders are all kinds of users (medical personnel, domain experts, quality managers of hospitals), developers and administrators. Long-running processes are common in medical incident reporting systems. Ideally, the process behind an incident report is adapted to the corresponding hospital, because they differ in size and organization. A precondition for that is the cooperation between clinical personnel or managers and IT specialists responsible for the IRS. Business process management (BPM) technologies and methodologies are the key to successfully fulfilling this task. BPM is a systematic approach to capture, arrange, document, measure, monitor and steer automatic and non-automatic processes in order to reach the goals of the organization (van der Aalst et al., 2003; European Association of Business Process Management,). A business process management system (BPMS) supports the BPM life cycle, which consists of the four major steps: process design, system configuration, process enactment and diagnosis (van der Aalst et al., 2003). The life cycle supports continual improvement of business processes. Modern approaches provide methods and technologies to get business people naturally involved in the BPM life cycle. First of all, graphical notations like OMG's Business Model an Notation (BPMN) (Object Management Group, 2011; White and Miers, 2008) can be used to design process models without programming skills. The resulting business process models can be augmented by technical details ideally using the same notation. The execution of the detailed process model is done by a process engine that coordinates user interaction with the running processes by a web application (Ouyang et al., 2009). Although, there is an increasing need in health care to support patient-oriented processes, BPM technology is rarely used. Especially for IRS web applications, the embedding of process-oriented technologies has several advantages over conventionally implemented web applications. Namely, improved transparency of IRS processes, complete involvement of all IRS participants in the processes's life cycle, easier maintenance and extendibility, faster "time to market" for individual IRS processes of new participating hospitals, flexible change management, support of several process versions per hospital and enhanced process monitoring capabilities that help improving reaction times.

In (Schmollinger et al., 2011), we proposed a general methodology to systematically incorporate process oriented technologies into web applications. Furthermore, we sketched a process-oriented target architecture based on open source products and showed how to implement processes with an adequate tool chain using a simplified incident reporting process.

The used modeling strategy provides three types of models. First, the strategic model. It describes the main participants (user roles and systems) and the order of their main activities. It disregards all the special cases and exceptional or error conditions of the real process. Second, the operative process model is much more detailed than the strategic process model. It describes all operative tasks of all process participant (human beings, as well as systems) and their relations and dependencies. Third, the technical process model contains all technical details and can be executed on a process engine that is embedded in the web application. As proof of concept, we realized a simplified process (”happy path”) that gave us precious insights and confirmed the proposed methodology.

Practical use showed two main drawbacks of the proposed architecture. First, the integration of the process engine into the web application is too complex. The PHP application communicates with a JEE web application using HTTP. The JEE application uses the Java API of the process engine (JBOSS jBPM (JBoss, 2011)) to communicate with the processes. Further frameworks were necessary to realize a seamless integration e.g. W3C’s XForms standard was used to describe web forms and Orbeon was used to render these forms. One objective of the process-oriented approach is to simplify maintenance and extensibility. But the architecture we sketched was too complex to meet this objective. The effort in managing the architecture is too high and too many skills are necessary to create new processes or to update the current versions.

Another problem with the architecture is the use of different process modeling notations for the design of the business process and for executable technical process model. jBPM works best with the native jPDL notation. Hence, there is always an additional step of translation, because the common process design with the users is done with BPMN. In order to improve the overall architecture, we propose the following two actions: first, use ”light-weight” integration of the process engine into the web application using REST web services (Fielding, 2000); second, use BPMN 2.0 for the design and for the execution on the process engine.

Concerning process modeling, the remaining tasks were to turn the strategic model into an accurate operative model. Then, the operative model itself has to be turned into a technical process model that can be executed on the process engine of choice.

In the following two sections, we show improvements of the process architectures and lessons learned in the area of process modeling in health care.

3 A PROCESS-ORIENTED WEB APPLICATION ARCHITECTURE

The Activiti-Project (Activiti, 2011) is an open source BPM platform with a BPMN 2.0 process engine for Java as its core. Furthermore, Activiti includes a REST web service API to the engine that can easily be used by web applications and therefore reduces the complexity of the BPM integration into web applications dramatically. No additional user interface technology is necessary, it is completely up to the web application how to display the process information. This reduces the effort in customizing the application for new modeled processes. The resulting new architecture for PaSIS is sketched in Figure 1. Besides the engine, Activiti contains several tools supporting the BPM life cycle, especially for modeling technical (Eclipse designer) as well as non-technical business processes (Activiti Modeler). Two databases are used. The first one stores all information of the web application such as login data. The Activiti database stores process information and user data of running process instances. After completing a process instance the user data is transferred back to the PaSIS database for archiving purposes.

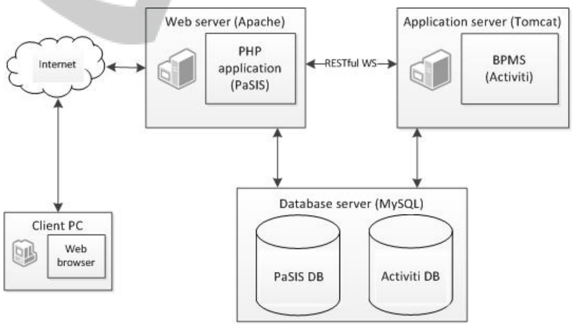

Figure 1: A process-oriented web application architectue using a process engine and REST web services.

As already mentioned, Activiti implements a REST interface, which offers extensive possibilities for controlling the engine. These include the starting of processes, completing tasks and reading of data from the process engine. Information is transferred in the JavaScript Object Notation (JSON), which is much simpler than XML and has a smaller grammar. JSON objects were transferred into multidimensional PHP arrays by native implemented PHP functions. Vice versa, we used cURL functions to wrap information, entered by the user of the web application, for sending them to Activiti. For future reusability new PHP functions were implemented, which simplify the invocation of the built-in REST methods.

4 MODELING EXPERIENCES

In this section, we want to address the task of modeling processes for the sketched architecture. It is out of scope of this paper to discuss the process content itself e.g. the details/tasks of incident reporting. Our focus lies on the modeling methodology.

Due to the strategic model (see (Schmollinger et al., 2011)), we have created a common understanding of incident reporting in health care organizations among the development team. This model helps to understand the core of the process by ignoring complex aspect like e.g. realizing 4-eyes principles, several iterations of discussions between external experts and hospital delegates or call backs to the anonymous reporter. It is the task of creating the operative process model to address all those details.

In order to model the operative process out of the strategic model, we had to analyze the actual incident reporting system and the interaction with the process participants thoroughly. A successful strategy is to model the view of each process participant separately in a single pool. The pool representing the process engine contains all tasks that are necessary to coordinate the other process participants. In our case, we defined three pools. One for the participating hospital and within this pool one lane for the anonymous reporter and one lane for the quality manager. One for the process engine (in the middle) and finally one for the incident reporting organization with its experts for deidentification and analysis. The message flows into the process engine's pool represent REST web service calls from the web application to Activiti.

In general, the operative process model is always huge and complex, because it represents a large part of the system's logic. Our resulting operative process model consists of three pools integrating 6 swimlanes, 15 activities spread over the swimlanes, 11 gateways, 19 events and several sequence and message flows. A precise description together with a complete illustration is out of scope of this paper. We focus on the modeling methodology and show the relation between the operative and the technical process model (see Figure 2).

Ideally, the process engine pool is very close to the technical process model that is finally deployed to the process engine of the process-oriented architecture in order to facilitate the realization of the BPM round trip. The resulting technical process model is shown in Figure 3. It can be easiliy seen that in our case the technical process model really looks similar to the process engine pool in the operative model. This model is optimized for execution in the Activiti BPM engine. Each human task is assigned to a process participant as defined in the operative model. In addition all elements which were not required, such as notice-elements, were deleted because the engine does not need them. Gateways within the sequence flow use process variables (most of their values are generated by user interaction) in order to decide the route of the sequence flow. The experiences made led to the following *lessons learned:*

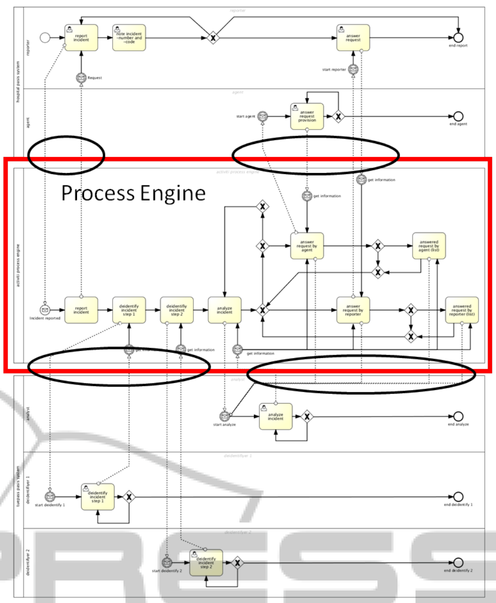

Figure 2: Overview of the operative process model of the entire incident reporting process. The pool in the middle emphasized by the rectangle represents the process engine. This part of the operative model is the base for the implementation of the technical process model that is executable on the target process engine. The ellipses mark message flows between the process engine pool and the other pools. The message flows have to be implemented by service calls from the web application to the process engine.

1. **Optimization and Adaption of the Original Web Application is Recommended.** Although the original web application might be instrinsically process-oriented, it is very complex to extract these processes. In general most applications grew historically and include several workarounds or bypasses for special cases and customers. Sometimes these construct are de facto not used anymore. Modeling all these constructs would lead to even more complex models without an additional benefit. Hence, we recommend to clean up and simplify the system while modeling. It is worth questioning suspicous constructs in order to otimize the new models.

2. **Redefinition of Authorization.** Process-oriented systems need a clear mapping of tasks to process participants. Tasks are assigned to roles. A role

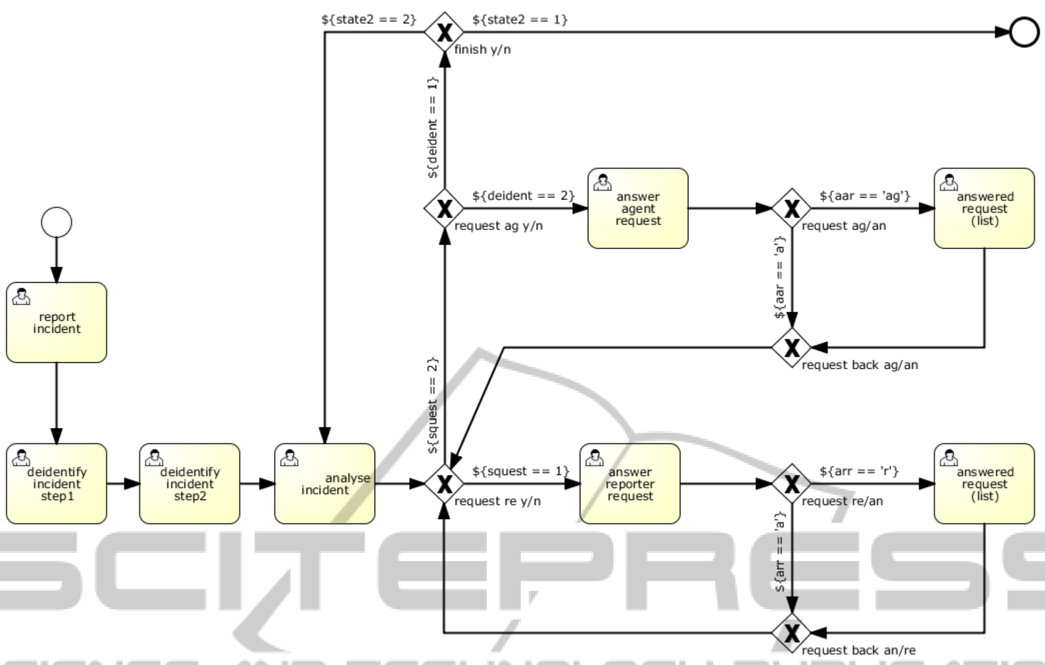

Figure 3: Technical process model that is executable on the process engine of Activiti. The labels at the outgoing sequences at the gateways are conditions using process variables. The values of these variables are generate by user input during the process execution. The model mainly consists of human tasks that are executed by service calls from the web application. The participant's todo-lists in the web application can be updated by reading the actual state of the process instance.

can be taken by single users or user groups. There are web application that have a good authentification, but a very crude authorization. In these case, it is not the system that ensures that the right people do the right things at the right time, but the organization of the daily work (they know themselves what they are allowed to do or not). Without a distinct authorization, the application is more flexible because nearly everyone can do everything. But on the other hand side, it is not possible to manage accurate todo-lists and there is always the danger of misusage. Therefore, we recommend to redesign the authorization system while building up a process-oriented system.

3. **Use of Modeling Conventions.** During process modeling it happens that the same construct can be modeled in several ways. In order to receive similar models over time it is recommended to define modeling conventions, use process patterns or general accepted guidelines like (Mendling et al., 2010).

4. **Know your Process Engine.** There are a lot of process engines around that pretend to be a BPMN 2.0 compatible engine. BPMN 2.0 is a very mighty standard with a lot of new constructs and symbols. If you want to use all its features, you first have to check whether your engine does really support them. Furthermore, if it supports it, you have to check how it is implemented (e.g. parallel gateway: is it really executed in parallel?). In theory a BPMN 2.0 model should run on every BPMN 2.0 process engine. Unfortunatelly, this is not true. You have to consider the capabilities of your engine while modeling the operative and of course technical process model. Otherwise, there will be a high probability that you can not reuse your operative model for the design of the technical model. Although this the wrong way, your engine defines the BPMN subset you can use for your models. Hence, we recommend to define this subset of BPMN before modeling.

5 CONCLUSIONS AND FUTURE WORK

Intrinsically process-oriented web applications, like incident reporting systems in health care are usually implemented conventionally (e.g. LAMP/WAMP). The processes are implicitly realized in the system. This leads to major disadvantages concerning transparency, maintainability and extendibility. The customization of processes is very time-consuming and error-prone. Even if the current process engines are not able to handle the full BPMN 2.0 notation, BPM methodologies and technologies have a high potential to improve this situation. In this paper, we presented a process-oriented architecture that integrates a BPMS into classical web application using REST web services and BPMN 2.0. We implemented the architecture for an incident reporting systems in health care. Further, we showed how to model processes for the given architecture using incident reporting processes as an example and derived several lessons learned from it. The presented approach has great potential, but has to pass the practice test. Besides the future lauch of a production system, further open questions arised.

After establishing process-orientation in our incident reporting system, we want to exploit the new monitoring capabilities of the process-oriented architecture. Future work will be about the design of a management cockpit for incident reporting with the goal to improve reaction times and the quality of the analysis.

The exploration of processes in web applications is a very time consuming task. Process mining is a research area that surveys methodologies and algorithms with which processes of applications can be detected out of log-Files. We are about to realize a software infrastructure that helps collecting process relevant data of clinical web applications in a convenient way over time. Using process mining techniques on this data collection promises to accelerate the exploration of business processes that are implicitly realized by clinical web applications.

Although a predefined incident reporting process is a good orientation for the process participants, in some cases it is necessary to extend or modify the process dynamically. The actual BPM technology approach does not support these so called ad hoc processes. We will survey how social media technology can improve this aspect.

REFERENCES

Activiti (2011). Bpm platform. http://www.activity.org/.

Cohen, M. (2000). Why error reporting systems should be voluntarily. *BMJ 2000*, pages 728–9.

Cullen, D., Bates, W., Small, S., Cooper, J., Nemeskal, A., and Leape, L. e. a. (1995). The incident reporting system does not detect adverse drug events: A problem in quality assurance. *Joint Commission Journal on Quality Improvement*, (21):521–8.

Dieckmann, P., Stricker, E., and Rall, M. (2006). The role of incident reporting systems in the evaluation of medical devices. In *Proceedings of the Annual Meeting of the International Ergonomics Association*.

European Association of Business Process Management. *Business Process Management Common Body of Knowledge - BPM CBOK: Guidelines for Business Process Management*. EABPM.

Fielding, R. (2000). Architectural styles and the design of network-based software architectures. Doctoral dissertation, University of California, Irvine.

Institute of Medicine (1999). To err is human: building a safety health system. *Washington DC: National Academy Press*.

JBoss (2011). jbpm. http://www.jboss.com/products/jbpm/.

Kohn, L. T. (2006). Building a safer health system. *Washington DC: National Acad. Press.*

Leape, L. (2002). Health policy report. patient safety. reporting of adverse events. *NEJM*, pages 1633–8.

Leape, L., Brennan, T., and Laird, N. e. a. (1991). The nature of adverse events in hospitalized patients: Results from the harvard medical practice study i & ii. *New Engl. J. Med.*, I & II:324, 370–384.

Leape, L. L. (1994). Error in medicine. *JAMA 1994*, pages 1151–7.

Mendling, J., Reijers, H., and van der Aalst, W. (2010). Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52(2):127–136.

Object Management Group (2011). Bpmn 2.0 specifications. http://www.omg.org/spec/BPMN/2.0/.

O'Reilley, K. (2009). Patient safety improving slightly, 10 years after iom report on errors. http://www.ama-assn.org/amednews/2009/12/28/prsb1228.htm.

Ouyang, C., Dumas, M., van der Aalst, W., ter Hofstede, A. H. M., and Mendling, J. (2009). From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology, 19*, 19(2).

Rall, M. and Gaba, D. M. (2005). *Millers's Anesthesia*, chapter Human Performance and Patient Safety, pages 3021–3072.

Schmollinger, M., Iwanowski, M., Kußmaul, T., Schwarting, D., Stark, J., Stricker, E., and Rall, M. (2011). A challenge for healthcare web applications: From data- to process-orientation. In *Proceedings of the International Conference on Health Informatics HEALTHINF 2011*. SciTePress.

van der Aalst, W., ter Hofstede, A., and Weske, M. (2003). Business process management: A survey. In *In Proceedings of the International Conference Business Process Management 2003*. Springer Verlag.

Vincent, C. (2006). *Patient safety*. Edinburgh: Elsevier.

White, S. and Miers, D. (2008). *BPMN modeling and reference guide. Understanding and using BPMN - develop rigorous yet understandable graphical representations of business processes.* Future Strategies Inc., Lighthouse Point.

WHO (2011). World alliance for patient safety: Draft guidelines for adverse event reporting and learning systems. http://www.who.int/patientsafety/events/05/Reporting_Guidelines.pdf.