Process Modeling for Requirements Engineering: A Medical System Case Study

Brian Berenbach
Siemens Corporate Technology
Princeton, New Jersey USA
brian.berenbach@siemens.com

Diane Rea
Siemens Healthcare Diagnostics
Newark, Delaware USA
diane.m.rea@siemens.com

Florian Schneider
Technical University Munich
Munich, Germany
florian.schneider@in.tum.de

Abstract. The Unified Requirements Modeling Language (URML) was created to support early systems engineering; permitting the capture of process, potential variation points, hazards, threats and mitigations during the elicitation process. It has been used on Siemens projects, but to date they have been proprietary, precluding public exposure and limiting the critical review needed to refine and improve the language. To that end, a standard medical process, phlebotomy, was modeled in detail to determine the efficacy of the language. One of the authors of this paper is an expert in the area of medical diagnostics and provided the domain expertise. The modeling experience exceeded our expectations, and provided some unanticipated benefits. The results of our study are presented in this paper.

1. Introduction

Siemens Corporation has four business sectors, Industry, Healthcare, Infrastructure & Cities, and Energy. Using a generic SysML [1] or UML [2] modeling tool to capture requirements and processes across these sectors can be challenging, because the two languages do not support certain concepts needed in requirements engineering. Using special purpose languages like BPMN [3] or URN [4] is still not feasible because of missing tool support or missing tool integration. Two of the authors have previously reported on the rationale for and features of a new visual modeling language, the URML and initial experience with its use [5], [6]. In short, the URML has support for modeling requirements, goals, stakeholders, processes, systems, product lines, and features. It thus combines language features that have not been yet combined in a single language. Showing actual project examples was not feasible in those papers as the projects where the URML was used were proprietary. Consequently, a case study was created to model an end-to-end medical process in order to provide an actual example of its use as well as to stress test the language for improving it. Medical processes in general are difficult to model; they involve subjective decisions on the part of the participants (e.g. nurses, doctors, patients) (see figure 1), can involve complex administrative procedures, may have many potential hazards, and are subject to heavy regulation. Luckily, an expert in medical diagnostics was able to participate in creating a “straw man” model of an idealized blood sample management process (figure 2) and participate in the reporting on the model in this paper.

What initially appeared to be a simple and straightforward example, once begun turned into a complex and demanding exercise, a perfect illustration of the difficulty of capturing medical processes. It was clear from the get-go that the URML enabled the modeling effort, without it, capturing the various types of regulatory codes (figures 3 and 4), requirements, and mitigations would have been significantly more difficult. Furthermore the semiotic clarity of the URML allowed full participation of the domain expert in the creation of the model. In
In section 2 we define the phlebotomy process. In section 3 we present views on the modeled phlebotomy process; in section 4 we summarize the authors' observations and conclusions, and in section 5 we suggest possible future research in process modeling with the URML.

Figure 1: A context diagram, showing the clinical chemistry laboratory system and its stakeholders

Figure 2: A use case diagram, showing the first level of decomposition of the blood
2. The Blood Sample Management Process and Phlebotomy

Clinical Laboratory processes can be broken down into three phases: pre-analytical, analytical and post-analytical. The pre-analytical phase begins with the order of the laboratory tests, sample collection, storage, transport, and sample preparation prior to entry.
into a laboratory analyzer. The analytical process begins with entry of the sample into the analyzer and ends with the creation of the test result. The post-analytical phase includes the quality check and release of the test results to the medical professional responsible for patient care and treatment.

The pre-analytical process is greatly influenced by the interaction of patient care providers (nurse, physician), patient and the phlebotomist (technician responsible for drawing the blood sample). Due to these interactions and varied environments in which the phlebotomist performs his or her duty, it is an error prone (i.e. hazards) part of the total diagnostic process. Unfortunately, errors in this phase often go undetected and can affect all downstream activities.

The analytical phase includes testing of samples using reagents and a detection system. For this model, the assumption was made that sample testing was performed using an automated analyzer rather than a manual test procedure. The overwhelming majority of clinical laboratories use an automated analyzer for the bulk of test processing. Due to the prevalence of automated analyzers and the long-standing practice of quality control and performance evaluation testing, error rates in analytical testing tend to be lower than in other phases of the diagnostic testing process.

The post analytical process includes the procedures used to assess the quality of the test result and the timely reporting to the physician. Advances in information technology have greatly improved the speed of reporting and accuracy of test reports in recent years. For many laboratories, use of decision algorithms, delta checks of previous test results and automated reporting (auto-verification) have replaced the labor intensive manual procedures of researching medical records and retesting of samples.

Many types of samples are used for clinical laboratory diagnostic testing. Examples of sample types include blood (whole blood, serum, and plasma), urine, and other body fluids (spinal, synovial, and pericardial). Each type of sample requires a different collection, preparation and storage technique. Our focus in this paper was on the blood samples that are collected by laboratory personnel and the process that is used in preparation of the blood sample for analysis.

Phlebotomy is a process whereby blood is collected from a patient’s veins, arteries or capillaries. Venous blood is obtained through venipuncture using a needle and blood collection tube or syringe. Arterial blood is collected though arterial puncture using a syringe and capillary blood is obtained through skin puncture with a lancet and collection into a micro-container. Although the techniques for each type of blood sample are varied, the elements in the collection process are very similar (highlighted in the model). The key steps in the phlebotomy process include: test order entry, test order receipt, blood collection material preparation, patient identification, blood collection, sample identification, storage and transport.

The quality of healthcare has come under increased scrutiny in recent years, as evidenced by the Institute of Medicine report- “To Err is Human: Building a Safer Health System” [7]. In this report, the authors advocate that healthcare delivery should be examined as a complex system with any one element in the system belonging to multiple systems (i.e. the chemistry laboratory is part of the central laboratory, which in turn is part of the hospital). Understanding the components of a system along with the interactions (both internal and external) and processing steps is an important first step in the improvement of healthcare quality.

When the total laboratory process (from pre-analytical to post analytical) is examined, the pre-analytical phase stands out as the most error prone part of the testing process. Plebani further breaks down the pre-analytical phase into pre-pre-analytical (outside the laboratory confines) and pre-analytical (after specimen receipt in laboratory and before entry into the
The author reports error rates of 46-68% in the pre-pre-analytical phase, 3-5% in the pre-analytical phase, 7-13% in the analytical phase, 13-20% in the post-analytical phase (within laboratory) and 25-46% in the post-post-analytical phase (outside of laboratory jurisdiction).

3. Creating the Clinical Chemistry Laboratory Model

We started modeling by determining the stakeholders of the clinical chemistry laboratory (See Figure 1), and the high-level goals of blood analysis (Figure 5). There is one top-level, fuzzy (or as we call it, soft) goal, expressed by a physician (not shown on the diagram). The physician wants blood analysis to support treatment and diagnosis of the patient. There are several concrete (or as we call them, “hard”) goals that contribute to this goal. As can be seen on the diagram, some of them might be conflicting. This allows an early analysis of the potential future tradeoffs that a systems engineer might have to decide upon.

![Diagram](image)

Figure 5: Some of the high-level goals regarding blood analysis

We then created a hierarchy of processes, by decomposing the high-level processes into lower-level ones, until we arrived at a technical level, where we were able to assign specific sub-processes to subsystems of the hospital. The breakdown of a subset of these processes can be seen in Figure 6. It shows how the blood sample management process consists of a pre-analytical, an analytical, and a post-analytical phase. The pre-analytical phase itself consists of eight major steps, originating from the physician’s order of a blood analysis and ending with the step where the sample is prepared for analysis. On the lowest level shown in the figure, one can see how sample preparation, which could eventually be broken down into even finer-grained steps, might suffer from incomplete centrifugation of samples or incomplete clotting. These two dangers might be mitigated technically through clog detection or non-technically through the observation of laboratory technicians. The URML proved useful here in that it provided concepts for both requirements and procedural mitigations. For handling complexity, we used a feature of the modeling tool that let us create “composite diagrams” (instead of having to hyperlink to a detailed diagram). A composite diagram can be created for every model element that is decomposable. A click on the element leads the user of the modeling tool to the composite diagram. This way, elements of a diagram can be ‘expanded’. A composite diagram may, for example, show the intrinsics of a process or the parts of a structure.
Figure 6: The hierarchical decomposition of use cases. The lowest level shows dangers and mitigations thereof.
As described before, we did not descend until we arrived at atomic (i.e. non-decomposable) processes, but rather stopped at about one level above due to time constraints and the goals of the modeling effort. At this level, we identified dangers the processes were vulnerable to and documented known measures to mitigate them. On some occasions, we had to back up and model the atomic sub-processes of a process later as processes being vulnerable to many dangers were too complex to visualize on a single diagram.

![Diagram](image)

**Figure 7**: Sample processing is vulnerable to instrument malfunction. This can be mitigated technically by instrument function checks or non-technically by quality checks in the laboratory.

(For this paper, we consider a danger to be an “umbrella” term covering both hazards and threats. However, for the phlebotomy process, as we are primarily concerned with hazards and their mitigations, here we use the terms “hazard” and “danger” interchangeably). On a higher level, the dangers were condensed into an abstract danger. Similarly, requirements and mitigating procedures were grouped (Figures 7, 8, and 9).

In figure 7, it can be seen that sample processing in general might be subject to instrument malfunction. While it may be positive to see that there are two possible kinds of mitigations, it does not help very much in the task of formulating unambiguous requirements for a blood analyzer. Also, there are different kinds of malfunctions. To see exactly which kind of danger can occur, we had to decompose processes several levels. First, we inquired about the exact nature of instrument malfunction. As can be seen in figure 8, it is potentially composed of various concrete dangers. Then, we collected knowledge regarding potential requirements to a system that would allow mitigating such dangers. This can be seen in figure 9. Finally, we had to decompose the sample processing use case into sub-use cases to see which kind of instrument malfunction can occur in which process step. Then it can also be determined which kind of requirement would be able to mitigate such a danger. Furthermore, we were able to determine new use cases for the system that extend existing ones in order to provide a higher level of safety. In figure 10, two sub-processes of sample processing are vulnerable to two different kinds of instrument malfunction, which are related to temperature. Requiring the system under discussion to have facilities to perform temperature checks could mitigate both of them. This requirement then leads to the formulation of two new extending use cases that deal with the monitoring and maintenance of temperature inside a blood analyzer device.
Figure 8: Instrument malfunction is a rather abstract danger. This diagram shows how it can be broken down into more concrete dangers.

Figure 9: Possible function checks that could be implemented by a blood analyzer to mitigate instrument malfunction
Figure 10: Both shown sub-processes are vulnerable to dangers related to wrong temperature. Both can be technically mitigated by the requirement to have temperature sensors in the analyzer. The diagram also shows that there might be a future requirement multiple sensors, as they are needed by different use cases.

4. On Using the URML to Model a Medical Process

Difficulties encountered. When modeling systems-of-systems, it is difficult to maintain strict conventions regarding the current point of view of the modeler. When mentally switching between different systems under discussion, the boundaries between systems sometimes blur. While the URML tries to face the problem by distinguishing between environment processes (external to a system) and use cases (internal to a system), experience suggests that further differentiation of internal processes might be needed. Use cases, interpreted in a strict sense would only be processes where an actor (external to the system) uses the system. This would however render opaque internal processes. This is not always desirable, even though requirements engineering should not focus too much on the inner parts of a system (as only the requirements for a system, should be discovered and not the system itself be designed). Sometimes however, a system itself contains other systems that might trigger dangers that pose a danger to the whole system. While this is perfectly possible with the URML, there might be situations where the modelers would want to distinguish between processes with actor involvement (use case in a strict sense, processes that are on the boundary of the system, in other words) and internal processes, because an impact analysis shall be performed. This distinction however is not possible with the current state of the URML. A distinction would also help with handling complexity, as the modeler could decide not to focus on internal processes in early stages of the modeling process, but to recall them in a later state, for the sake of completeness of the model. Complexity is the main challenge when dealing with large models. The chemistry laboratory with its processes has proven to be a complex system. We had to model many details and reorganize them quite often before arriving at useful results regarding analysis of the domain. An open question then is, if a
process or use case decomposes into a set of both processes and use cases, how should it be presented? Since the model started with a hospital process, of which medical diagnostic equipment was a part, we initially used process symbols, and then a mix of process and use case symbols to capture both the interaction of actors with the system of interest, and their activities external to the equipment of interest.

Positive effects of the URML. As figures 7 and 8 suggest, the URML helps with the collection of knowledge from experts: Once we found that there was a specific instrument malfunction, we could ask questions such as “Which other kinds of instrument malfunction do you know of?” Similarly, we could ask for kinds of instrument function checks (figure 8). The main benefit of the URML is however that it allows connecting these pools of knowledge. In a low-level diagram, a concrete use case could be connected to a concrete hazard, mitigated by a concrete functional requirement (figure 10). The URML supports decomposition for most abstract concepts that the analyst may encounter. Thus a process of collecting first and then connecting the identified elements afterwards by relationships of the language is supported.

The relationships between processes and dangers helped the expert condensing existing knowledge regarding problems in the blood sample management process. The connection of hazards and mitigating requirements helps to illustrate how the problems can be addressed from the perspective of a domain expert. The connection of requirements and use cases can then highlight new functions that have the mitigating requirements as a prerequisite. As a result, a single diagram (as in figure 10) can show at once the hazards of a process including their origins, and their technical mitigation(s). This e.g. helped us in confirming the hypothesis that the pre-analytical phase is as error-prone as described in other publications.

5. Future Research into early modeling using the URML

While a modeling language can facilitate dealing with complexity by supporting decomposition relationships, it cannot do the job alone. A tool implementation should support modelers with assistance in navigating multitudes of diagrams and model elements, and easily creating different viewpoints on a model, e.g. making the creation of diagrams easy or by providing filtering mechanisms that are fast to set up and easy to handle. Furthermore, given that models are normally created by more than one person, a standard or guide for model creation is imperative so that the model will have a uniform look and feel across the different sections. Ideally, it should be possible to embed the standard in the tool of choice so that the tool can perform programmatic checking against a modeling standard [9].

It became clear during the creation of the model that some constructs or facilities that were not present in the URML meta-model would have been helpful:

- In a regulated environment many different regulatory codes and regulations may apply to a specific step of a process or use case. Capturing them is challenging; it was found that a rollup mechanism was needed to minimize complexity while capturing the relevant information. One possibility would be a hyperlink to a separate regulatory model; the solution we chose was to have abstract constructs that were decomposed in lower level diagrams. Techniques for capturing the complexity of the regulatory landscape for models created in heavily regulated environments is something the authors will look into.
- Similarly, there were process and use case steps where the modelers were presented with a large number of functional and non-functional requirements, sometimes orthogonal and sometimes interlocking. When this situation was encountered we
initially created abstract requirements as placeholders, and decomposed the requirement sets on a lower level diagram. This is not necessarily the optimum solution, as “fan-in-fan-out” might obscure traceability. Yet to the depth of detail we planned, dropping down further levels was not practical.

Other areas the authors may pursue include a guide or recommendation set for level of detail before switching from requirements to design, and the nature of the requirement/design relationship in a complex model.

6. Conclusions

This paper describes the application of the unified modeling language (URML) to the modeling of a medical process, regarding the management of blood samples in a hospital and one of its facilities, the clinical chemistry laboratory. The model is non-proprietary and thus a little bit general, because it cannot contain information that is under disclosure. While we were able to show much more model excerpts than in previous papers regarding the URML, the full model still could not be exposed. We might address this shortcoming by the publication of a technical report, describing the full model, in the future.

The numerous excerpts of the model in this paper should however satisfactorily show the URML allows the expression of statements that neither the UML nor the SysML (to name two prominent modeling languages) exactly reproduce. The formal connection of hazards to use cases, use cases to regulatory requirements, and hazards to requirements is not present in any other modeling language existing to date. Thus we hope that the URML can support the engineering and reengineering of complex systems better than existing languages, not only in the medical domain.

7. References


8. **Biography**

**Brian Berenbach** is a senior staff engineer with Siemens Corporate Technology. He is an INCOSE certified ESEP and an ACM distinguished engineer. Mr. Berenbach has worked in the field of systems engineering for over 45 years, and has published widely on various SE topics.

**Diane Rea** is a senior manager in requirements engineering at Siemens Healthcare Diagnostics. Her technical background includes assay and instrument requirements development, clinical trial management and assay test parameter development. Prior to employment at Siemens, Diane was a medical technologist for 17 years in the chemistry, hematology, coagulation and blood banking departments of hospital laboratories. She holds a Master of Science in Clinical Chemistry and a Bachelor of Science in Medical Technology.

**Florian Schneider** is a doctoral candidate at Technical University of Munich and a Software Engineer at Linova Software GmbH, Munich. He has worked on several software projects and consulted with the requirements engineering of these projects. He holds a diploma degree in Informatics. His research interests are requirements engineering for software and systems, meta-modeling, modeling languages, and the nature of abstraction.